



p/n 88-019933-01 A

# Gemini GV6K and Gemini GT6K Command Reference

Effective: October 1, 2001

---

INFNC OUTFNC C  
TDIR LIMFNC A LS TSKAX GO WHILE  
TRGFN D PS ONIN OUT IF  
WAIT TPE LH WHEN PCOMP  
STARTP DEF END GOBUF ELSE  
TASF ERASE DEL RESET PRUN SFB  
HOM ENCCNT JOY JOG VAR VARB  
ERROR READ MA FOLEN TOUT MC  
GOWHEN TSTAT FOLMAS DRES DRFEN  
SOFFS TIN TFSF STRGTE TSSF ERES  
V TREV S CMDDIR TSSF  
K GOTO DATSIZ  
TINOF

# IMPORTANT

## User Information



### WARNING



Gem6K Series products are used to control electrical and mechanical components of motion control systems. You should test your motion system for safety under all potential conditions. Failure to do so can result in damage to equipment and/or serious injury to personnel.

Gem6K Series products and the information in this user guide are the proprietary property of Parker Hannifin Corporation or its licensors, and may not be copied, disclosed, or used for any purpose not expressly authorized by the owner thereof.

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

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 the equipment or this user guide.

© 2001, Parker Hannifin Corporation  
All Rights Reserved

Motion Planner and Servo Tuner are trademarks of Parker Hannifin Corporation.  
Microsoft and MS-DOS are registered trademarks, and Windows, Visual Basic, and Visual C++ are trademarks of Microsoft Corporation.

### Technical Assistance Contact your local automation technology center (ATC) or distributor, or ...

#### North America and Asia:

Compumotor Division of Parker Hannifin  
5500 Business Park Drive  
Rohnert Park, CA 94928  
Telephone: (800) 358-9070 or (707) 584-7558  
Fax: (707) 584-3793  
FaxBack: (800) 936-6939 or (707) 586-8586  
e-mail: tech\_help@cmotor.com  
Internet: <http://www.compumotor.com>

#### Europe (non-German speaking):

Parker Digiplan  
21 Balena Close  
Poole, Dorset  
England BH17 7DX  
Telephone: +44 (0)1202 69 9000  
Fax: +44 (0)1202 69 5750

#### Germany, Austria, Switzerland:

HAUSER Elektronik GmbH  
Postfach: 77607-1720  
Robert-Bosch-Str. 22  
D-77656 Offenburg  
Telephone: +49 (0)781 509-0  
Fax: +49 (0)781 509-176



Technical Support

E-mail: [Tech\\_Help@cmotor.com](mailto:Tech_Help@cmotor.com)


# Introduction


## Purpose of this Document

This document is a reference for all the Gem6K Series commands. To gain a full understanding of how the Gem6K Series commands are used together to implement specific features, refer to the *Gem6K Series Programmer's Guide* (p/n 88-019934-01). For hardware-related information (e.g., electrical wiring connections, specifications, tuning, etc.), refer to the *Gem6K Series Hardware Installation Guide* (p/n 88-019932-01). Information on the Gem6K communications server (COM6SRVR) is provided in the *Communications Server Programmer's Reference*.

**NOTE**

The contents of this document are available from the Motion Planner help system. To access the help system, use the Info window (right), or press the F1 key.

 This document is also available in Adobe Acrobat PDF format from our web site:  
<http://www.compumotor.com>



The screenshot shows a window titled "Motion Planner - [Editor1]" with a menu bar (File, Edit, Debug, View, Communication) and a toolbar. Below the toolbar is a help window with a tree view containing the following items:

- Welcome, 6K user!
- 6K Program Objects and Wizards
- 6K Programming -- Introduction
- Gemini Programming -- Introduction
- 6K Command Language
- Gemini Command Language
- Gemini6K Programming -- Introduction
- Gem6K Command Language

## Table of Contents

Page 1	<i>Introduction:</i> <ul style="list-style-type: none"><li>Command Description Format</li><li>Syntax -- Letters and Symbols</li><li>Syntax -- General Guidelines</li><li>Syntax -- Command Value Substitutions</li><li>Programmable I/O Bit Patterns</li><li>Programming Error Messages</li><li>S-Curve Accel/Decel Profiling</li><li>Units of Measure and Scaling</li></ul>
Page 23	<i>Special Programming Characters:</i> Operator symbols for multi-tasking, math and bitwise operations, relational expressions, bit select operations, etc.
Page 39	<i>Command Descriptions:</i> Operator symbols are described first, followed by the rest of the Gem6K Series commands in alphabetical order.
Page 343	<i>Appendix A: Gem6K Series Command List:</i> Alphabetical list of all Gem6K Series commands.
Page 349	<i>Appendix B: ASCII Character Table</i>
Page 351	<i>Index</i>

# Description of Format

1.	2.	3.
<b>INEN</b>	<b>Input Enable</b>	
4. Type	Inputs or Program Debug Tools	<b>Product</b>
5. Syntax	<!><B>INEN<d><d><d>...<d>	GT6K
6. Units	d = 0, 1, E, or X	GV6K
7. Range	0 = off, 1 = on, E = enable, X = don't care	<b>Rev</b>
8. Default	E	6.0
9. Response	INEN: *INENEEEE_EEEE_EEEE_EEEE_EEEE_EEEE_EEEE	6.0
10. See Also	[IN], INFNC, INLVL, INPLC, INSTW, TIN, TIO	

Item Number	Description
1.	<b>Mnemonic Code:</b> This field contains the command's mnemonic code. If the command is in brackets (e.g., [ IN ]), it is an operator that must be used within the syntax of another command (e.g., IN may be used in a conditional expression like IF( IN . 3=b1 )).
2.	<b>Full Name:</b> This field contains the command's full name.
3.	<b>Valid Product &amp; Revision:</b> This field lists the Gem6K 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 <b>Rev</b> is specified as "n/a".  You can use the TREV command to determine which product revision you are using. For example, if the TREV response is *TREV92-016740-01-6.0, the product revision is 6.0.
4.	<b>Type:</b> This field contains the command's type.
5.	<b>Syntax:</b> 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 <i>Syntax</i> sections below.
6.	<b>Units:</b> This field describes what unit of measurement the parameter (b, d, i, r, or t) in the command syntax represents.
7.	<b>Range:</b> This is the range of valid values that you can specify for an argument (or any other parameter specified).
8.	<b>Default:</b> 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.	<b>Response:</b> 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 (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.	<b>See Also:</b> Commands related or similar to the command described are listed here.

# Syntax – Letters and Symbols

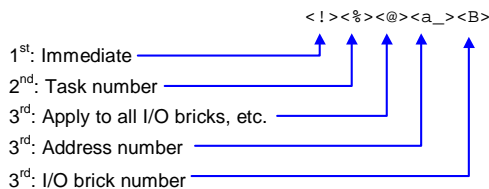
The command descriptions provided within this manual use alphabetic letters and ASCII symbols within the **Syntax** description (see example below) to represent different parameter requirements.

INEN		Input Enable		Product	Rev
Type	Inputs or Program Debug Tools				
→ Syntax	<!><%><B>INEN<d><d><d>...<d>			GT6K	6.0
Units	d = 0, 1, E, or X			GV6K	6.0
Range	0 = off, 1 = on, E = enable, X = don't care				
Default	E				
Response	INEN: *INENEEEE_EEEE_EEEE_EEEE_EEEE_EEEE_EEEE				
See Also	[IN], INFNC, INLVL, INPLC, INSTW, TIN, TIO				

Letter/Symbol	Description
a_ .....	Represents an address specifier, numeric value from 0 to 99. An address specifier is required if multiple Gem6K drives are connected in a daisy-chain or multi-drop configuration; in fact, leaving off the address specifier will cause parameter assignment commands to affect all units and response/transfer commands to request information from all units at the same time (multiple units transmitting characters at one time will garble the communication). To assign unique unit addresses to multiple drives, refer to the ADDR command.
B .....	Represents the number of the product's I/O bricks. External I/O bricks are represented by numbers 1 through n (to connect external I/O bricks, refer to the <i>Installation Guide</i> ). On-board I/O are address at brick location zero (Ø). If the brick identifier is omitted from the command, the controller assumes the command is supposed to affect the onboard I/O.
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. In the ANIEN command, the "d" symbol may also represent a real numeric value.
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.
% .....	(Multitasking Only) Represents a task identifier. To address the command to a specific task, prefix the command with "i%", where "i" is the task number. For example, the 4% <b>CUT</b> command uses task #4 to execute the program called "CUT".
, .....	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 <b>do not</b> require field separators (but they may be included). See <i>General Guidelines</i> table below.
@ .....	Represents a global specifier, where only one field need be entered. Applicable to all commands with multiple command fields. (e.g., @ <b>OUT1</b> sets output #1 on for all I/O bricks).
< > .....	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=b10001, and comparisons such as IF(3IN=b1001X1).

## Order of Precedence for Command Prefix Characters (from left to right):



# Syntax – General Guidelines

Guideline Topic	Guideline	Examples
Command Delimiters (<cr>, <lf>, and :)	All commands must be separated by a delimiter. A carriage return is the most commonly used. The colon (:) allows you to place multiple commands on one line of code.	Set acceleration to 10 rev/sec/sec: A10<cr> A10<lf> A10: V25 : D25000 : GO<cr>
Neutral Characters (<sp> and <tab>)	Using neutral characters anywhere within a command will not affect the command.	Set velocity to 10 rps: V<sp>10<cr>
Case Sensitivity	There is no case sensitivity. Use upper or lower case letters within commands.	Initiate motion: GO1 go1
Comment Delimiter (;)	All text between a comment delimiter and a command delimiter is considered <i>program comments</i> .	Add a comment to the command: V10<tab> ;set velocity
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 <b>do not</b> require field separators (but they may be included).	Display variable #1 on the RP240 at the current cursor location with 3 characters displayed to the left of the decimal, 2 to the right of the decimal, and the sign: DVAR1,3,2,1  Enable error checking for hard limits and drive faults: ERROR0101
Global Command Identifier (@)	When you wish to set the command value equal on all outputs, add the @ symbol at the beginning of the command (enter only the value for one command field).  The @ symbol is also useful for checking the status of all inputs or outputs on all I/O bricks.	Set the digital outputs (#1 & #2) active on all digital outputs (onboard and external): @OUT11  Check the status of all digital outputs (onboard, and on external I/O bricks): @OUT
Bit Select Operator (.)	The bit select operator allows you to affect one or more binary bits without having to enter all the preceding bits in the command. Syntax for setup commands: [command name].[bit #]-[binary value] Syntax for conditional expressions: [command name].[bit #]=[binary value]	Enable error-checking bit #9: ERROR.9-1  Enable error-check bits #9-12: ERROR.9-1,1,1,1  IF statement based on value of axis status bit #12 for axis #1: IF (IAS.12=b1)
Left-to-right Math	All mathematical operations assume left-to-right precedence.	VAR1=5+3*2 Result: Variable 1 is assigned the value of 16 (8*2), not 11 (5+6).
Binary and Hexadecimal Values	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". In the binary syntax, an "x" simply means the status of that bit is ignored.	Binary: IF (IN=b1x01) ↑ Hexadecimal: IF (IN=h7F) ↑
Multi-tasking Task Identifier (%)	Use the % command prefix to identify the command with a specific task.	Launch the "move1" program in Task 1: 1%move1  Check the error status for Task 3: 3%TER  Check the system status for Task 3: 3%TSS

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

# Syntax – Command Value Substitutions

Many commands can substitute one or more of its command field values with one of these substitution items (demonstrated in the programming example below):

VAR ..... Places current value of the numeric variable in the corresponding field of the command.  
VARB..... Uses the value of the binary variable to establish all the fields in the command.  
VARI..... Places current value of the integer variable in the corresponding field of the command.  
READ..... Information is requested at the time the command is executed.  
DREAD ..... Reads the RP240's numeric keypad into the corresponding field of the command.  
DREADF ..... Reads the RP240's function keypad into the corresponding field of the command.  
TW ..... Places the current value set on the thumbwheels in the corresponding field of the command.  
DAT ..... Places the current value of the data program (DATP) in the corresponding field of the command.

**Programming Example:** (NOTE: The substitution item must be enclosed in parentheses.)

```
VAR1=15          ; Set variable 1 to 15
A(VAR1)         ; Set acceleration to 15
VARB1=b101XX    ; Set binary variable 1 to 101XX (bits 4 & 5 not affected)
OUT(VARB1)      ; Turn on outputs 1 & 3, turn off output 2, don't change outputs 4 &
                5.
VAR$1="Enter Velocity" ; Set string variable 1 to the message "Enter Velocity"
V(READ1)        ; Read in the velocity, output variable string 1 as the prompting
                message
                ; 1. Operator sees "ENTER VELOCITY" displayed on the screen.
                ; 2. Operator enters velocity prefixed by '!' (e.g., '!20').
HOMV(TW1)       ; Read in the home velocity from thumbwheel set 1
HOMVF(DAT1)     ; Read home final velocity from data program 1.
VARI1=2*3       ; Set integer variable 1 to 6 (2 multiplied by 3)
D(VARI2)        ; Set the distance equal to the value of
                ; integer variable 2.
```

## Rule of Thumb

Not all of the commands allow command field substitutions. In general, commands with a binary command field (<b> in the syntax) will accept the VARB substitution. Commands with a real or integer command field (<r> or <i> in the syntax) will accept VAR, VARI, READ, DREAD, DREADF, TW or DAT.

# Programming Interface Tools

Motion Planner™, a graphical programming interface, is provided as a tool for programming your Gem6K drive. These are the functions provided:

- Configuration (motor selection, tuning, motor matching and damping, etc.)
- Terminal emulation for sending commands and checking drive status
- Program editor for developing program files to send to the drive
- Downloading and uploading program and operating system files to/from the drive

Motion Planner runs on the Windows 95, Windows 98 and Windows NT operating systems.

Motion Planner is installed from the “Motion Planner” CD which is included in your Gem6k drive shipment (unless you ordered the -NK option).

**Communications Server:** Also available on the Motion Planner CD is the Communications Server (COM6SRVR.EXE). COM6SRVR.EXE is a 32-bit OLE automation server that allows you to add Gem6K (as well as Gemini and 6K) communication capability to your custom applications created with programming languages such as Visual Basic, Visual C++, and Delphi. The Motion Planner installation program installs COM6SRVR.EXE in the Motion Planner directory. Details on the Communication Server functions are provided in the *COM6SRVR Communications Server User Guide*.

---

## NOTE

---

The Gem6K commands described in this document can be used only with Motion Planner or the COM6SRVR Communications Server.

---

## Using Motion Planner with a Gem6K Drive

Motion Planner is a programming interface for the Gemini product family, as well as the 6K product family. Motion Planner runs on the Windows 95, Windows 98 and Windows NT operating systems. Below are instructions on how to use Motion Planner with your Gem6K.

### Installing Motion Planner:

System Requirements:

- IBM-compatible PC with a Pentium 166 MHz or higher processor.
- Operating system: Microsoft Windows 95, Microsoft Windows NT Workstation 4.0, or Microsoft Windows 98.
- 32MB RAM.
- Hard disk space: 16MB minimum.
- PCI VGA with 800 x 600 resolution or higher.
- CD-ROM drive or internet access for installation.
- Mouse or pointing device.
- RS-232C serial port for using serial RS-232C communications.

Insert the Motion Planner CD in your CD-ROM player. The installation program automatically launches and displays this dialog:



## Updating the Drive's Operating System:

Gem6K drives are digital motor drives that run under an internal software operating system. The operating system was loaded into your drive during the manufacturing process, and under ordinary circumstances you will not need to update your drive's operating system. However, because Compumotor continues to add enhancements and address software bugs, you may want to upgrade the operating system. You may obtain a new operating system file from the Compumotor web site, or from Technical Support (see phone numbers on the inside cover of this manual).

### *Web Site Download:*

The operating system file is located in the software download section of the *Compumotor Online* web site (<http://www.compumotor.com>). The file name is in this format: GEM\_n\_nm.ops. For example, the operating system file for version 1.50 is called GEM\_1\_50.ops. Download the file to the Motion Planner directory on your hard drive.

### *Update Procedure:*

1. Using the Gem6K's RS-232 connector (COM 2), connect to your computer's RS-232 serial communication port (see instructions in the *Hardware Installation Guide*).  
**NOTE:** You can download the operating system to only one drive unit at a time and you must use RS-232 communication (no daisy chains).  
**NOTE:** You must use the Gem6K's RS-232 connector (COM 2), not the RS-232/485 connector (COM 1).
2. Launch Motion Planner.
3. In the Default Communications Settings dialog box, select your Gem6K drive and select the serial port to which the drive is connected, then click "OK".
4. Click on the Terminal tab to expose the terminal emulator.
5. From the **Communications** pull-down menu, select **Download OS**. When presented with the **Locate Gem6K Operating System** dialog, locate the operating system file and click the **Open** button. This initiates the download to the drive and displays the download status dialog.
6. When the download is completed successfully, Motion Planner displays a confirmation message. Also, the drive automatically resets itself and displays the TREV response in the terminal emulator window. Check the TREV report to verify that the proper operating system revision is now in the drive (e.g., the response "\*TREV-GV-L3E\_D1.05\_F1.00" indicates that the drive is using OS revision 1.05, denoted by "D1.05").

<p><b>NOTE:</b> If the download is interrupted or corrupted, the drive will flash the left LED (red) until a valid operating system is downloaded.</p>
--

# Programmable I/O Bit Patterns

The Gem6K has programmable inputs and outputs. The total number of onboard inputs and outputs (analog inputs, digital inputs and digital outputs) is fixed. The total number of expansion inputs and outputs depends on your configuration of expansion I/O bricks.

These programmable I/O are represented by binary bit patterns, and it is the bit pattern that you reference when programming and checking the status of specific inputs and outputs. The bit pattern is referenced 1 to *n*, from left to right.

- Onboard I/O. For example, the status command to check all onboard inputs is `TIN`.  
An example response is: `*TIN0100_0`.  

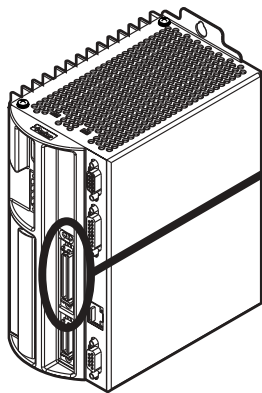
Bit 1
Bit 5
- Expansion I/O. For example, the status command to check all digital inputs on I/O brick 2 is `2TIN`.  
An example response is: `*2TIN0010_0110_1100_0000_XXXX_XXXX_XXXX_XXXX`.  

I/O Brick 2
Bit 1
Bit 32

## Onboard I/O

I/O	Location	Programming	Status Report, Assignment
Limit Inputs	"DRIVE I/O" connector	LIMFNC, LIMEN, LIMLVL	TLIM, LIM
Inputs (digital)	"DRIVE I/O" connector	INFNC, INLVL, INEN, ONIN, INPLC, INSTW	TIN, IN
Outputs (digital)	"DRIVE I/O" connector	OUT, OUTFNC, OUTLVL, OUTEN, OUTALL, OUTPLC, OUTTW, POUT	TOUT, [OUT]

### Limit Inputs ("DRIVE I/O" connector)



Input bit pattern for LIM, TLIM, LIMEN, LIMFNC, and LIMLVL:

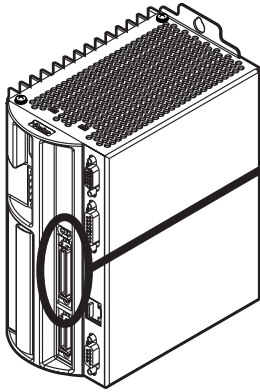
Bit #	Pin #	Function *
1	28	Positive end-of-travel limit
2	29	Negative end-of-travel limit
3	31	Home limit

\* The functions listed are the factory default functions; other functions may be assigned with the LIMFNC command.

Sample response to TLIM (limit inputs status) command:

`*TLIM110`

**Inputs** (“DRIVE I/O” connector)



Input bit pattern for TIN, IN, INFNC, INLVL, INEN, INPLC, INSTW, and ONIN:

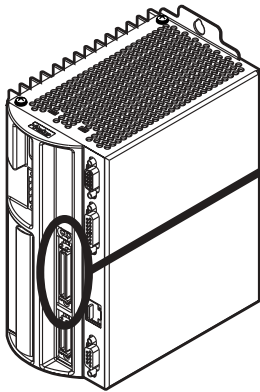
Bit #	Pin #	Function *
1	37	Input 1 (if assigned, TRIG-A)
2	38	Input 2 (if assigned, TRIG-B)
3	39	Input 3
4	34	Input 4
5	35	Input 5 (if assigned, TRIG-M)

\* If the input is assigned the “trigger interrupt” function with the INFNCi-H command:  
 TRIG-A must be on Input 1  
 TRIG-B must be on Input 2  
 TRIG-M must be on input 5

Sample response to TIN (input status) command:

\*TIN0010\_1

**Outputs** (“DRIVE I/O” connector)



Output bit pattern for TOUT, [OUT], OUT, OUTFNC, OUTLVL, OUTEN, OUTALL, OUTPLC, OUTTW, POUT:

Bit #	Pin #	Function
1	41	Output 1.
2	43	Output 2.
3	45	Output 3.
4	46	Output 4.
5	48	Output 5.
6	49	Output 6.
7	Relay	Output 7.

Sample response to TOUT (onboard outputs status) command:

\*TOUT0000\_000

## Expansion I/O Bricks

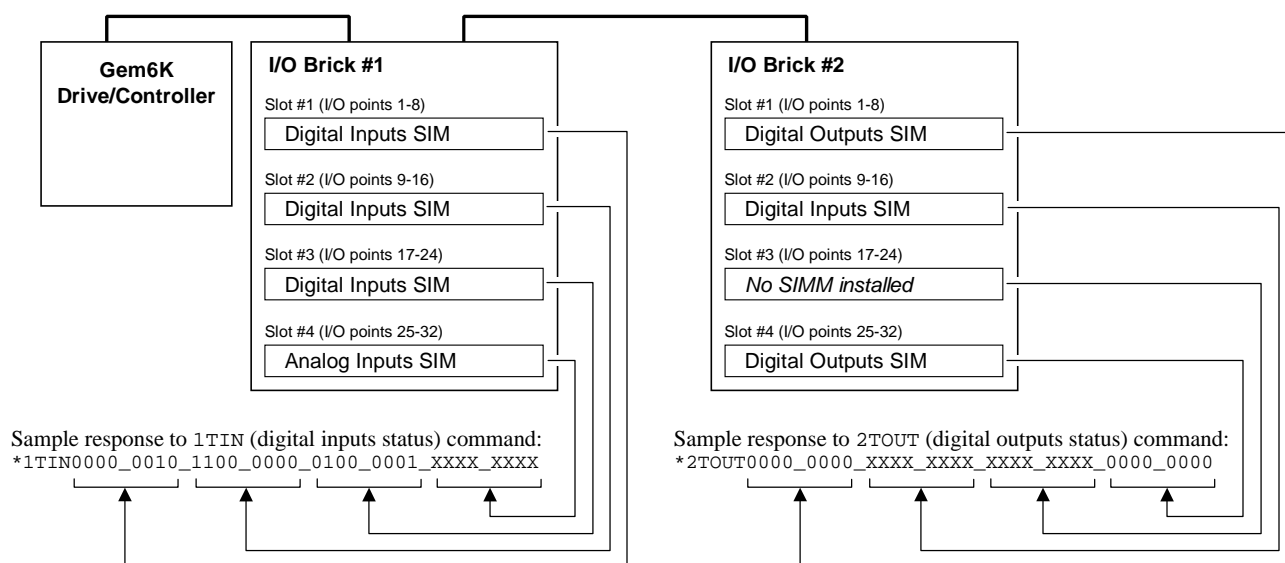
The Gem6K allows you to expand your system I/O by connecting up to eight I/O bricks (see *Installation Guide* for connections). Expansion I/O bricks may be ordered separately (referred to as the “EVM32”). Each I/O brick can hold from 1 to 4 of these I/O SIM modules in any combination:

SIM Type	Programming	Status Report, Assignment
Digital Inputs SIM (8 inputs)	INFNC, INLVL, INEN, ONIN, INPLC, INSTW	TIN, IN, TIO
Digital Outputs SIM (8 outputs)	OUT, OUTFNC, OUTLVL, OUTEN, OUTALL, OUTPLC, OUTTW, POUT	TOUT, [OUT], TIO
Analog Inputs SIM (8 inputs)	<ul style="list-style-type: none"> <li>• Enable/Disable: ANIEN.</li> <li>• Voltage range: ANIRNG.</li> <li>• Joystick setup: JOYAXH, JOYCDB, JOYCTR, JOYEDB, JOYZ.</li> <li>• Following master source: ANIMAS, FOLMAS</li> </ul>	<ul style="list-style-type: none"> <li>• Voltage: TANI, ANI, TIO</li> </ul>
Analog Outputs SIM (8 outputs)	ANO	TANO, [ANO], TIO

Each I/O brick has a unique “brick address”, denoted with the “<B>” symbol in the command syntax. The I/O bricks are connected in series to the “EXPANSION I/O” connector on the Gem6K. The 1<sup>st</sup> I/O brick has address #1, the next brick has address #2, and so on. (**NOTE:** If you leave out the brick address in the command, the Gem6K will assume you are addressing the command to the onboard I/O.) Each I/O brick has 32 I/O addresses, referenced as absolute I/O point locations:

- SIM slot 1 = I/O points 1-8
- SIM slot 2 = I/O points 9-16
- SIM slot 3 = I/O points 17-24
- SIM slot 4 = I/O points 25-32

### Example:



The TIO command identifies the connected I/O bricks (and installed SIMs), including the status of each I/O point:

```
*BRICK 1:  SIM Type      Status      Function
           1-8: DIGITAL INPUTS  0000_0000  AAAA_AAAA
           9-16: DIGITAL INPUTS  0000_0000  AAAA_AAAA
           17-24: DIGITAL INPUTS  0000_0000  AAAA_AAAA
           25-32: ANALOG INPUTS   0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000

*BRICK 2:  SIM Type      Status      Function
           1-8: DIGITAL OUTPUTS  0000_0000  AAAA_AAAA  -- SINKING
           9-16: DIGITAL INPUTS  0000_0000  AAAA_AAAA
           17-24: NO SIM PRESENT
           25-32: DIGITAL OUTPUTS  0000_0000  AAAA_AAAA  -- SOURCING
```

# Programming Error Messages

Depending on the error level setting (set with the `ERRLVL` command), when a programming error is created, the Gem6K 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 Gem6K responds with both the error message and the error prompt. At error level 3 (`ERRLVL3`), the Gem6K responds with only the error prompt.

Error Response	Possible Cause
ACCESS DENIED	Program security feature enabled, but the <i>program access input</i> ( <code>INFNCi-Q</code> or <code>LIMFNCi-Q</code> ) is 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.
ALTERNATIVE TASK NOT ALLOWED	Attempting to execute a <code>LOCK</code> command directed to another task.
AXES NOT READY	Compiled Profile path compilation error.
COMMAND NOT IMPLEMENTED	Command is not applicable to the Gem6K Series product.
COMMAND NOT ALLOWED IN PROGRAM	Command is not allowed inside a program definition (between <code>DEF</code> and <code>END</code> ).
COMMAND/DRIVE MISMATCH	The command is not appropriate to the type of drive being used (e.g., attempting to execute a servo tuning command on a stepper axis)
ERROR: MOTION ENDS IN NON-ZERO VELOCITY	Compiled Motion: The last <code>GOBUF</code> segment within a <code>PLOOP/PLN</code> loop does not end at zero velocity, or there is no final <code>GOBUF</code> segment placed outside the loop.
FOLMAS NOT SPECIFIED	No <code>FOLMAS</code> for the axis is currently specified. It will occur if <code>FMCNEW</code> , <code>FSHFC</code> , or <code>FSHFD</code> commands are executed and no <code>FOLMAS0</code> command was executed, or <code>FOLMAS0</code> was executed.
INCORRECT AXIS	Axis specified is incorrect.
INCORRECT BRICK NUMBER	Attempted to execute a command that addresses an I/O brick that is not connected to your Gem6K controller.
INCORRECT DATA	Incorrect command syntax.  Following: Velocity ( <code>v</code> ), acceleration ( <code>A</code> ) or deceleration ( <code>AD</code> ) command is zero (used by <code>FSHFC</code> & <code>FSHFD</code> ).
INPUT(S) NOT DEFINED AS JOYSTICK INPUT	Attempted to execute <code>JOYCDB</code> , <code>JOYCTR</code> , <code>JOYEDB</code> , or <code>JOYZ</code> before executing <code>JOYAXH</code> to assign the analog input.
INSUFFICIENT MEMORY	Not enough memory for the user program or compiled profile segments. This may be remedied by reallocating memory (see <code>MEMORY</code> command description).
INVALID COMMAND	Command is invalid because of existing conditions

## Programming Error Messages (continued)

Error Response	Possible Cause
INVALID CONDITIONS FOR COMMAND	<p>System not ready for command (e.g., LN command issued before the L command).</p> <p>Following (these conditions can cause an error during Following):</p> <ul style="list-style-type: none"> <li>The FOLMD value is too small to achieve the preset distance and still remain within the FOLRN/FOLRD ratio.</li> <li>A phase shift cannot be performed: <ul style="list-style-type: none"> <li>FSHFD .... Error if already shifting or performing other time based move.</li> <li>FSHFC .... Error if currently executing a FSHFD move, or if currently executing another FSHFC move in the opposite direction.</li> </ul> </li> <li>The FOLEN1 command was given while a profile was suspended by a GOWHEN.</li> </ul>
INVALID CONDITIONS FOR S_CURVE ACCELERATION—FIELD n	Average (AA) acceleration or deceleration command (e.g., AA, ADA, HOMAA, HOMADA, etc.) with a range that violates the equation $\frac{1}{2}A \leq AA \leq A$ (A is the max. accel or decel command—e.g., A, AD, HOMA, HOMAD, etc.)
INVALID DATA	<p>Data for a command is out of range.</p> <p>Following (these conditions can cause an error during Following):</p> <ul style="list-style-type: none"> <li>The parameter supplied with the command is valid. <ul style="list-style-type: none"> <li>FFILT Error if: smooth number is not 0-4</li> <li>FMCLN.. Error if: master steps &gt; 999999999 or negative</li> <li>FMCP ..... Error if: master steps &gt; 999999999 or &lt; -999999999</li> <li>FOLMD .... Error if: master steps &gt; 999999999 or negative</li> <li>FOLRD .... Error if: master steps &gt; 999999999 or negative</li> <li>FOLRN .... Error if: follower steps &gt; 999999999 or negative</li> <li>FSHFC .... Error if: number is not 0-3</li> <li>FSHFD .... Error if: follower steps &gt; 999999999 or &lt; -999999999</li> <li>GOWHEN.. Error if: position &gt; 999999999 or &lt; -999999999</li> <li>WAIT ..... Error if: position &gt; 999999999 or &lt; -999999999</li> </ul> </li> <li>Error if a GO command is given in the preset positioning mode (MCØ) and: <ul style="list-style-type: none"> <li>FOLRN = zero</li> <li>FOLMD = zero, or too small</li> </ul>                     (see Following chapter in the <i>Programmer's Guide</i>)                 </li> </ul>
INVALID FOLMAS SPECIFIED	Following: An illegal master was specified in FOLMAS. A follower may never use its own commanded position or feedback source as its master.
INVALID RATIO	Following: Error if the FOLRN:FOLRD ratio after scaling is > 127 when a GO is executed.
INVALID TASK IDENTIFIER	Attempting to launch a PEXE or EXE command into the supervisor task (task 0).
LABEL ALREADY DEFINED	Defining a program or label with an existing program name or label name.
MASTER SLAVE DISTANCE MISMATCH	Attempting a preset Following move with a FOLMD value that is too small.
MAXIMUM COMMAND LENGTH EXCEEDED	Command exceeds the maximum number of characters.
MAXIMUM COUNTS PER SECOND EXCEEDED	Velocity value is greater than 1,600,000 counts/sec.
MOTION IN PROGRESS	<p>Attempting to execute a command not allowed during motion (see Restricted Commands During Motion section in the <i>Programmer's Guide</i>.)</p> <p>Following: The FOLEN1 command was given while that follower was moving in a non-Following mode.</p>
NEST LEVEL TOO DEEP	IFs, REPEATs, WHILEs, or GOSUBs nested greater than 16 levels (for each type).

## Programming Error Messages (continued)

Error Response	Possible Cause
NO MOTION IN PROGRESS	Attempting to execute a command that requires motion, but motion is not in progress.
NO PATH SEGMENTS DEFINED	Compiled Profile compilation error.
NO PROGRAM BEING DEFINED	END command issued before a DEF command.
NOT ALLOWED IN PATH	Compiled Profile path compilation error.
NOT VALID DURING FOLLOWING MOTION	A GO command was given while moving in the Following mode (FOLEN1) and while in the preset positioning mode (MCØ).
NOT VALID DURING RAMP	A GO command was given while moving in a Following ramp and while in the continuous positioning mode (MC1). Following status (FS) bit #3 will be set to 1. A FOLEN command was given during one of these conditions: <ul style="list-style-type: none"><li>• During a shift (FSHFC or FSHFD)</li><li>• During a change in ratio (FOLRN/FOLRD)</li><li>• During deceleration to a stop</li></ul>
OUTPUT USED AS OUTFNC	Attempting to change an output that is not an OUTFNCi-A output.
PROFILE ALREADY MOVING	Compiled Profile compilation error.
PROFILE NOT COMPILED	Attempting to execute a profile that has not been compiled.
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 (DATTC) 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	<b>ENABLE</b> input is no longer connected to ground ( <b>GND</b> ).
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.

**Using Motion Planner:** If you are typing the command in a live terminal emulator session, the controller will detect the bad command and respond with an error message, followed by the ERRBAD error prompt (?). If the bad command was detected on download, the bad command is reported automatically (see example below).

**NOTE:** If you are not using Motion Planner, you'll have to type in the TCMDER command at the error prompt to display the bad command.

Once a command error has occurred, the command and its fields are stored and system status bit #11 (reported in the TSSF, TSS and SS commands) is set to 1. The status bit remains set until the TCMDER command is issued.

### Example Error Scenario:

1. In Motion Planner's program editor, create and save a program with a programming error:

```
DEL badprg      ; Delete a program before defining and downloading
DEF badprg      ; Begin definition of program called badprg
MA1             ; Select the absolute preset positioning mode
A25            ; Set acceleration
AD11           ; Set deceleration
V5             ; Set velocity
VAR1=0         ; Set variable #1 equal to zero
GO1            ; Initiate move
IF(VAR1<)16    ; MISTYPED IF STATEMENT - should be typed as "IF(VAR1<16)"
VAR1=VAR1+1    ; If variable #1 is less than 16, increment the counter by 1
NIF            ; End IF statement
END            ; End programming of program called badprg
```

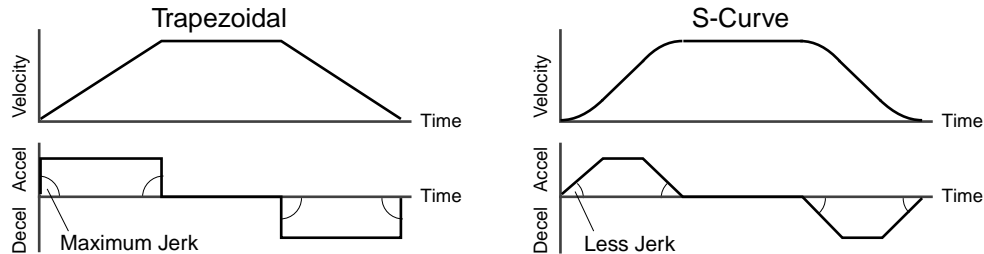
2. Using Motion Planner's terminal emulator, download the program to the Gem6K Series product. Notice that an error response identifies the bad command as an "INCORRECT DATA" item and displays it:

```
> *NO ERRORS
*INCORRECT DATA
> *IF(VAR1<)16
```



# S-Curve Acceleration/Deceleration Profiling

The Gem6K allows you to perform *S-curve* move profiles, in addition to the usual trapezoidal profiles. S-curve profiling provides smoother motion control by reducing the *jerk* (rate of change) in acceleration and deceleration portions of the move profile (see drawing below). Because S-curve profiling reduces jerk, it improves position tracking performance, especially in linear interpolation applications.



## S-Curve Programming Requirements

To program an S-curve profile, you must use the *average accel/decel* commands provided in the Gem6K programming language. For every maximum accel/decel command (e.g., A, AD, HOMA, HOMAD, JOGA, JOGAD, etc.) there is an *average* command for S-curve profiling (see table below).

Maximum Accel/Decel Commands:		Average (“S-Curve”) Accel/Decel Commands:	
Command	Function	Command	Function
A	Acceleration	AA	Average Acceleration
AD	Deceleration	ADA	Average Deceleration
HOMA	Home Acceleration	HOMAA	Average Home Acceleration
HOMAD	Home Deceleration	HOMADA	Average Home Deceleration
JOGA	Jog Acceleration	JOGAA	Average Jog Acceleration
JOGAD	Jog Deceleration	JOGADA	Average Jog Deceleration
JOYA	Joystick Acceleration	JOYAA	Average Joystick Acceleration
JOYAD	Joystick Deceleration	JOYADA	Average Joystick Deceleration
LHAD	Hard Limit Deceleration	LHADA	Average Hard Limit Deceleration
LSAD	Soft Limit Deceleration	LSADA	Average Soft Limit Deceleration

## Determining the S-Curve Characteristics

The command values for average accel/decel (AA, ADA, etc.) and maximum accel/decel (A, AD, etc.) determine the characteristics of the S-curve. To smooth the accel/decel ramps, you must enter average accel/decel command values that satisfy the equation  $\frac{1}{2}A \leq AA < A$ , where A represents maximum accel/decel and AA represents average accel/decel. Given this requirement, the following conditions are possible:

Acceleration Setting	Profiling Condition
AA > ½ A, but AA < A	S-curve profile with a variable period of constant acceleration. Increasing the AA value above the pure S-curve level (AA > ½ A), the time required to reach the target velocity and the target distance is decreased. However, increasing AA also increases jerk.
AA = ½ 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 < ½ 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. (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.) However, if you enter an average decel command (e.g., ADA, HOMADA, etc.) equal to zero, you will receive the "INVALID DATA-FIELD n" error.
AA ≠ zero and AA ≠ A	S-curve profiling is enabled <b>only for standard moves</b> . All subsequent standard moves for that axis must comply with this equation: ½ A ≤ AA < A.
AA > ½ A	Average accel/decel is raised above the pure S-curve level; this decreases the time required to reach the target velocity and distance. However, increasing AA also increases jerk. After increasing AA, you can reduce jerk by increasing A, but be aware that increasing A requires a greater torque to achieve the commanded velocity at the mid-point of the acceleration profile.
No AA value ever entered	Profile will default to trapezoidal. AA tracks A.

If you never change the A or AA deceleration commands, AA deceleration will track AA acceleration. However, once you change A deceleration, AA deceleration will no longer track changes in AA acceleration. For example, if you never change the AD or ADA command values, ADA will track the AA command value. But once you change AD, the ADA command value will no longer track the changes in the AA command value.

**Calculating Move Times.** The calculation for determining S-curve average accel and decel move times is as follows (*calculation method identical for S-curve and trapezoidal moves*):

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

**Scaling** affects the AA average acceleration (AA, ADA, etc.) the same as it does for the A maximum acceleration (A, AD, etc.). See page 19 for details on scaling.

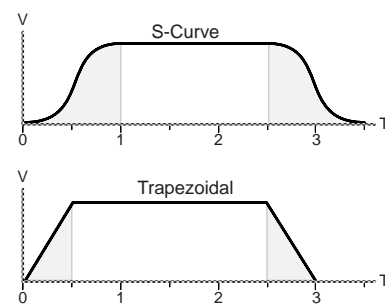
**NOTE:** Equations for calculating jerk are provided on page 17.

### Programming Example (see move profile drawings below)

```
; In this example, prog1 executes a pure S-curve and takes 1 second
; to reach a velocity of 5 rps; prog2 executes a trapezoidal profile
; and takes 0.5 seconds to reach a velocity of 5 rps.
```

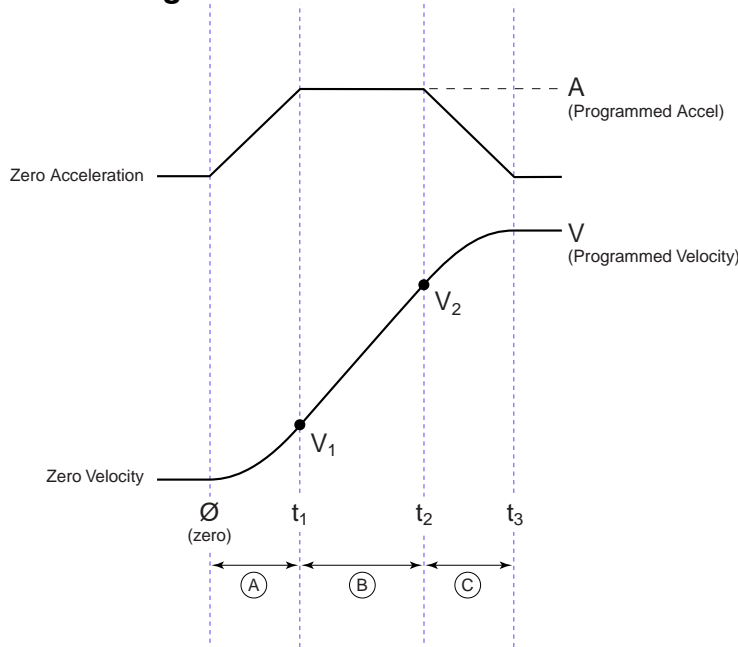
```
SCALE0      ; Disable scaling
DEL Prog1   ; Delete program called prog1
DEF Prog1   ; Begin definition of prog1
MA0         ; Select incremental positioning mode
D40000     ; Set distance to 40,000
A10        ; Set max. accel to 10 revs/sec/sec
AA5        ; Set avg. accel to 5 revs/sec/sec
AD10       ; Set max. decel to 10 revs/sec/sec
ADA5       ; Set avg. decel to 5 revs/sec/sec
V5         ; Set velocity to 5 revs/sec
GO1        ; Execute motion
END        ; End definition of prog1
```

```
DEL Prog2   ; Delete program called prog2
DEF Prog2   ; Begin definition of prog2
MA0         ; Select incremental positioning mode
D40000     ; Set distance to 40,000
A10        ; Set max. accel to 10 revs/sec/sec
AA10       ; Set avg. accel to 10 revs/sec/sec
AD10       ; Set max. decel to 10 revs/sec/sec
ADA10      ; Set avg. decel to 10 rev/sec/sec
V5         ; Set velocity to 5 revs/sec
GO1        ; Execute motion
END        ; End definition of program #2
```



Move profiles

## Calculating Jerk



### Rules of Motion:

$$\text{Jerk} = \frac{da}{dt}$$

$$a = \frac{dv}{dt}$$

$$v = \frac{dx}{dt} \quad (x = \text{distance})$$

Assuming the accel profile starts when the load is at zero velocity and the ramp to the programmed velocity is not compromised:

$$\text{Jerk} = J_A = \frac{A^2 * AA}{V (A-AA)}$$

A = programmed acceleration (A, AD, HOMAD, etc.)

AA = average acceleration (AA, ADA, HOMAA, etc.)

V = programmed velocity (V, HOMV, etc.)

$$t_1 = \frac{A}{J_A}$$

$$t_2 = \frac{V}{AA} - \frac{A}{J_A}$$

$$t_3 = \frac{V}{AA}$$

NOTE:  $t_3 - t_2 = t_1$

$$V_1 = \frac{J_A * t_1^2}{2} = \frac{A^2}{2 * J_A}$$

$$V_2 = V - \frac{A^2}{2 * J_A}$$

Ⓐ  $t_1 \geq t \geq 0$

$$a(t) = J_A * t$$

$$v(t) = \frac{J_A * t^2}{2}$$

$$d(t) = \frac{J_A * t^3}{6}$$

a(t) = acceleration at time t  
v(t) = velocity at time t  
d(t) = distance at time t

Ⓑ  $t_2 \geq t > t_1$

$$a(t) = A$$

$$v(t) = \frac{A^2}{2J_A} + A * (t - t_1)$$

$$d(t) = \frac{J_A * t_1^3}{6} + \left( \frac{A * (t - t_1)^2}{2} \right) + \left( V_1 * (t - t_1) \right)$$

Ⓒ  $t_3 \geq t > t_2$

$$a(t) = A - (J_A * (t - t_2))$$

$$v(t) = V - \left( \frac{J_A * (t_3 - t)^2}{2} \right)$$

$$d(t) = \frac{V^2}{2AA} + \left( \frac{J_A (t_3 - t)^3}{6} \right) - \left( V * (t_3 - t) \right)$$

**Starting at a Non-Zero Velocity:** If starting the acceleration profile with a non-zero initial velocity, the move comprises two components: a constant velocity component, and an s-curve component. Typically, the change of velocity should be used in the S-curve calculations. Thus, in the calculations above, you would substitute “(V<sub>F</sub> - V<sub>O</sub>)” for “V” (V<sub>F</sub> = final velocity, V<sub>O</sub> = initial velocity). This is shown in the jerk equation (right).

$$\text{Jerk} = J_A = \frac{A^2 * AA}{(V_F - V_O) (A-AA)}$$

# Units of Measure and Scaling

## Units of Measure without Scaling

Scaling is disabled (SCALEØ) as the factory default condition:

- Stepper axes: All distance values entered are in commanded counts (sometimes referred to as *motor steps*), and all acceleration, deceleration and velocity values entered are internally multiplied by the DRES command value.

Motion Attribute	Units of Measure Encoder or Resolver
Accel/Decel	Revs/sec/sec *
Velocity	Revs/sec *
Distance	Counts **

\* All accel/decel & velocity values are internally multiplied by the ERES command value.

\*\* Distance is measured in the counts received from the feedback device.

## What is Scaling?

Scaling allows you to program acceleration, deceleration, velocity, and position values in units of measure that are appropriate for your application. The SCALE command is used to enable or disable scaling (SCALE1 to enable, SCALEØ to disable). The motion type(s) you are using in your application determines which scale factor commands you need to configure:

Type of Motion	Accel/Decel Scaling	Velocity Scaling	Distance Scaling
Standard Point-to-Point Motion	SCLA	SCLV	SCLD
Following	SCLA	SCLV	SCLD for follower distances SCLMAS for master distances

## When Should I Define Scaling Factors?

Scaling calculations are performed when a program is defined or downloaded. Consequently, you must enable scaling (SCALE1) and define the scaling factors (SCLD, SCLA, SCLV, SCLMAS) *prior* to defining (DEF), uploading (TPROG), or running (RUN) the program. **NOTE:** All scaling settings (SCALE, SCLA, SCLD, SCLV and SCLMAS) are automatically saved in the Gem6K's battery-backed RAM.

**RECOMMENDED:** Place the scaling commands at the beginning of your program file, *before* the location of any defined programs. This ensures that the motion parameters in subsequent programs in your program file are scaled correctly. When you use Motion Planner's scaling setup wizard, the scaling commands are automatically placed in the appropriate location in your program file.

**ALTERNATIVE:** Scaling factors could be defined via a terminal emulator *just before* defining or downloading a program. Because scaling command values are saved in battery-backed RAM (remembered after you issue a RESET command or cycle power to the Gem6K), all subsequent program definitions and downloads will be scaled correctly.

### NOTES

- Scaling commands are not allowed in a program. If there are scaling commands in a program, the Gem6K will report an error message ("COMMAND NOT ALLOWED IN PROGRAM") when the program is downloaded.
- If you intend to upload a program with scaled motion parameters, be sure to use Motion Planner. Motion Planner automatically uploads the scaling parameters and places them at the beginning of the program file containing the uploaded program from the Gem6K. This ensures correct scaling when the program file is later downloaded.

## Acceleration & Deceleration Scaling (SCLA)

**Stepper Axes:** If scaling is enabled (SCALE1), all accel/decel values entered are internally multiplied by the acceleration scaling factor to convert user units/sec/sec to commanded counts/sec/sec. The scaled values are always in reference in commanded counts, regardless of the existence of an encoder.

**Servo Axes:** If scaling is enabled (SCALE1), all accel/decel values entered are internally multiplied by the acceleration scaling factor to convert user units/sec/sec to encoder or resolver counts/sec/sec.

All accel/decel commands (e.g., A, AA, AD, HOMA, HOMAD, JOGA, etc.) are multiplied by the SCLA command value.

As the accel/decel scaling factor (SCLA) changes, the resolution of the accel and decel values and the number of positions to the right of the decimal point also change (see table at right). An accel/decel value with greater resolution than allowed will be truncated (e.g., if scaling is set to SCLA10, the A9.9999 command would be truncated to A9.9).

SCLA value (counts/unit/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.

Axis Type	Min. Accel or Decel (resolution)	Max. Accel or Decel
Stepper	$\frac{0.001 * DRES}{SCLA}$	$\frac{999.9999 * DRES}{SCLA}$
Servo	Encoder Feedback: $\frac{0.001 * ERES}{SCLA}$	Encoder Feedback: $\frac{999.9999 * ERES}{SCLA}$

## Velocity Scaling (SCLV)

**Stepper Axes:** If scaling is enabled (SCALE1), all velocity values entered are internally multiplied by the velocity scaling factor to convert user units/sec to commanded counts/sec. The scaled values are always in reference to commanded counts (sometimes referred to as “motor steps”).

**Servo Axes:** If scaling is enabled (SCALE1), all velocity values entered are internally multiplied by the velocity scaling factor to convert user units/sec to encoder or resolver counts/sec.

All velocity commands (e.g., V, HOMV, HOMVF, JOGVH, JOGVL, etc.) are multiplied by the SCLV command value.

As the velocity scaling factor (SCLV) changes, the velocity command's range and its decimal places 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 V9.9999 command would be truncated to V9.9.

SCLV Value (counts/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 table to determine the maximum velocity for your product type.

Max. Velocity for Stepper Axes	Max. Velocity for Servo Axes
50 revs/sec	200 revs/sec

**NOTE:** These velocity limitations are hardware based. Make sure the velocity scaling value used with the desired velocity (V command) does not exceed these velocities. The Gem6K will generate an error message if you try to exceed these maximums. The maximum servo velocity may also be limited when using encoder feedback, depending on the encoder resolution used. Refer to the *Gem6K Hardware Installation Guide* for the maximum encoder input frequency.

## Distance Scaling (SCLD and SCLMAS)

**Stepper Axes:** If scaling is enabled (SCALE1), all distance values entered are internally multiplied by the distance scaling factor to convert user units to commanded counts (“motor steps”).

**Servo Axes:** If scaling is enabled (SCALE1), all distance values entered are internally multiplied by the distance scaling factor to convert user units to encoder or analog input counts.

All standard motion distance commands (e.g., D, PSET, REG, SMPER, STRGTD) and are multiplied by the SCLD command value. The only exception is for master distance values (see table below):

**Scaling for Following Motion:** The SCLD command defines the follower axis distance scale factor, and the SCLMAS command defines the master’s distance scale factor. The Following-related commands that are affected by SCLD and SCLMAS are listed in the table below.

### Commands Affected by Master Scaling (SCLMAS)

FMCLEN: *Master Cycle Length*  
 FMCP: *Master Cycle Position Offset*  
 FOLMD: *Master Distance*  
 FOLRD: *Follower-to-Master Ratio (Denominator)*  
 GOWHEN: *Conditional GO (left-hand variable is PMAS)*  
 TPMAS & [PMAS]: *Position of Master Axis*  
 TVMAS & [VMAS]: *Velocity of Master Axis*

### Commands Affected by Follower Scaling (SCLD)

FOLRN: *Follower-to-Master Ratio (Numerator)*  
 FGADV: *Geared Advance*  
 FSHFD: *Preset Phase Shift*  
 GOWHEN: *Conditional GO (left-hand variable ≠ PMAS)*  
 TPSHF & [PSHF]: *Net Position Shift of Follower*  
 TPSLV & [PSLV]: *Position of Follower Axis*

As the SCLD or SCLMAS scaling factor changes, the distance command’s range and its decimal places also change (see table below). A distance value with greater resolution than allowed will be truncated. For example, if scaling is set to SCLD4000, the D105.2776 command would be truncated to D105.277.

SCLD or SCLMAS Value (counts/unit)	Distance Resolution (units)	Distance Range * (units)	Decimal Places
1 - 9	1.0	0 - ±999999999	0
10 - 99	0.10	0.0 - ±99999999.9	1
100 - 999	0.010	0.00 - ±9999999.99	2
1000 - 9999	0.0010	0.000 - ±999999.999	3
10000 - 99999	0.00010	0.0000 - ±99999.9999	4
100000 - 999999	0.00001	0.00000 - ±9999.99999	5

### NOTE

### FRACTIONAL STEP TRUNCATION

### NOTE

If you are operating in the incremental mode (MAØ), or specifying master distance values with FOLMD, when the distance scaling factor (SCLD or SCLMAS) and the distance value are multiplied, a fraction of one step may 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 SCLD or SCLMAS to 1, or a multiple of 10.

## Scaling Example — Steppers (GT6K)

We configure for a 25,000 step/rev system attached to 5-pitch leadscrew. The user wants to program motion parameters in inches; therefore the scale factor calculation is: 25,000 steps/rev x 5 revs/inch = 125,000 steps/inch. For instance, with a scale factor of 125,000, the operator could enter a move distance value of 2.000 and the controller would send out 250,000 pulses, corresponding to two inches of travel.

```
SCALE1           ; Enable scaling
DRES25000        ; Set drive resolution to 25,000 steps/rev
SCLD125000       ; Allow user to enter distance in inches
SCLV125000       ; Allow user to enter velocity in inches/sec
SCLA125000       ; Allow entering accel/decel in inches/sec/sec
DRESET           ; Reset the drive so the DRES value is accepted
```

## Scaling Example — Servos (GV6K)

We configure for a 4,000 count/rev servo motor/drive system (using a 1000-line encoder) attached to a 5-pitch leadscrew. The user wants to position in inches; therefore, the scale factor calculation is 4,000 counts/rev x 5 revs/inch = 20,000 counts/inch.

```
SFBI             ; Select encoder feedback
ERES4000         ; Set encoder resolution to 4000 steps/rev (post quadrature)
SCALE1           ; Enable scaling
SCLD20000        ; Allow user to enter distance values in inches
SCLV20000        ; Allow user to enter velocity values in inches/sec
SCLA20000        ; Allow user to enter accel/decel values in inches/sec/sec
DRESET           ; Reset the drive so the DRES value is accepted
```

## Scaling Example — Following

Typically, the master and follower scale factors are programmed so that master and follower units are the same, but this is not required. Consider the scenario below as an example.

The master is a 1000-line encoder (4000 counts/rev post-quadrature) mounted to a 50 teeth/rev pulley attached to a 10 teeth/inch conveyor belt, resulting in 80 counts/tooth (4000 counts/50 teeth = 80 counts/tooth). To program in inches, you would set up the master scaling factor with the SCLMAS800 command (80 counts/tooth \* 10 teeth/inch = 800 counts/inch).

The follower is a servo motor with position feedback from a 1000-line encoder (4000 counts/rev). The motor is mounted to a 4-pitch (4 revs/inch) leadscrew. Thus, to program in inches, you would set up the follower scaling factor with the SCLD16000 command (4000 counts/rev \* 4 revs/inch = 16000 counts/inch).

```
SCALE1           ; Enable scaling
SCLMAS800        ; Master scaling (80 counts/tooth * 10 teeth/inch = 800 counts/inch)
SCLD16000        ; Follower scaling (4000 counts/rev * 4 revs/inch = 16000 counts/inch)
```

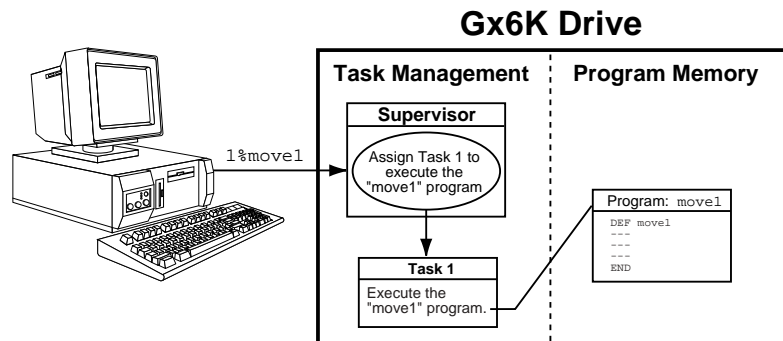




# Special Programming Characters

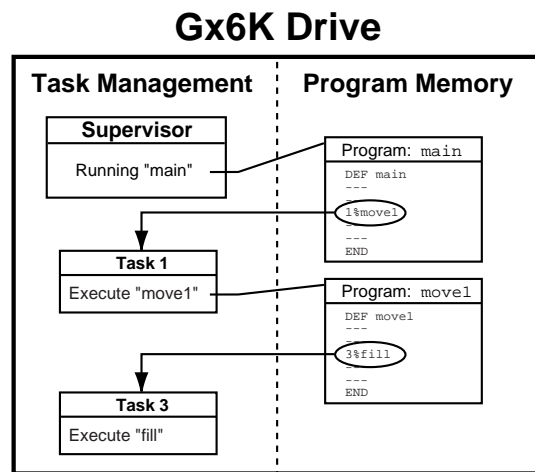
%	Task Identifier	Product	Rev
Type	Multi-Tasking	GT6K	6.0
Syntax	<a_>i%<command>	GV6K	6.0
Units	i = task number		
Range	1-10		
Default	1		
Response	n/a		
See Also	LOCK, [SWAP], [TASK], TSKAX, TSKTRN, TSWAP, TTASK		

Use the Task Identifier (%) prefix to specify that the associated command will affect the indicated task number. For most simple multi-tasking applications, the % prefix is used to start a program running in a specific task. For example, the drawing on the right illustrates how the 1%move1 command starts the program called "move1" in task 1 (specified with the 1% prefix).



Because the % prefix specifies the task number that the associated command will affect, new tasks can be started from within other tasks, as shown in the drawing on the right. →

Within a program in a task, it is not necessary to use the % prefix unless trying to initiate a program or command in a different task. For example, if the fill program running in task 3 executes a COMEXC1 command, only task 3 is placed into COMEXC1 mode. If the fill program running in task 3 also executes a 2%PS command, task 3 executes the command, but the program being executed in task 2 is paused, not task 3.



**How the Task Supervisor Works:** The “Task Supervisor” (also referred to as Task  $\emptyset$ ) is the main program execution environment. It contains the command buffer and parser. Immediate commands and commands executed from the communications buffer are implicitly directed to affect the supervisor unless explicitly directed to a task with the % prefix. Only the supervisor executes buffered commands from the communications buffer. If the supervisor is executing a program, incoming commands will be buffered, not executed. If the supervisor is not executing a program, it will execute commands from the input command buffer, even if the other tasks are executing programs. If a command in the command buffer has a task prefix, it is still executed by the supervisor, but affects the task specified by the prefix.

[ ! ] Immediate Command Identifier		Product	Rev
Type	Operator (Other)		
Syntax	<a_>!<command>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	COMEXC		

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 Gem6K commands are buffered.

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

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

---

**[ @ ]****Global Command Identifier**

		<b>Product</b>	<b>Rev</b>
Type	Operator (Other)		
Syntax	@<command><field1>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	ANIEN, ANIRNG, INEN, INLVL, OUT, OUTALL, OUTEN, OUTLVL, TANI, TANO, TIN, TIO, TOUT		

---

The Global Command Identifier (@) is used to set the value of all fields to the value entered only in the first field. For example, @OUT1 sets output #1 on for all I/O bricks. If you have any doubts about which commands can use the @ symbol, refer to the **Syntax** portion of the command description.

---

**;****Begin Comment**

		<b>Product</b>	<b>Rev</b>
Type	Operator (Other)		
Syntax	;<this is a comment>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	None		

---

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**

		<b>Product</b>	<b>Rev</b>
Type	Operator (Other)		
Syntax	<a_><!>\$<t>	GT6K	6.0
Units	t = text name	GV6K	6.0
Range	Text name of 6 characters or less		
Default	n/a		
Response	n/a		
See Also	DEF, DEL, END, GOSUB, GOTO, JUMP, RUN, TLABEL		

---

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.

**NOTE:** The maximum number of labels possible is 600.

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

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

**NOTE:** A label cannot have the same name as a Gem6K command. For example, \$A and \$A123 are illegal labels.

**Example**

```

DEF pick          ; Begin definition of program called pick
GO1              ; Initiate motion
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
TAS              ; Report axis status
BREAK           ; Break out of current subroutine or program
$pick2           ; Label declaration for pick2
TPC              ; Report commanded position
END              ; End program definition
RUN pick         ; Execute program named pick

```

---

**[ # ] Step Through a Program**

Type	Operator (Other)	Product	Rev
Syntax	<a_>!#<i>	GT6K	6.0
Units	i = number of commands to execute from the buffer	GV6K	6.0
Range	i = 1 - 200		
Default	1		
Response	n/a		
See Also	DEF, HELP, STEP, TRACE, TRANS		

---

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:**

```

DEF tst          ; Begin definition of program named tst
V1              ; Set velocity to 1 unit/sec
A10             ; Set acceleration to 10 units/sec/sec
D1              ; Set distance to 1 unit
GO1             ; Initiate motion
OUT11X1         ; Turn on on-board 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 A10.
!#              ; Execute 1 command from program; command to execute is D1
!#1            ; Execute 1 command from program; command to be executed is GO1
!#2            ; Execute 2 commands from program; commands to be executed are
                ; OUT11X1 and END

```

Enter Interactive Data			
Type	Operator (Other)	Product	Rev
Syntax	!'<numeric data>	GT6K	6.0
Units	Numeric data is command-dependent	GV6K	6.0
Range	Numeric data is command-dependent		
Default	n/a		
Response	n/a		
See Also	[ READ ], VARI, VARS		

To enter data interactively, two operations must occur. First, numeric information must be requested. Requesting the numeric information is accomplished with the VAR<sub>x</sub>=READ<sub>y</sub> 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:**

```

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)	Product	Rev
Syntax	<command>.i	GT6K	6.0
Units	i = bit number	GV6K	6.0
Range	Command-dependent		
Default	None		
Response	n/a		
See Also	[ AS ], [ ER ], ERROR, [ IN ], INEN, INLVL, [ INO ], INTHW, LIMLVL, [ MOV ], ONIN, ONUS, OUT, OUTEN, OUTLVL, POUT, [ SS ], TAS, TER, TIN, TINO, TIO , TOUT, TSS, TUS, [ US ]		

The Bit Select (.) operator specifies which bit to select. The primary purpose of this command is to let the user specify a specific bit (or range), instead of having to type in an entire 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(AS.12=b1) is legal, but IF(b1=AS.12) is illegal.

**Command Shortcut Examples** (affect only one binary bit location):

- Activate outputs at I/O location Brick 3, I/O point 9: 3OUT.9=1
- Enable analog input at I/O location Brick 2, I/O point 2: 2ANIEN.2=E
- Enable error-checking bit 6 for task 3: 3%ERROR.6=1

**Example:**

```

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
2OUT.5=1 ; Activate the output at location Brick 2, I/O point 5

```

---

**[ " ]****Begin and End String**

Type	Operator (Other)	<b>Product</b>	<b>Rev</b>
Syntax	"<message>" (see below for possibilities)	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	DWRITE, VARS, WRITE, WRVARS		

---

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 B](#). 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**

Type	Operator (Other)	<b>Product</b>	<b>Rev</b>
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
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 B](#). 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:**

WRITE"cd\92GEM6K\13\10" ;Displays: cd\GEM6K<cr><lf>

**[ = ]**

**Assignment or Equivalence**

Type	Operator (Mathematical or Relational)	<b>Product</b>	<b>Rev</b>
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	[ > ], [ >= ], [ < ], [ <= ], [ <> ], [ AND ], IF, [ OR ], UNTIL, VAR, VARB, VARI, 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, VARI, 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)	<b>Product</b>	<b>Rev</b>
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	[ = ], [ >= ], [ < ], [ <= ], [ <> ], [ AND ], IF, [ OR ], UNTIL, WAIT, WHILE		

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).

<b>[ &gt;= ]</b>		<b>Greater Than or Equal</b>	
Type	Operator (Relational)	<b>Product</b>	<b>Rev</b>
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	[ = ], [ > ], [ < ], [ <= ], [ <> ], [ AND ], IF, [ OR ], UNTIL, WAIT, WHILE		

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).

<b>[ &lt; ]</b>		<b>Less Than</b>	
Type	Operator (Relational)	<b>Product</b>	<b>Rev</b>
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	[ = ], [ > ], [ >= ], [ <= ], [ <> ], [ AND ], IF, [ OR ], UNTIL, WAIT, WHILE		

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).

<b>[ &lt;= ]</b>		<b>Less Than or Equal</b>	
Type	Operator (Relational)	<b>Product</b>	<b>Rev</b>
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	[ = ], [ > ], [ < ], [ >= ], [ <> ], [ AND ], IF, [ OR ], UNTIL, WAIT, WHILE		

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).

---

<b>[ &lt;&gt; ]</b>	<b>Not Equal</b>	<b>Product</b>	<b>Rev</b>
Type	Operator (Relational)	GT6K	6.0
Syntax	See below	GV6K	6.0
Units	n/a		
Range	n/a		
Default	n/a		
Response	n/a		
See Also	[ = ], [ >= ], [ < ], [ <= ], [ AND ], <i>IF</i> , [ OR ], <i>UNTIL</i> , <i>WAIT</i> , <i>WHILE</i>		

---

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).

---

<b>[ ( ) ]</b>	<b>Operation Priority Level</b>	<b>Product</b>	<b>Rev</b>
Type	Operator (Mathematical)	GT6K	6.0
Syntax	See below	GV6K	6.0
Units	n/a		
Range	n/a		
Default	n/a		
Response	n/a		
See Also	[ = ], [ - ], [ * ], [ / ], [ SQRT ], <i>VAR</i> , <i>VARI</i>		

---

The Operation Priority Level operator 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.

Example of valid command: *VAR1*=(*VAR2* \* 3) \* (3 + *VAR4*)

<b>[ + ] Addition</b>		<b>Product</b>	<b>Rev</b>
Type	Operator (Mathematical)		
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	[ = ], [ ( ) ], [ - ], [ * ], [ / ], [ SQRT ], VAR, VARI, VARB		

The addition (+) operator adds the value on the left of the operator to the value on the right of the operator. The addition operator can only be used in conjunction with the VAR, VARI 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=b1101 + b11001

<b>[ - ] Subtraction</b>		<b>Product</b>	<b>Rev</b>
Type	Operator (Mathematical)		
Syntax	See Below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	[ = ], [ ( ) ], [ + ], [ * ], [ / ], [ SQRT ], VAR, VARI, VARB		

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, VARI 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 command s: VAR1=1-2-3-4-5-6-7-8-9  
VAR2=VAR1-1+(5\*3)  
VARB1=b111101 - b11001

<b>[ * ] Multiplication</b>		<b>Product</b>	<b>Rev</b>
Type	Operator (Mathematical)		
Syntax	See Below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	[ = ], [ ( ) ], [ + ], [ - ], [ / ], [ SQRT ], VAR, VARI, VARB		

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, VARI and VARB commands. (Values assigned to VARI variables are truncated to integer values.)

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=b111101 \* b11001

---

**[ / ]****Division**

Type	Operator (Mathematical)	<b>Product</b>	<b>Rev</b>
Syntax	See Below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	[ = ], [ ( ) ], [ + ], [ - ], [ * ], [ SQRT ], VAR, VARI, VARB		

---

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 (VARI integer variables are truncated). The division operator can only be used in conjunction with the VAR, VARI 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=b111101 / b11001

**DIVISION BY ZERO IS NOT ALLOWED.**

---

**[ & ]****Boolean And**

Type	Operator (Bitwise)	<b>Product</b>	<b>Rev</b>
Syntax	See Below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	[ = ], [   ], [ ~ ], [ ^ ], [ << ], [ >> ], VAR, VARI, VARB		

---

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 or VARI 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 or VARI), the possible combinations are as follows:

positive number & positive number	= 1
positive number & zero or a negative number	= 0
zero or negative number & positive number	= 0
zero or negative number & zero or negative number	= 0

Example: VAR1=5 & -1  
Result: VAR1=0

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	X & X = X
1 & 0 = 0	1 & X = X
0 & 1 = 0	X & 1 = X
0 & 0 = 0	0 & X = 0
	X & 0 = 0

Example: VARB1=b0000 1000 & b1000 1011 1

Response to VARB1 is \*VARB1=0000\_1000\_XXXX\_XXXX\_XXXX\_XXXX\_XXXX\_XXXX

Example: VARB1=h32FD & h23

Response to VARB1 is \*VARB1=0100\_0100\_0000\_0000\_0000\_0000\_0000\_0000

Example: VARB1=h23 & b1101

Response to VARB1 is \*VARB1=0100\_XX00\_0000\_0000\_0000\_0000\_0000\_0000

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**

Type	Operator (Bitwise)	<b>Product</b>	<b>Rev</b>
Syntax	See Below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	[ = ], [ & ], [ ~ ], [ ^ ], [ << ], [ >> ], VAR, VARI, 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 or VARI 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 or VARI), 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	=	0
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	X		X = 1
1		0 = 1	1		X = 1
0		1 = 1	X		1 = 1
0		0 = 0	0		X = X
			X		0 = X

Example: VARB1=b1001 01X1 XX11 | b1000 1011 10  
Response to VARB1 is \*VARB1=1001\_1111\_1X11\_XXXX\_XXXX\_XXXX\_XXXX\_XXXX

Example: VARB1=h1234 | hFAD31  
Response to VARB1 is \*VARB1=1111\_0101\_1111\_1110\_1000\_0000\_0000\_0000

Example: VARB1=h23 | b1101 001X 001X 1X11  
Response to VARB1 is \*VARB1=1101\_111X\_001X\_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)	<b>Product</b>	<b>Rev</b>
Syntax	See Below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	[ = ], [ & ], [ ~ ], [   ], [ << ], [ >> ], VAR, VARI, VARB		

---

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 or VARI 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 or VARI), the possible combinations are as follows:

positive number ^ positive number = 0  
 positive number ^ zero or a negative number = 1  
 zero or negative number ^ positive number = 1  
 zero or negative number ^ zero or negative number = 0

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 = 0                      X ^ X = X  
 1 ^ 0 = 1                      1 ^ X = X  
 0 ^ 1 = 1                      X ^ 1 = X  
 0 ^ 0 = 0                      0 ^ X = X  
                                     X ^ 0 = X

Example: VARB1=b0000 1111 XXX1 ^ b10XX 10XX 10XX  
 Response to VARB1 is \*VARB1=10XX\_01XX\_XXXX\_XXXX\_XXXX\_XXXX\_XXXX\_XXXX

Example: VARB1=h32FD ^ h6A  
 Response to VARB1 is \*VARB1=1010\_0001\_1111\_1011\_0000\_0000\_0000\_0000

Example: VARB1=h7FFF ^ b1101 1111 0000 1101  
 Response to VARB1 is \*VARB1=0011\_0000\_1111\_0010\_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	Product	Rev
Type	Operator (Bitwise)			
Syntax	See Below		GT6K	6.0
Units	n/a		GV6K	6.0
Range	n/a			
Default	n/a			
Response	n/a			
See Also	[ = ], [ & ], [ ^ ], [   ], [ << ], [ >> ], VAR, VARI, VARB			

The Boolean Not (~) operator performs a logical NOT on the value immediately to its right when used with the VAR or VARI 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 or VARI), the possible combinations are as follows:

~ (positive number) = 0  
 ~ (zero or a negative number) = 1  
 Example: VAR1=~ ( 5 ) ; Result: VAR1=0  
 Example: VAR1=~ (-1) ; Result: VAR1=1

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

Example: VARB1=~ (b0000 1000 1XX1)  
 Response to VARB1 is \*VARB1=1111\_0111\_0XX0\_XXXX\_XXXX\_XXXX\_XXXX\_XXXX  
 Example: VARB1=~ (h32FD)  
 Response to VARB1 is \*VARB1=0011\_1011\_0000\_0100\_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. For example, if the distance has the value \*D+25000, by issuing D~ the new value for distance would be \*D-25000.

---

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

Type	Operator (Bitwise)	<b>Product</b>	<b>Rev</b>
Syntax	See Below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	[ = ], [ & ], [ ^ ], [   ], [ ~ ], [ >> ], VAR, VARI, VARB		

---

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= h0000 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)	<b>Product</b>	<b>Rev</b>
Syntax	See Below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	[ = ], [ & ], [ ^ ], [   ], [ ~ ], [ << ], VAR, VARI, VARB		

---

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**.

<b>[ Send Response to Both Communication Ports</b>			
Type	Communication Interface	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!> [ <command><field1>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	BOT, PORT, ], ECHO, EOL, EOT, LOCK		

The Send Response to All Ports ( [ ] ) command is used to send the response from the command which follows it to both communication ports. If a syntax error occurs, an error message will be sent to both communication ports.

**NOTE:** COM1 refers to the “RS-232/485” or “ETHERNET” connector, and COM2 refers to the “RS-232” connector.

**Example:**

```
[TER          ;Transfer TER Status to both communication ports
```

<b>] Send Response to Alternate Communication Port</b>			
Type	Communication Interface	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!> ] <command><field1>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	BOT, PORT, [, ECHO, EOL, EOT, LOCK		

The Send Response to Alternate Port ( ] ) command is used to send the response from the command which follows it to the alternate port from the one selected. If a report back is requested from port COM1, the response will be sent out port COM2, and vice-versa. If a command is in a stored program, the report will be sent out the alternate port from the one selected by the PORT command. If a syntax error occurs an error message will be sent to the alternate port from the one selected.

**NOTE:** COM1 refers to the “RS-232/485” or “ETHERNET” connector, and COM2 refers to the “RS-232” connector.

**Example:**

```
; *****
; In this example, we place the "]TAS" statement in a program so that
; we can select the port (in this case, "PORT1" selects COM1) as a
; reference. Otherwise, executing "]TAS" outside of a program sends
; the response to whatever port you are not communicating through.
; *****
DEF COM          ; Begin definition of program called "COM"
PORT1           ; Select COM1
TER             ; Transfer TER Status to port COM1
]TAS           ; Transfer TAS Status to port COM2
END            ; End program definition
```





# Command Descriptions

<b>A</b>		<b>Acceleration</b>	
Type	Motion	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>A<r>	GT6K	6.0
Units	r = units/sec/sec (linear motors: see DMEPIT for linear/rotary conversion)	GV6K	6.0
Range	0.0001 - 9999.9999		
Default	10.0000		
Response	A: *A10.0000		
See Also	[ A ], AA, AD, ADA, DRES, ERES, GO, MC, SCALE, SCLA, TSTAT, DMEPIT, IF, VARI, WAIT		

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

**UNITS OF MEASURE and SCALING:** refer to page 16.

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:** 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 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:**

```
SCALE1           ; Enable scaling
SCLA25000        ; Set the acceleration scaling factor for to
                 ; 25000 counts/unit
SCLV25000        ; Set the velocity scaling factor to 25000 counts/unit
SCLD1           ; Set the distance scaling factor 1 count/unit
DEL progA        ; Delete program called progA
DEF progA        ; Begin definition of program called progA
MA0              ; Incremental positioning mode
MC0              ; Preset positioning mode
A10              ; Set the acceleration to 10 units/sec/sec
V1               ; Set the velocity to 1 units/sec
D100000         ; Set the distance to 100000 units
GO1              ; Initiate motion
END              ; End definition of program called progA
```

---

## [ A ] Acceleration Assignment

Type	Assignment or Comparison	<b>Product</b>	<b>Rev</b>
Syntax	See below	GT6K	6.0
Units	units/sec/sec	GV6K	6.0
Range	0.0001 - 9999.9999		
Default	n/a		
Response	n/a		
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=A, where n is the variable number, or A can be used in an expression such as IF(A<25000). The (A) value used in any comparison, or in any assignment statement is the programmed (A) value.

**UNITS OF MEASURE and SCALING:** refer to page 16.

**Example:**

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

---

## AA Average Acceleration

Type	Motion (S-Curve)	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>AA<r>	GT6K	6.0
Units	r = units/sec/sec (linear motors: see DMEPIT for linear/rotary conversion)	GV6K	6.0
Range	0, or 0.0001 - 9999.9999		
Default	10.00 (Default is a constant accel ramp, where AA tracks A. S-curve accel is achieved when AA is changed independent of A. To restore a constant accel ramp and tracking, set AA = 0.)		
Response	AA: *AA10.0000		
See Also	A, AD, ADA, SCALE, SCLA, DMEPIT		

---

The 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*. Refer to page 15 for details on S-curve profiling.

Scaling affects the average acceleration (AA) the same as it does for the maximum acceleration (A). Refer to page 16 for details on scaling.

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

**Example:**

```
; In this example, prog1 executes a pure S-curve and takes 1 second
; to reach a velocity of 5 rps; prog2 executes a trapezoidal profile
; and takes 0.5 seconds to reach a velocity of 5 rps.
SCALE0          ; Disable scaling
DEL Prog1       ; Delete program called prog1
DEF Prog1       ; Begin definition of prog1
MA0             ; Select incremental positioning mode
D40000          ; Set distance to 40,000 positive-direction counts
A10            ; Set max. accel to 10 revs/sec/sec
AA5            ; Set avg. accel to 5 revs/sec/sec
AD10           ; Set max. decel to 10 revs/sec/sec
```

```

ADA5          ; Set avg. decel to 5 revs/sec/sec
V5            ; Set velocity to 5 revs/sec
GO1          ; Execute motion
END          ; End definition of prog1

DEL Prog2     ; Delete program called prog2
DEF Prog2     ; Begin definition of prog2
MA0          ; Select incremental positioning mode
D40000       ; Set distance to 40,000 positive-direction counts
A10          ; Set max. accel to 10 revs/sec/sec
AA10         ; Set avg. accel to 10 revs/sec/sec
AD10         ; Set max. decel to 10 revs/sec/sec
ADA10        ; Set avg. decel to 10 rev/sec/sec
V5           ; Set velocity to 5 revs/sec
GO1          ; Execute motion
END          ; End definition of program #2

```

---

## AD Deceleration

Type	Motion	Product	Rev
Syntax	<a_><!>AD<r>	GT6K	6.0
Units	r = units/sec/sec (linear motors: see DMEPIT for linear/rotary conversion)	GV6K	6.0
Range	0, or 0.0001 - 9999.9999		
Default	10.0000 (by default, AD tracks A until AD is changed independent of A; to restore tracking, set AD = 0)		
Response	AD: *AD10.0000		
See Also	[ A ], A, AA, ADA, DMEPIT, DRES, ERES, GO, IF, LHAD, LSAD, MC, VARI, SCALE, SCLA, TSTAT, WAIT		

---

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

**UNITS OF MEASURE and SCALING:** refer to page 16.

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 (ADØ), then the deceleration will once again track whatever the A command is set to.

**ON-THE-FLY CHANGES:** 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 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:**

```

SCALE1       ; Enable scaling
SCLA25000    ; Set the acceleration scaling factor to 25000 counts/unit
SCLV25000    ; Set the velocity scaling factor to 25000 counts/unit
SCLD1        ; Set the distance scaling factor to 1 count/unit
DEL progA    ; Delete program called progA
DEF progA    ; Begin definition of program called progA
MA0          ; Incremental positioning mode
MC0          ; Preset positioning mode
A10          ; Set the acceleration to 10 units/sec/sec
AD1          ; Set the deceleration to 1 units/sec/sec
V1           ; Set the velocity to 1 units/sec
D100000     ; Set the distance to 100000 units
GO1         ; Initiate motion
END         ; End definition of program called progA

```

---

## [ AD ]                      Deceleration Assignment

Type	Assignment or Comparison	<b>Product</b>	<b>Rev</b>
Syntax	See below	GT6K	6.0
Units	units/sec/sec (linear motors: see DMEPIT for linear/rotary conversion)	GV6K	6.0
Range	0, or 0.0001 - 9999.9999		
Default	n/a		
Response	n/a		
See Also	[A], A, AA, AD, ADA, DRES, ERES, GO, SCALE, SCLA		

---

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=AD where n is the variable number, or [AD] can be used in an expression such as IF(AD<25000). The (AD) value used in any comparison, or in any assignment statement is the programmed (AD) value.

**UNITS OF MEASURE and SCALING:** refer to page 16.

**Example:**

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

---

## ADA                              Average Deceleration

Type	Motion (S-Curve)	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>ADA<r>	GT6K	6.0
Units	r = units/sec/sec (linear motors: see DMEPIT for linear/rotary conversion)	GV6K	6.0
Range	0.0001 - 9999.9999		
Default	10.00 (By default, ADA tracks AA until ADA is changed independent of AA. To restore tracking, set AD = 0.)		
Response	ADA: *ADA10.0000		
See Also	A, AA, AD, DMEPIT, LHADA, LSADA, SCALE, SCLA		

---

The 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*. Refer to page 15 for details on S-curve profiling.

Scaling affects the average deceleration (ADA) the same as it does for the maximum deceleration (AD). Refer to page 16 for details on scaling.

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

**Example:** (refer to the AA command description)

<b>ADDR</b>		<b>Multiple Unit Auto-Address</b>	
Type	Drive Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>ADDR<i>	GT6K	6.0
Units	i = unit number	GV6K	6.0
Range	0 to 99		
Default	0		
Response	ADDR: *ADDR0		
See Also	BAUD, E, PORT		

The ADDR command automatically configures unit addresses for a daisy-chain or multi-drop. The factory default address for a Gem6K drive is address zero (0). The ADDR command allows up to 99 units on a chain to be uniquely addressed. After unique addresses are established, you can address commands to specific units by prefixing the commands with the unit's address followed by an underscore (e.g., 2\_TAS checks the status on unit #2).

The ADDR value is stored in non-volatile memory.

#### RS-232C Daisy Chain:

Sending ADDR*i* to the first unit in the 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 set to (*i* + *n* - 1).

NOTE: ADDR configures only the first unit in the chain.

#### RS-485 Multi-Drop:

To use the ADDR command, you must address each unit individually before it is connected on the multi drop. For example, given that each product is shipped configured with address zero, you could set up a 4-unit multi-drop with the commands below, and then connect them in a multi drop:

1. Connect the unit that is to be unit #1 and transmit the Ø\_ADDR1 command to it.
2. Connect the unit that is to be unit #2 and transmit the Ø\_ADDR2 command to it.
3. Connect the unit that is to be unit #3 and transmit the Ø\_ADDR3 command to it.
4. Connect the unit that is to be unit #4 and transmit the Ø\_ADDR4 command to it.

If you need to replace a unit in the multi drop, send the Ø\_ADDR*i* command to it, where "i" is the address you wish the new unit to have.

To send a Gem6K command from the master unit to a specific unit in the multi-drop, prefix the command with the unit address and an underscore (e.g., 3\_OUTØ turns off output #1 on unit #3). The master unit (if it is not a Gem6K product) may receive data from a multi-drop unit.

For more information on controlling multiple Gem6K Series controller/drives in an RS-232 daisy-chain or RS-485 multi-drop, refer to the *Programmer's Guide*.

#### **Example:**

```
ADDR1           ; Set the address of the first unit in the daisy-chain to 1.
                ; Subsequent units in the chain are automatically numbered
                ; 2, 3, 4, 5, and so on, in their order in the chain.
```

<b>[ AND ]</b>		<b>And</b>	
Type	Operator (logical)	<b>Product</b>	<b>Rev</b>
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	IF, [ NOT ], [ OR ], 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>0 AND VAR2<3) : TPME : 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) : TPME : 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

Type	Assignment or comparison	<b>Product</b>	<b>Rev</b>
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	ANIRNG, TANI		

---

Use the ANI operator to assign the voltage level present at an analog input (ANI) to a variable, or to make a comparison against another value. The ANI value is measured in volts.

The ANI value is derived from the voltage applied to the corresponding analog input and ground. The analog value is determined from a 12-bit analog-to-digital converter. Under the default ANI voltage range, set with ANIRNG, the range of the ANI operator is -10.000VDC to +10.000VDC (see ANIRNG command for optional voltage ranges).

**Syntax:** `VARn=<B>ANI.i` where “n” is the variable number, “<B>” is the number of the I/O brick, and “i” is the I/O brick address where the analog input resides; or ANI can be used in an expression such as `IF(1ANI.2=2.3)`. If no brick identifier (<B>) is provided, it defaults to brick “0”, the onboard analog input. To understand the I/O brick addressing convention, refer to page 8.

**Example:**

```
VAR1=ANI.1      ; Voltage value at onboard analog input is assigned
                 ; to variable 1
VAR2=3ANI.2     ; Voltage value at analog input 2 on I/O brick 3 is assigned
                 ; to variable 2
IF(1ANI.1<8.2)  ; If voltage value at analog input 1 on brick 1 < 8.2V, do the
                 ; commands between the IF statement and the NIF statement.
TREV            ; Transfer revision level
NIF            ; End if statement
```

---

## ANIEN Analog Input Enable

Type	Inputs	Product	Rev
Syntax	To enable only: <a_><!><@><B>ANIEN<.i>=<E> To override only: <a_><!><@><B>ANIEN<.i>=<r>	GT6K	6.0
Units	B = I/O brick number i = input location on I/O brick "B" E = Enable r = volts	GV6K	6.0
Range	B = 0-8 i = 1-32 (dependent on I/O brick configuration) r = -10.000 to +10.000 (voltage override value)		
Default	E (enabled)		
Response	2ANIEN: *2ANIENx,x,x,x,x,x,x,x (SIM slot 1) x,x,x,x,x,x,x,x (SIM slot 2) E,E,E,E,E,E,E,E (SIM slot 3) x,x,x,x,x,x,x,x (SIM slot 4)		
See Also	[ ANI ], ANIMAS, ANIRNG, FOLMAS, TANI, TIO		

---

The ANIEN command enables or disables specific analog inputs on expansion I/O bricks. The default state for each input is the enabled condition. ANIEN can also be used to set analog inputs to specific override voltage levels. To disable an analog input, set an override voltage of 0.

**NOTE:** The onboard analog input (ANI.1) is always enabled, and is not affected by the ANIEN command.

**Performance:** The rate at which the drive samples each analog input depends on how many are enabled on the SIM; each enabled analog input adds 2 ms to the sample rate for all analog inputs on the SIM. For example, if 4 of the 8 analog inputs on a SIM are enabled, the sample rate for any specific input on the same SIM is 8 ms (4 inputs x 2 ms). Disabling input channels increases the performance of the remaining channels; this is important if an input channel is to be used as a Following source (ANIMAS and FOLMAS selection).

**Example:**

```
1ANIEN.9=E,E           ; Enable the 1st & 2nd analog input in SIM slot 2
                       ; (I/O locations 9 & 10) on I/O brick 1.
1ANIEN.11=2.4         ; Override the 3rd analog input in SIM slot 2 (I/O location 11)
                       ; on I/O brick 1 with a voltage of 2.4 volts.
2ANIEN                ; Check status of analog inputs on I/O brick 2. As an example,
                       ; a response of *2ANIENx,x,x,x,x,x,x,x
                       ;           x,x,x,x,x,x,x,x
                       ;           E,E,E,E,E,E,E,E
                       ;           x,x,x,x,x,x,x,x
                       ; indicates that an analog input SIM is installed in slot 3 of
                       ; I/O brick 2 and all eight channels are enabled ("E").
```

---

## ANIMAS Assign Analog Inputs as Following Master

Type	Following	Product	Rev
Syntax	<a_><!>ANIMAS<B-i>	GT6K	6.0
Units	B = I/O brick number i = input location on I/O brick "B"	GV6K	6.0
Range	B = 0-8 i = 1-32 (dependent on I/O brick configuration)		
Default	0-0 (No assignment)		
Response	ANIMAS: *ANIMAS1-1		
See Also	ANIEN, FOLMAS		

---

The ANIMAS command assigns an analog input channel to a specific Following master for use when an ANI master is selected with the FOLMAS command. The ANIMAS command only has an effect if an analog input Following master is selected with a subsequent FOLMAS command (ANIMAS command must be issued before the FOLMAS command).

**Example:**

```
ANIMAS4-17           ; Select the first analog input channel in SIM slot 3
                       ; (I/O location 17) of I/O brick 4 to be used for master
FOLMAS12             ; Define axis 1 to be follower of the analog input
                       ; selected for master
```

## ANIRNG Analog Input Voltage Range

Type	Drive Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!><@><B>ANIRNG<.i><=i>	GT6K	6.0
Units	B = I/O brick number 1st i = input location on I/O brick "B" 2nd i = voltage range selector number	GV6K	6.0
Range	B = 0-8 1st i = 1-32 (dependent on I/O brick configuration) 2nd i = 1 (0 to +5VDC), 2 (-5 to +5VDC), 3 (0 to +10VDC), or 4 (-10 to +10VDC)		
Default	4 (range is set to -10 to +10VDC)		
Response	2ANIRNG: *2ANIRNGx,x,x,x,x,x,x,x 4,4,4,4,4,4,4,4,4 x,x,x,x,x,x,x,x x,x,x,x,x,x,x,x 2ANIRNG.9: *4		
See Also	[ ANI ], ANIEN, JOYCDB, JOYCTR, JOYEDB, SCLA, SCLV, TANI, TIO,		

Use the ANIRNG command to select voltage ranges for specific analog inputs on the expansion I/O bricks connected to your Gem6K. The default range for all analog inputs is -10VDC to +10VDC.

**NOTE:** The onboard analog input (ANI.1) is fixed at -10VDC to +10VDC, and is not affected by the ANIRNG command.

Be aware that changing the analog input voltage range affects these settings:

ANIRNG Setting	Voltage Range	Counts/volt resolution
1	0 to +5VDC	819
2	-5 to +5VDC	410
3	0 to +10VDC	410
4	-10 to +10VDC	205

### Example

```
2ANIRNG.9=3 ; For the 1st analog input on SIM2 of I/O brick 2, select a voltage
; range of 0 to +10VDC
```

## ANO Set Analog Output Voltage

Type	Output	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!><B>ANO<.i>=<r>	GT6K	6.0
Units	B = I/O brick number i = input/output location on brick "B" r = volts	GV6K	6.0
Range	B = 1-8 (depending on I/O brick configuration) i = 1-32 (depending on I/O brick configuration) r = -10.00 to +10.00 (or 0 to 4095 in PLC mode)		
Default	0.0		
Response	n/a		
See Also	[ ANO ], DMONAS, DMONAV, DMONBS, DMONBV, TANO, TIO		

Use the ANO command to assign a voltage of an analog output associated with an analog output SIM.

**NOTE:** See the DMONxx commands for information on controlling the onboard analog monitors.



**Performance:** Each analog output channel is updated once every 16ms (2ms per analog output channel). Each DAC has 10-bit granularity, giving approximately 20 mV/bit over the 20 volt range.

**PLC Mode:** When commanding an analog output using PLC mode, the raw DAC value must be used. To calculate what value to program, use the following formula or table:

$$\text{ANO Value} = (\text{Vout} + 10) \times 4095 / 20$$

Vout	DAC Value
-10V	0
-5V	1024
0V	2048
+5V	3071
+10V	4095

**Example**

```

1ANO.9=8.5      ; Command 8.5 volts on analog output 9, I/O brick 1
                ; (analog output SIM must be present in slot 2 of I/O brick 1)

```

---

**[ ANO ] Analog Output Voltage Assignment**

Type	Outputs; Assignment or Comparison	<b>Product</b>	<b>Rev</b>
Syntax	See below	GT6K	6.0
Units	Voltage	GV6K	6.0
Range	-10.00VDC to +10.00VDC		
Default	n/a		
Response	n/a		
See Also	ANO, TANO, TIO		

---

Use the ANO operand to assign the voltage level present at one of the analog outputs to a variable, or to make a comparison against another value.

The ANO value is derived from the voltage applied to the corresponding analog output and ground. The analog value is determined from a 10-bit digital-to-analog converter and the value set with the ANO command. The range of the ANO operand is -10.00VDC to +10.00VDC.

**Syntax:** VARn=<B>ANO.i where “n” is the variable number, “<B>” is the number of the I/O brick, and “i” is I/O brick address where the analog output resides; or ANO can be used in an expression such as IF(1ANO.2=2.3). If no brick identifier (<B>) is provided, it defaults to 1. To understand the I/O brick addressing convention, refer to page 8.

**PLC Mode:** When assigning or comparing an analog output value in PLC mode, the raw DAC value must be used. To calculate what this value will be, use the following formula or table:

$$\text{ANO Value} = (\text{Vout} + 10) \times 4095 / 20$$

Vout	DAC Value
-10V	0
-5V	1024
0V	2048
+5V	3071
+10V	4095

**Example:**

```

IF(1ANO.9>8.5)  ; If the commanded analog output is greater than 8.5
  WRITE"HIGH"   ; write warning to screen
  NIF
IF(1ANO.9<-8.5) ; If the commanded analog output is less than -8.5
  WRITE"LOW"    ; write warning to screen
  NIF

```

## [ AS ]

### Axis Status

Type	Assignment or Comparison	Product	Rev
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	[ ASX ], ESTALL, GOBUF, GOWHEN, INDUST, SMPER, TAS, TASF, TRGFN, TSTAT, VARB		

Use the AS operator to assign the axis 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 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. The function of each axis status bit is shown below.

Bit # (left to right)	Function (1 = Yes; Ø = No)	GT6K	GV6K	To Clear This Bit:
1	Motion is <u>commanded</u> . (CAUTION: This bit could be cleared even while the motor is still "moving" – e.g., due to end-of-move settling.)	X	X	-----
2	Negative/positive-direction (1 = negative, Ø = positive).	X	X	-----
3	Accelerating (commanded acceleration only – TACC). This bit does not indicate commanded deceleration (bit is set to 0 during decel); to check if the axis is decelerating, the state of TAS bits 1, 3 and 4 should be: TAS1x00.	X	X	-----
4	At commanded velocity (TVEL).	X	X	-----
5	Home Successful (HOM) (YES/NO)	X	X	-----
6	Absolute/Incremental (MA) position mode selected. (1 = MA1, Ø = MA0)	X	X	-----
7	Continuous/Preset (MC) position mode selected. (1 = MC1, Ø = MC0)	X	X	-----
8	Jog Mode selected (JOG) (1 = JOG1, Ø = JOG0)	--	--	-----
9	Joystick Mode selected (JOY) (1 = JOY1, Ø = JOY0)	--	--	-----
10	RESERVED	--	--	-----
11	Position Maintenance Mode	X	--	-----
12	Stall detected (ESTALL)	X	--	-----
13	Drive shut down. <b>NOTE:</b> If operating in the FLTDSB1 mode, a shutdown (DRIVE0 or open the Enable input interlock) will cause a "fault condition" – see note * below.	X	X	DRIVE1
14 *	Drive Faults occurred. Check the TASX response to identify which fault(s) occurred.	X	X	DRIVE1
15	Positive-direction hardware limit hit. This is <u>not</u> a "fault condition".	X	X	LHØ or GO in opposite direction.
16	Negative-direction hardware limit hit. This is <u>not</u> a "fault condition".	X	X	LHØ or GO in opposite direction.
17	Positive-direction Software Limit (LSPOS) Hit.	X	X	LSØ or GO in opposite direction.
18	Negative-direction Software Limit (LSNEG) Hit.	X	X	LSØ or GO in opposite direction.
19	Position Error	X	--	-----

20	RESERVED	--	--	-----
21	RESERVED	--	--	-----
22	RESERVED	--	--	-----
23 *	Position error exceeded (SMPER).	--	X	DRIVE1
24	In Target Zone (defined with STRGTD & STRGTV). This bit is set only after the <i>successful completion</i> of a move (if STRGTD and STRGTV criteria have been satisfied). This bit is usable even if the Target Zone mode is not enabled (STRGTE0).	--	X	GO1
25	Target Zone Timeout occurred (STRGTT).	--	X	GV6K: GO1
26	Change in motion is suspended pending GOWHEN (YES/NO). This bit is cleared if the GOWHEN condition is true, or if STOP (!S) or KILLL (!K or ^K) is executed	X	X	(see description)
27	RESERVED	--	--	-----
28	Registration move initiated by trigger since last GO command. This bit is cleared with the next GO command.	X	X	GO1
29	GOWHEN error occurred. The input state or the position relationship specified in a GOWHEN command was already true when the subsequent GO, FGADV, FSHFC or FSHFD command was executed.	X	X	-----
30	Pre-emptive (OTF) GO or Registration profile not possible	X	X	GO1
31	Compiled GOBUF profile is executing.	X	X	-----
32	RESERVED	--	--	-----

\* **FAULT CONDITIONS:** If one or more of these conditions exist, the drive automatically disables (DRIVE0), it activates any output configured as a fault output, and it opens the dry contact relay (labeled "RELAY COM" and "RELAY N.O.") on the 4 pin removable connector.

**Syntax:** VARBn=AS where n is the binary variable number, or AS can be used in an expression such as IF(AS=b1101), or IF(AS=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 operator, in conjunction with the bit number, is used to specify a specific axis status bit (e.g., VARB1=AS.12 assigns status bit 12 to binary variable 1).

**Example:**

```

VARB1=AS           ; Axis status assigned to binary variable 1
VARB2=AS.12       ; Status bit 12 assigned to binary variable 2
VARB2              ; Response, if bit 12 is set to 1, is
                  ; "*VARB2=XXXX_XXXX_XXX1_XXXX_XXXX_XXXX_XXXX_XXXX"
IF(AS=b111011X11) ; If the axis status contains 1's for
                  ; bits 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(AS=h7F00)      ; If the axis status contains 1's for
                  ; bits 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

```

**[ ASX ] Extended Axis Status**

Type	Assignment or Comparison	Product	Rev
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	[AS], DSTALL, ESTALL, TASX, TASXF, TAS, TASF, VARB		

The ASX command is used to assign the axis 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 that the axis status is being compared against. 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 that the axis status is being compared against. The hexadecimal value itself must only contain the letters A through F, and the numbers 0 through 9.

Bit #	Function (1 = Yes; 0 = No)	GT6K	GV6K	To Clear This Status Bit:
1 *	Motor temperature fault (hardware switch in the motor).	--	X	DRIVE1
2 *	Low voltage fault	X	X	DRIVE1
3 *	Drive over-temperature fault (hardware thermal sensor in drive).	X	X	DRIVE1
4	RESERVED	--	--	-----
5 *	Resolver failed.	--	X	RESET or cycle power
6	RESERVED	--	--	-----
7 *	Motor configuration error occurred (checked on power up). Use TCS to ascertain the cause of the error. This is a fault condition which causes a drive shutdown (DRIVE0).	X	X	Resolve error condition (see TCS) and issue RESET or DRESET or cycle power
8	RESERVED	--	--	-----
9 *	Velocity error limit (SMVER value) has been exceeded.	--	X	DRIVE1
10 *	Bridge fault (hardware signal from the bridge)	X	X	RESET or DRESET or cycle power
11 *	Bridge temperature fault (this is a software control).	--	X	DRIVE1
12 *	Over-voltage.	--	X	RESET or DRESET or cycle power
13-16	RESERVED	--	--	-----
17	Stall detected (DSTALL or ESTALL)	X	--	DRIVE1
18	Override mode was invoked. (See DMVLIM command.)	X	X	DCLRLR or RESET or DRESET or cycle power
19	Bridge is in foldback mode.	--	X	DCLRLR or RESET or DRESET or cycle power
20	Power Dissipation Circuit has been active – not applicable to GV6K-U3, GV6K-U6, and GV6K-U12 drives	X	X	DCLRLR or RESET or cycle power
21 *	Bad Hall state detected. (Use THALL for diagnostics.)	--	X	RESET or cycle power
22 *	Unrecognized hardware — consult factory	X	X	RESET or cycle power
23 *	User Fault input activated (INFNCi-F)	X	X	Deactivate the input
24	Keep Alive active (User supplied +24VDC)	X	X	-----
25	Power dissipation circuit fault (excessive power dissipation)	X	X	DRIVE1
26	RESERVED	--	--	-----
27	RESERVED	--	--	-----
28	Motor configuration warning resulting from trying to run the motor beyond acceptable limits. Use TCS to ascertain the cause of the warning.	X	X	Resolve error condition (see TCS) and issue RESET or DRESET or cycle power
29 *	ORES failure (encoder output or step & direction output exceeds the maximum output frequency).	X	X	DRIVE1
30 *	Motor Thermal model fault.	--	X	Let the motor cool, then issue DRIVE1.
31	Commanded torque/force is at limit (TTRQ = DMTLIM).	--	X	DCLRLR or RESET or DRESET or cycle power
32	RESERVED	--	--	-----

\* **FAULT CONDITIONS:** If one or more of these conditions exist, the drive automatically disables (DRIVE0), it activates any output configured as a fault output, and it opens the dry contact relay (labeled "RELAY COM" and "RELAY N.O.") on the 4 pin removable connector.

**Syntax:** VARbn=ASX where n is the binary variable number, or ASX can be used in an expression such as IF(ASX=b1100), or IF(ASX=h70). 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=ASX.3 assigns status bit 3 to binary variable 1).

**Example:**

```

VARB1=ASX           ; Extended Axis status assigned to binary variable 1
VARB2=ASX.3        ; Extended Axis status bit 3 assigned to binary variable 2
VARB2              ; Response if bit 3 is set to 1:
                   ; "*VARB2=XX1X_XXXX_XXXX_XXXX_XXXX_XXXX_XXXX_XXXX"
IF(ASX=b101XXXXX) ; If the extended axis status contains 1's
                   ; for bits 1 and 3, and a 0 for bit location 2, do the
                   ; IF statement
TREV               ; Transfer revision level
NIF               ; End if statement

```

**[ ATAN() ] Arc Tangent**

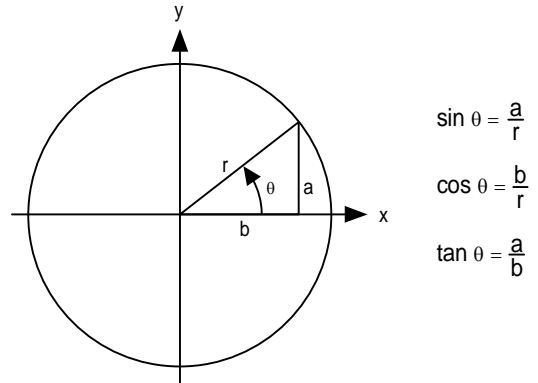
Type	Operator (Trigonometric)	Product	Rev
Syntax	VARi=ATAN(r)	GT6K	6.0
Units	r = real number	GV6K	6.0
Range	0.00000 to ±999,999,999		
Default	none		
Response	n/a		
See Also	[=], [COS], [PI], RADIAN, [SIN], [TAN], VAR		

The 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:

$$\theta = \arctan \frac{a}{b}$$

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:  
 $360^\circ = 2\pi$  radians.



**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:**

```

RADIAN1           ; Enable radian mode
VAR1=ATAN(0.75)  ; Set variable 1 equal to the inverse tangent of 0.75 radians

```

**BAUD Baud Rate**

Type	Communication Interface	Product	Rev
Syntax	BAUD<i>	GT6K	6.0
Units	i = Baud rate	GV6K	6.0
Range	i = 1200, 2400, 4800, 9600, 19200, 38400, or 115200 (COM2 only)		
Default	9600		
Response	BAUD *BAUD9600		
See Also	ADDR, E, PORT		

**NOTE:** The baud rate for the “RS-232/485” (COM1) serial port is fixed at 9600 baud, and is not adjustable by the BAUD command.

BAUD establishes the baud rate for the “RS-232” (COM2) serial port, as selected by the last PORT command. The default is 9600 baud (BAUD9600). The BAUD setting is automatically saved in battery backed RAM. **NOTE:** Changing the baud rate for the “RS-232” port will result in a loss of communication until the baud rate of the terminal is changed accordingly.

**Example:**

```
PORT2          ; Select COM2 ("RS-232") port
BAUD38400      ; Set the baud rate for COM2 to 38400 baud
```

---

## BOT Beginning of Transmission Characters

Type	Communication Interface	<b>Product</b>	<b>Rev</b>
Syntax	<!>BOT<i>,<i>,<i>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	i = 0 - 256		
Default	0,0,0		
Response	BOT: *BOT0,0,0		
See Also	EOT, ERROK, ERBAD, PORT, DRPCHK, EOL, ], [		

---

The BOT command designates the characters to be placed at the beginning of every response. Up to 3 characters can be placed before the first line of a multi-line response, or before all single-line responses. The characters are designated with their ASCII equivalent. 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:** ASCII 256 transmits nothing. ASCII 256 transmits a null ASCII zero character, which is typically used to create a null terminated string in a C language.

For a more complete list of ASCII Equivalents, refer to the ASCII Table in [Appendix B](#).

**Example:**

```
BOT13,10,26    ; Place a carriage return, line feed, and Ctrl-Z before
                ; the first line of a multi-line response, and before
                ; all single line responses
```

---

## BP Set a Program Break Point

Type	Program Flow Control or Program Debug Tool	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>BP<i>	GT6K	6.0
Units	i = break point number	GV6K	6.0
Range	1 - 32		
Default	n/a		
Response	n/a		
See Also	BREAK, C, HALT, K, S, [ SS ], TSS		

---

The 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 32 break points available, BP1 to BP32, 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:**

```
DEF prog1      ; Begin definition of program named prog1
D50000         ; Set distance to 50000 units
MA1           ; Absolute mode
GO1           ; Initiate motion
IF(PC>40000)   ; Compare commanded position to 40000
BP1           ; If the motor position is > 40000 units, set break point #1
NIF           ; End IF statement
D80000         ; Set distance to 80000 units
GO1           ; Initiate motion
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

Type	Program Flow Control	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>BREAK	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
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:**

```

DEF prog1          ; Define a program called prog1
GO1               ; Initiate motion
GOSUB prog2       ; Gosub to subroutine named prog2
GO1               ; Initiate motion
END               ; End program definition
DEF prog2          ; Define a program called prog2
D5                ; Change distance
IF(IN=b1X0)       ; IF condition is: status of input 1 is active (1)
                  ; and input 3 is inactive (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 encoder
NIF               ; End If statement
END               ; End program definition
RUN prog1         ; Execute program prog1
;
; Upon completion of motion, 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 will be initiated.

```

---

## C Continue Command Execution

Type	Program Flow Control	<b>Product</b>	<b>Rev</b>
Syntax	<a_>!C	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
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 Gem6K 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:**

```

PS                ; Stop execution of command buffer until !C command
MA0               ; Select incremental positioning mode
D10000            ; Set distance to 10000 counts
GO1               ; Initiate motion

```

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      ; Set distance to 50000 units
MA0         ; Select incremental positioning mode
GO1         ; Initiate motion
IF(VAR1>6)  ; Compare VAR1>6
BP1         ; If the motor position is > 50000 units, set break point #1
NIF         ; End IF statement
GO1         ; Initiate motion
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, 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      ; Set distance to 50000 units
GO1         ; Initiate motion
!S          ; Stop motion
```

When the Gem6K drive processes the !S command, motion 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.

---

## CERRLG Clear the Error Log

Type	Error Handling	Product	Rev
Syntax	<a_><!>CERRLG	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	TAS, TASX, TDHRS, TDTEMP, TERRLG, TMTEMP		

---

The CERRLG command erases the stored contents of the error log. Clearing the error log is a helpful diagnostic tool; it allows you to start the diagnostic process when the error log is in a known state so that you can check the error log in response to subsequent events.

The error log is updated every time an event occurs, as selected by the ERRORL command. The TERRLG command displays the last ten error conditions that the drive has experienced, as recorded in these status registers:

- TAS (axis status binary report)
- TASX (extended axis status binary report)
- TDHRS (number of hours since the drive was powered up or RESET)
- TDTEMP (measured temperature of the drive in centigrade)
- TMTEMP (estimated temperature of the motor in centigrade for GV6K only; GT6K always reports zero)



---

## COMEXC    Continuous Command Processing Mode

Type	Command Buffer Control	Product	Rev
Syntax	<a_><!>COMEXC<b>	GT6K	6.0
Units	b = 0 or 1	GV6K	6.0
Range	0 = Disable, 1 = Enable		
Default	0		
Response	COMEXC: *COMEXC0		
See Also	[ ! ], A, AA, AD, ADA, COMEXL, COMEXS, D, ERRORP, FOLRD, FOLRN, GO, GOWHEN, MA, MC, V		

---

Use COMEXC to enable or disable Continuous Command Execution Mode. Under default operation (COMEXC0), when a motion command is received, command processing is temporarily paused until the motion is complete. In continuous command execution mode (COMEXC1), however, command processing continues while motion is taking place. **NOTE:** 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 Commands During Motion section in Chapter 1 of the *Programmer's Guide*.

The Continuous Command Processing Mode is useful in the following situations:

- When trying to check the status of inputs while the Gem6K is commanding motion.
- Performing calculations ahead of time, possibly decreasing cycle time.
- Executing buffered on-the-fly acceleration (A, AA), and deceleration (AD, ADA), distance (D), positioning mode (MA & MC), Following ratio (FOLRD & FOLRN), and velocity (V) changes. (The buffered A, AA, AD, ADA, D, FOLRD, FOLRN, MA, MC, or V change can be executed only with a buffered Go (GO) command.) For more information about on-the-fly motion changes, refer to the *Programmer's Guide*.
- Pre-processing the next move while the current move is in progress (see CAUTION note below). This reduces the processing time for the subsequent move to only a few microseconds.

<b>CAUTION: Avoid Executing Moves Prematurely</b>
---

With continuous command execution enabled (COMEXC1), if you wish motion to stop before executing the subsequent move, place a WAIT(AS.1=b0) statement before the subsequent GO command. If you wish to ensure the load settles adequately before the next move, use the WAIT(AS.24=b1) command instead (this requires you to define end-of-move settling criteria — see STRGTE command or the *Target Zone* section in the *Programmer's Guide* for details).

---

### Example:

```
DEL Prog1          ; Delete program called Prog1
DEF Prog1          ; Begin definition of program called Prog1
COMEXC1           ; Enable continuous command execution mode
L50                ; Loop 50 times
D50000            ; Set distance to 50000 counts
GO1               ; Initiate motion
; Normally at this point, the Gem6K drive would wait for the motion to complete
; before processing the next command. However, with continuous command execution
; enabled (COMEXC1), processing will continue with the statements that follow.
IF(IN.1=b1)       ; Check for input #1 becoming active
OUT.3-1           ; If it does, turn on output #3
ELSE              ;
OUT.1-1           ; If input #1 is not on, turn on output #1
NIF
WAIT(AS.1=b0)     ; Wait for no commanded motion
LN                ; End loop
COMEXC0           ; Disable continuous command mode
END               ; End definition of program
```

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

```
DEF vsteps          ; Begin definition of program vsteps
COMEXC1             ; Enable continuous command execution mode
MC1                 ; Set mode to continuous
A10                 ; Set acceleration to 10 rev/sec/sec
V1                  ; Set velocity to 1 rps
GO1                 ; Initiate move
WAIT(VEL=1)         ; Wait for motor to reach continuous velocity
T3                  ; Time delay of 3 seconds
A50                 ; Set acceleration to 50 rev/sec/sec
V10                 ; Set velocity to 10 rps
GO1                 ; Initiate move
T5                  ; Time delay of 5 seconds
S1                  ; Initiate stop of move
WAIT(MOV=b0)        ; Wait for motion to completely stop
COMEXC0             ; Disable continuous command execution mode
END                  ; End definition of program vsteps
```

---

## COMEXL Continue Execution on Limit

Type	Command Buffer Control	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>COMEXL<b>	GT6K	6.0
Units	b = 0 or 1	GV6K	6.0
Range	0 = Disable, 1 = Enable		
Default	0		
Response	COMEXL: *COMEXL0		
See Also	COMEXC, COMEXS, ERROR, LH, LIMLVL, LS		

COMEXL determines whether the command buffer will be saved upon hitting a hardware end-of-travel limit (LH), or a soft limit (LS). If save command buffer on limit is enabled (COMEXL1), 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 (COMEXL0), then every command in the buffer will be discarded, and program execution will be terminated.

#### Example:

```
COMEXL1             ; Save the command buffer if the limit is hit.
```

---

## COMEXR Continue Motion on Pause/Continue Input

Type	Command Buffer Control	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>COMEXR<b>	GT6K	6.0
Units	b = 0 or 1	GV6K	6.0
Range	0 = disable, 1 = enable		
Default	0		
Response	COMEXR: *COMEXR0		
See Also	C, COMEXS, INFNC		

The COMEXR command determines the functionality of programmable inputs defined as pause/continue inputs with the INFNCi-E command. When the input is activated, the command being processed will be allowed to finish executing.

COMEXR0: 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:**

```

COMEXR1          ; Allow both motion and program execution to be paused upon
                  ; receiving a pause input
2INFNC1-E        ; Define input 1 on I/O brick 2 as a pause/continue input

```

---

## COMEXS Continue Execution on Stop

Type	Command Buffer Control	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>COMEXS<i>	GT6K	6.0
Units	i = function identifier	GV6K	6.0
Range	0, 1, or 2		
Default	0		
Response	COMEXS: *COMEXS0		
See Also	COMEXC, COMEXL, COMEXR, INFNC, S		

---

The COMEXS command determines the impact on motion, program execution, and the command buffer when the Gem6K receives a Stop command (S, !S, S1, or !S1) or an external Stop input (an input assigned a stop function with INFNCi-D).

COMEXS0: Under factory default conditions (COMEXS0), when the Gem6K receives a stop command (S, !S, S1, or !S1) or a stop input (INFNCi-D), the following will happen:

- Motion decelerates to a stop, using the present AD and ADA deceleration values. The motion profile cannot be resumed.
- All commands in the Gem6K's command buffer are discarded.
- Program execution is terminated.

COMEXS1: Using the COMEXS1 mode, the Gem6K allows more flexibility in responding to stop conditions, depending on the stop method (see table below).

Stop Method	What Stops?		Resume Motion Profile. (Allow resume with a !C command or a resume input *)	Resume Program. (Allow resume with a !C command or a resume input *)	Save Command Buffer. (Save the commands that were in the command buffer when the stop was commanded)
	Motion	Program			
!S or S	Yes	Yes	Yes	Yes	Yes
!S1 or S1	Yes	No	No	No	Yes
Stop input	Yes	No	No	No	Yes
Pause input * (if COMEXR1)	Yes	Yes	Yes	Yes	Yes
Pause input * (if COMEXR0)	No	Yes	No	Yes	Yes

\* A *Pause* input is an input configured with the INFNCi-E command command. This is also the *Resume* input that can be used to resume motion and program execution after motion is stopped.

COMEXS2: Using the COMEXS2 mode, the Gem6K responds as it does in the COMEXS0 mode, with the exception that you can still use the program-select inputs to select programs (INSELP value is retained). The program-select input functions are: BCD select (INFNCi-B), and one-to-one select (INFNCi-P). For further details on program selection with inputs, refer to INFNC and INSELP.

## [ COS() ] Cosine

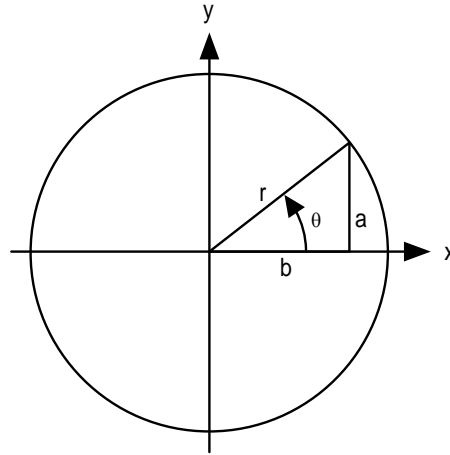
Type	Operator (Trigonometric)	Product	Rev
Syntax	COS(r) (see below)	GT6K	6.0
Units	r = radians or degrees (depending on RADIAN command)	GV6K	6.0
Range	r = 0.00000 - ±17500		
Default	n/a		
Response	n/a		
See Also	[ ATAN ], [ PI ], RADIAN, [ SIN ], [ TAN ], VAR		

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:

$$\cos \theta = \frac{b}{r} \quad (\text{see illustration at right})$$

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

$$360^\circ = 2\pi \text{ radians.}$$

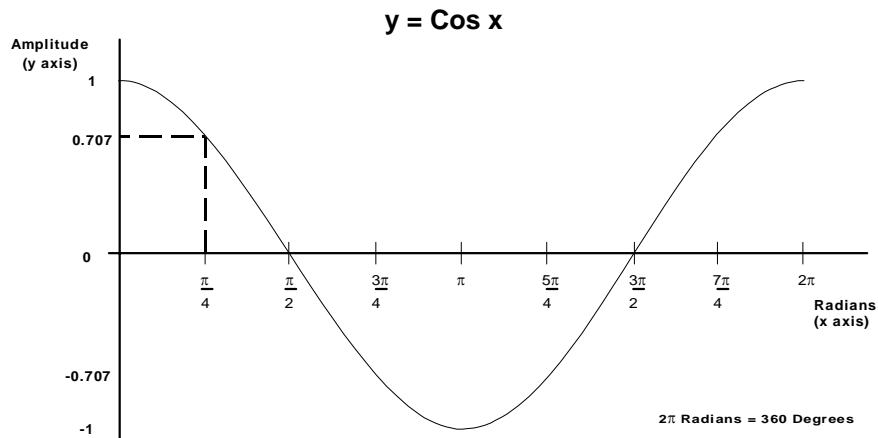


$$\sin \theta = \frac{a}{r}$$

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

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

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



**Syntax:** VAR*i*=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:**

```
VAR1=5 * COS(PI/4) ; Set variable 1 equal to 5 times the cosine of
                   ; pi divided by 4
```

## D Distance

Type	Motion	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>D<r>	GT6K	6.0
Units	r = distance units (scalable by SCLD)	GV6K	6.0
Range	±999,999,999.99999		
Default	4000		
Response	D: *D+4000		
See Also	[ D ], DMEPIT, DRES, ERES, GO, IF, MA, MC, PSET, SCLD, TSTAT, VARI, WAIT		

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 distance by placing a dash or hyphen (-) in front of the distance value (e.g., D-10000). Otherwise, the direction is considered positive. 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.

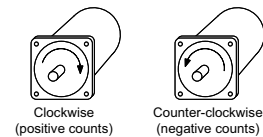
The distance remains set until you change it with a subsequent distance command. Distances outside the valid range are flagged as an error, returning the message \*INVALID DATA-FIELD x, where x is the field number.

**UNITS OF MEASURE and SCALING:** refer to page 16.

**ON-THE-FLY CHANGES:** You can change distance *on the fly* (while motion is in progress) in two ways. One way is to send an immediate distance command (!D) followed by an immediate go command (!GO). The other way is to enable the continuous command execution mode (COMEXC1) and execute a buffered distance command (D) followed by a buffered go command (GO).

### Direction of Motion for Rotary Motors:

Positive distance values (e.g., D20000) represent clockwise motion and negative values (e.g., D-20000) represent counter-clockwise motion. This assumes you connected the motor (and feedback device for servos) according to the *Hardware Installation Guide* instructions.



### Example:

```
SCALE1          ; Enable scaling
SCLA25000       ; Set the acceleration scaling factor for to 25000 steps/unit
SCLV25000       ; Set the velocity scaling factor to 25000 steps/unit
SCLD1           ; Set the distance scaling factor to 1 step/unit
DEL progA       ; Delete program called progA
DEF progA       ; Begin definition of program called progA
MA0             ; Incremental positioning mode
MC0             ; Preset positioning mode
A10             ; Set the acceleration to 10 units/sec/sec
AD1             ; Set the deceleration to 1 units/sec/sec
V1             ; Set the velocity to 1 units/sec
D100000        ; Set the distance to 100000 units
GO1             ; Initiate motion
END             ; End definition of program called progA
```

---

## [ D ] Distance Assignment

Type	Assignment or Comparison	<b>Product</b>	<b>Rev</b>
Syntax	See below	GT6K	6.0
Units	distance units (scalable by SCLD)	GV6K	6.0
Range	±999,999,999.99999		
Default	n/a		
Response	n/a		
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=D where n is the variable number, or [D] can be used in an expression such as IF(D<25000). 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 PC command for steppers, or the PE command for servos.

**UNITS OF MEASURE and SCALING:** refer to page 16.

**Example:**

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

---

## DABSD Enable ABS Damping

Type	Drive Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DABSD<b>	GT6K	6.0
Units	b = enable bit	GV6K	n/a
Range	0 (disable) or 1 (enable)		
Default	0 (disabled)		(applicable only to stepper axes)
Response	DABSD: *DABSD1		
See Also	DACTDP, DDAMPA, DELVIS		

---

The DABSD command enables or disables the ABS damping function. ABS is a damping technique designed for use at very low to zero speed. ABS damping requires no additional user setup or configuration. When enabled (DABSD1), ABS damping takes precedence over electronic viscosity (DELVIS) at speeds less than approximately 0.2 revs/sec (motor dependent). ABS damping can be disabled during acceleration with the DDAMPA0 command (DDAMPA0 is the factory default setting).

**GT6K Damping Features:** The GT6K drive provides damping features that reduce vibration, increase low-speed smoothness, and decrease load settling time. (A procedure for configuring damping settings is provided in the Configuration chapter of the *Hardware Installation Guide*.)

Command	Damping Function	Velocity *	Default	Related Parameters *
DABSD	<u>ABS Damping</u> . Load-independent damping at extreme low speeds. This feature targets applications that require minimal zero-speed settling (e.g., pick-and-place applications).	0 to 0.2 revs/sec **	Disabled	DMTRES, DMTIND
DELVIS	<u>Electronic Viscosity</u> . This feature targets applications that require reduced low-speed velocity ripple and increased smoothness, as well as aggressive low-speed damping. (NOTE: If ABS Damping is enabled, it overrides electronic viscosity in the 0-0.2 rev/sec velocity range.) Start with DELVIS set to zero, and increase until the required performance is achieved.	0 to 3 revs/sec **	Disabled	DMTJ, DMTSTT, DPOLE, DMTIC, DMTIND, LJRAT
DACTDP	<u>Active Damping</u> . This feature targets applications that require high accelerations, fast settling at commanded speed, mechanical vibration rejection, and highly stable (non-resonant) motion. Start configuration with a low DACTDP value, as highly aggressive damping can lead to mechanical failure.	> 3 revs/sec	Enabled, gain = 4	DMTJ, DMTIND, DMTSTT, LJRAT
DDAMPA	<u>Damping During Acceleration</u> . When enabled, ABS Damping and Electronic Viscosity are allowed to function at accel and decel rates greater than 50 revs/sec/sec. If your application requires more responsive acceleration and deceleration (full motor torque) above 50 revs/sec/sec, you can disable this feature; but be aware that doing so increases jerk in your mechanical system.	Affects damping at accelerations > 50 revs/sec/sec	Disabled	n/a

\* These features are based on motor and load parameters that you set up with the configuration utility in Motion Planner (see page 6). *For optimum damping performance, accurate motor and load parameters are required.*  
**NOTE:** If you select a Parker motor with the configuration utility, all of the motor parameters (excluding LJRAT, which sets the load-to-rotor ratio) are automatically set accordingly.

\*\* Actual transition velocity is based on motor and load parameters, and is therefore application dependent.

## DACTDP Active Damping

Type	Drive Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DACTDP<i>	GT6K	6.0
Units	i = gain	GV6K	n/a
Range	0-40; 0 disables active damping		
Default	4		(applicable only to stepper axes)
Response	DACTDP *DACTDP0		
See Also	DABSD, DELVIS, DMTR, LJRAT		

The DACTDP command controls the gain of the Active Damping function for a specific motor and load. Active damping works at speeds greater than 3 revolutions per second.

**NOTE:** To be fully effective, the active damping function requires that you first set the system inertia ratio (LJRAT) and configure your motor parameters. Motor parameters are automatically configured when you select a Parker motor with the DMTR command (if you are not using a Parker motor you must individually configure each command listed in the DMTR command description). **With a setting of DACTDP20**, the nominal gains (calculated based on LJRAT and the motor parameters) give the best performance over the entire speed range, but you may adjust the DACTDP setting further as your application warrants.

An overview of the GT6K's damping features is provided in the DABSD command description

---

## [ DAT ] Data Assignment

Type	Data Storage	<b>Product</b>	<b>Rev</b>
Syntax	DATi	GT6K	6.0
Units	i = data program #	GV6K	6.0
Range	1-50		
Default	n/a		
Response	n/a		
See Also	DATA, [ DATP ], DATPTR, DATRST, DATTCH		

---

The 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 DAT can be used as a command argument such as A(DAT1)

If the data is to be loaded into a command field, the DAT command must be placed within parentheses (e.g., AD(DAT2)). If the data is loaded into a variable, parentheses are not required. (e.g., VAR1=DAT2). **Rule of Thumb for command value substitutions:** If the command syntax shows that the command field requires a real number (denoted by <r>) or an integer value (denoted by <i>), you can use the DAT substitution (e.g., HOMV(DAT1)).

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

Type	Data Storage	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DATA=<r>, <r>, <r>, <r>	GT6K	6.0
Units	r = data value	GV6K	6.0
Range	±999,999,999.99999999		
Default	n/a		
Response	n/a		
See Also	[ DAT ], [ DATP ], DATPTR, DATRST, DATTCH, MEMORY		

---

The 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

Type	Data Storage	<b>Product</b>	<b>Rev</b>
Syntax	DATPi	GT6K	6.0
Units	i = data program number	GV6K	6.0
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:**

```

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 acceleration.
                  ; Axis acceleration = 1
V(DAT5)          ; Load data from data program 5 and store in axis velocity.
                  ; Axis velocity = 2
D(DAT5)          ; Load data from data program 5 and store in axis distance.
                  ; Axis distance = 3
A(DAT5)          ; Load data from data program 5 and store in axis acceleration.
                  ; Axis acceleration = 4
A(DAT5)          ; Load data from data program 5 and store in axis acceleration.
                  ; Axis acceleration = 5.62

```

---

## DATPTR      Set Data Pointer

Type	Data Storage	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DATPTRi,i,i	GT6K	6.0
Units	n/a	GV6K	6.0
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 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=2.0,4.0,8.0,64.0
*DATA=0.0,0.0,0.0,0.0
*DATA=0.0,0.0,0.0,0.0
*DATA=0.0,0.0,0.0

```

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.

**Example:** (See Also: DATSIZ command)

```

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 acceleration.
                  ; Axis acceleration = 1
V(DAT5)           ; Load data from DATP5 and store in axis velocity.
                  ; Axis velocity = 2
D(DAT5)           ; Load data from DATP5 and store in axis distance.
                  ; Axis 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 acceleration.
                  ; Axis acceleration = 1
A(DAT5)           ; Load data from DATP5 and store in axis acceleration.
                  ; Axis acceleration = 2

```

---

<b>DATRST</b>		<b>Reset Data Pointer</b>		
Type	Data Storage		<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DATRST<i>,<i>		GT6K	6.0
Units	n/a		GV6K	6.0
Range	1st i = program # 1 to 50, 2nd i = data element # 1 to 6500			
Default	n/a			
Response	n/a			
See Also	[ DAT ], DATA, [ DATP ]			

---

The 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:**

```

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 acceleration.
                  ; Axis acceleration = 1
V(DAT5)           ; Load data from data program 5 and store in axis velocity.
                  ; Axis velocity = 2
D(DAT5)           ; Load data from data program 5 and store in axis distance.
                  ; Axis 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 acceleration.
                  ; Axis acceleration = 1
A(DAT5)           ; Load data from data program 5 and store in axis acceleration.
                  ; Axis acceleration = 2

```

---

## DATSIZ Data Program Size

Type	Data Storage	Product	Rev
Syntax	<a_><!>DATSIZi<,i>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	1st i = program # 0 - 50 (0 = disable) 2nd i = data element # 1 - 6500		
Default	0,1		
Response	n/a		
See Also	[ DAT ], DATPTR, [ DATP ], DATTCH		

---

The 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 0 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 DATSIZ5,144 command and DATP5 already exists, first delete DATP5 with the DEL DATP5 command and then issue the DATSIZ5,144 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=0.0,0.0,0.0,0.0 commands up to the size indicated in the second integer.)

Each data statement, which contains four data elements, uses 43 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=0.0,0.0,0.0,0.0
*DATA=0.0,0.0,0.0,0.0
*DATA=0.0,0.0,0.0,0.0
*DATA=0.0
```

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:**

```

DEL DATP5           ; Delete existing data program #5 (DATP5)
DATSIZ5,200        ; Create data program #5 (DATP5) with 200 data elements
DEF TEACH          ; Begin definition of program called TEACH
COMEXC0           ; Disable continuous command execution mode
MA1               ; Enable the absolute positioning mode
HOM1              ; Home (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
JOY1              ; Enable joystick mode so that you can start moving the
                  ; axis into position with the joystick. Command processing stops
                  ; here until you activate IN.2 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 in variable #1
DATTCH1           ; Store variable #1 into consecutive data elements
WAIT(IN.2=b0)     ; 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
HOM1              ; Home (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)           ; Read position data from the data program to the distance command
GO1               ; 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

Type	Data Storage	Product	Rev
Syntax	<a_><!>DATTCHi<,i,i,i>	GT6K	6.0
Units	i = number of a numeric variable	GV6K	6.0
Range	i = 1-225		
Default	n/a		
Response	n/a		
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 DATTC1, 2, 3, 4 command is executed).

```
*DATA=2.0,4.0,8.0,64.0
*DATA=0.0,0.0,0.0,0.0
*DATA=0.0,0.0,0.0,0.0
*DATA=0.0,0.0,0.0
```

**Example:** Refer to the DATSIZ command.

---

## DAUTOS

### Auto Current Standby

Type	Drive Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DAUTOS<r>	GT6K	6.0
Units	r = percent reduction of motor current	GV6K	n/a
Range	0.00-99.99 : ±0.01		
Default	0.00 (no current reduction)		
Response	DAUTOS *DAUTOS0		(applicable only to stepper axes)
See Also			

---

The DAUTOS command allows you to let the motor cool when it is not moving. When automatic current standby is set to a value other than 0.00 (default), the motor current will be reduced by that percentage when the drive has not issued a motion command for one second. Full commanded current is restored upon the first motion command that the drive issues.

**WARNING:** Motor torque is reduced when the motor current is reduced. Applications with vertical loads or loads that require holding torque at zero speed should not use this feature.

---

## DCLEAR

### Clear Display

Type	Display (RP240) Interface	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DCLEARi	GT6K	6.0
Units	n/a	GV6K	6.0
Range	i = 0 (clear all lines), 1 (clear line 1), or 2 (clear line 2)		
Default	n/a		
Response	n/a		
See Also	DLED, DPASS, DPCUR, DSTP, DVAR, DVARB, DVARI, DWRITE		

---

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).

---

## DCLRLR

### Clear the Latched Status Register Bits

Type	Drive Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DCLRLR	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	DMODE, DMTLIM, DMVLIM, DIFOLD, TASX, TTRQ		

---

TASX status register bits 18, 19, 20 and 31 (see descriptions below) indicate conditions in which drive protective software has engaged, but drive operation continues. These status bits remain set (*latched*), regardless of whether the conditions persist, and are cleared with the DCLRLR command or when you issue a RESET or DRESET command, cycle power, or activate the Reset input.

- TASX bit 18 Commanded Velocity Exceeds DMVLIM Limit (GV6K and GT6K):  
 Bit 18 is set when the velocity demand from the internal Gem6K control loops exceeds the limits set by the DMVLIM command. The Gem6K drive responds to this condition by invoking the “Override Mode,” in which the drive software clamps the maximum allowable velocity command to the value set by DMVLIM. The Override Mode feature is applicable to the Gem6K drive in all operating modes (DMODE).
- TASX bit 19 Bridge is in Foldback Mode (GV6K only):  
 When a GV6K drive produces more than its rated continuous current, a software algorithm determines on an ongoing basis the amount of power being delivered by the drive to the motor. When this value exceeds the safe threshold for the drive, the drive either goes into Foldback Mode or faults, depending on the DIFOLD command. For motion profiles that push the limits of the drive’s capabilities, the drive might go into Foldback Mode for a short period of time. Bit 19 stays latched, however, so that the user can determine that foldback occurred.
- TASX bit 20 Power Dissipation Circuit Active (GV6K and GT6K):  
 When a Gem6K drive attempts to slow a motor down, the stored energy in the motor and load must be absorbed by the drive. This regenerative energy will increase the bus voltage in the drive until either a regenerative power dissipation circuit dissipates the energy or a drive over-voltage fault (reported in TASX bit 13) occurs. In all GT6K drives, the GV6K-L3, GV6K-H20 and GV6K-H40, internal regenerative power dissipation circuitry is provided to dissipate this energy; when this circuitry is activated, TASX bit 20 is set and latched. In the GV6K-U3, GV6K-U6, and GV6K-U12 drives, the external ‘Gemini Power Dissipation Module’ or ‘GPDM’ option can be used to dissipate this energy — TASX bit 20 does not get set for these drives.
- TASX bit 31 Commanding Maximum Torque/Force (GV6K):  
 When the GV6K’s commanded torque/force reaches the limit set by DMTLIM (TTRQ = DMTLIM), TAS bit #31 is set. This is not considered a fault condition.

---

<b>DDAMPA</b>		<b>Damping During Acceleration/Deceleration</b>	
Type	Drive Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DDAMPA<b>	GT6K	6.0
Units	b = enable bit	GV6K	n/a
Range	0 (disable damping) or 1 (enable damping)		
Default	0 disabled)		(applicable only to stepper axes)
Response	DDAMPA: *DDAMPA1		
See Also	DABSD, DELVIS		

---

When Damping During Acceleration is enabled (DDAMPA1), ABS damping (DABSD) and Electronic Viscosity (DELVIS) function normally.

If your application requires more responsive acceleration and deceleration, you can disable Damping During Accel/Decel with the DDAMPA0 command (DDAMPA0 is the factory default setting). This disables ABS damping and Electronic Viscosity during acceleration and deceleration rates greater than 50 revs/sec when the commanded speed exceeds 0.03 revs/sec.

**Be aware** that the DDAMPA0 mode allows increased jerk in your mechanical system.

An overview of the GT6K’s damping features is provided in the DABSD command description.

---

## DEF

### Begin Program/Subroutine Definition

Type	Program or Subroutine Definition	Product	Rev
Syntax	<a_><!>DEF<t>	GT6K	6.0
Units	t = alpha text string (name of a program)	GV6K	6.0
Range	text string of 6 characters or less		
Default	n/a		
Response	n/a		
See Also	\$, DEL, END, ERASE, GOBUF, GOSUB, GOTO, MEMORY, PCOMP, PLCP, PRUN, RUN, [ SS ], TDIR, TMEM, TPROG, TSS, TSTAT		

---

The DEF command is the beginning of a program 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. Refer to the MEMORY command description for information on program size restriction and total number of programs possible per product.

All programs are stored in a binary fashion within the Gem6K. 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.

<b>NOTE</b>
-------------

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 DEFhom1, because it is impossible for the operating system to distinguish the hom1 subroutine call from the HOM1 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 compiled motion profile must be compiled with the PCOMP command before it can be executed with the PRUN command; and a compiled PLC (PLCP) program must be compiled with PCOMP before it can be executed with the SCANP or PRUN command.

The GOTO and GOSUB commands can be used within a program/subroutine to go to another program/subroutine.

**NOTE:** Program, compiled profile, or subroutine names must be deleted (DEL) before they can be redefined.

**Example:**

```
DEL pick          ; Delete program named pick
DEF pick          ; Begin definition of program named pick
GOL               ; Initiate motion
END               ; End program definition
RUN pick          ; Execute program pick
```

<b>DEL</b>		<b>Delete a Program/Subroutine</b>	
Type	Program or Subroutine Definition	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DEL<t>	GT6K	6.0
Units	t = alpha text string (name of a program)	GV6K	6.0
Range	text string of 6 characters or less		
Default	n/a		
Response	n/a		
See Also	\$, DEF, END, ERASE, GOSUB, GOTO, RUN		

The Delete a Program/Subroutine (DEL) command removes a program 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. The DEL command can be placed inside a program (e.g., to delete a DATP program).

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

**Example:**

```
DEF pick          ; Begin definition of program named pick
GO1              ; Initiate motion
END              ; End program definition
RUN pick         ; Execute program pick
DEL pick         ; Deletes program named pick
```

<b>DELVIS</b>		<b>Electronic Viscosity</b>	
Type	Drive Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DELVIS<i>	GT6K	6.0
Units	i = gain	GV6K	n/a
Range	0-7 (0 disables Electronic Viscosity)		
Default	0 (disabled)		(applicable only to stepper axes)
Response	DELVIS *DELVIS0		
See Also	DACTDP, DABSD, DDAMPA, DMTR, LJRAT		

When the DELVIS command is set to a non-zero value (DELVIS1 through DELVIS7), Electronic Viscosity is invoked at speeds below 3 revs/sec. Electronic Viscosity is superseded by the ABS damping function (enabled with the DABSD1 command) at speeds below approximately 0.2 revs/sec.

If Damping During Acceleration is disabled (DDAMPA0), Electronic Viscosity is disabled during accelerations greater than 50 revs/sec/sec (DDAMPA0 is the factory default setting).

An overview of the GT6K's damping features is provided in the DABSD command description.

**NOTE:** To be fully effective, the electronic viscosity function requires that you first set the system inertia ratio (LJRAT) and configure your motor parameters. Motor parameters are automatically configured when you select a Parker motor with the configuration tool in Motion Planner (if you are not using a Parker motor you must individually configure each command listed in the DMTR command description). With a setting of DELVIS5, the nominal gains (calculated based on LJRAT and the motor parameters) give the best performance over the entire speed range, but you may adjust the DELVIS setting further as your application warrants.



---

## DHALL      Disable Hall Sensor Checking

Type	Drive Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DHALL<i>	GT6K	n/a
Units	i = control option #	GV6K	6.0
Range	0 (Enable Hall Error Checking) 1 (Disable Hall Error Checking)		(applicable only to servo axes)
Default	0		
Response	DHALL:      *DHALL0		
See Also	SHALL, THALL, TASX		

---

**NOTE:** When issued, this command takes effect immediately.

**Encoder Motors (with SFB1):** The DHALL command controls whether or not the Hall error checking routines are invoked once the Gemini drive determines a valid absolute position of the motor at initialization.

These error routines include checks:

1. to see if all three Hall sensors are reporting valid states (000 and 111 are *not* valid).
2. to see if the Hall state signal order is correct for either positive (clockwise) or negative (counter clockwise) motion. The Hall state order is 6, 2, 3, 1, 5, 4, 6, 2, 3, 1, 5, 4, 6 (and so on) for a motor turning in the positive direction.
3. to see if the encoder reports a position that is within +/- 90 electrical degrees from the Hall sensor reading.

If any or all of these three conditions are not met for more than 40 consecutive samples, the drive will fault with BIT21 of TASX set.

**NOTE:** At start up, the Hall signals are used to determine the initial position of the rotor with an accuracy of +/- 30 electrical degrees regardless of how DHALL is set. Once the motor begins to rotate and after it passes through another valid Hall state, the motor commutation sequence switches over to using the encoder only and therefore no longer requires the use of the Hall sensors. At this point a check is made to see if DHALL is set to 0 (Hall checking on) or 1 (Hall checking off.)

For a complete description on how to troubleshoot Hall sensors, especially for non-Compumotor motors, refer to the *Hardware Installation Guide* section on using non-Compumotor motors.

**Resolver Motors (with SFB4):** If you use a resolver motor, DHALL is not applicable.

---

## DIBW      Current Loop Bandwidth

Type	Tuning	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DIBW<i>	GT6K	n/a
Units	i = Hz	GV6K	6.0
Range	0-5000 (motor dependent)		(applicable only to servo axes)
Default	0 (DIBW of 0 results in motor configuration error)		
Response	DIBW:      *DIBW1000		
See Also	DMTLIM, DMTSCL, DMVLIM, DNOTAF, DNOTAQ, DNOTLD, DNOTLG, DPBW, SGIRAT, TASX, TCS, TGAIn, TSGSET		

---

**AUTO-SETUP:** This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6). If you did not use the configuration utility or are not using a Parker Motor, this command is set to zero and you will have to manually set this parameter. (Refer to DMTR for a list of auto-configured commands.) If the drive is powered up when this command is set to zero (for instance, if RFS is executed), the drive reports a motor configuration error with TASX bit 7, writes -32259 to TCS, and shuts down the drive (DRIVE0).

The DIBW command sets the bandwidth of the current loop, in hertz. The drive current will be progressively less responsive to inputs or disturbances above this frequency. Fast, short moves may require higher settings, while systems with mechanical resonance may require lower settings, or the use of filters (see DNOTAF, DNOTAQ, DNOTBF, DNOTBQ, DNOTLD, DNOTLG).

Low current loop bandwidth can limit the bandwidth and stiffness that can be attained in the velocity and position loops. High bandwidths can emphasize resonance and system noise, add to heating of both motor and drive, and increase acoustic noise produced by the motor.

**NOTE:** Attempting to set this value too low for the selected motor will result in a motor configuration error. This will set TASX bit #7 and write error -32259 in the TCS configuration status register.

**Working with servo gains.**

- Servo tuning process: refer to the *Hardware Installation Guide*.
- Check the values of all active gains (DIBW is one of many servo gains): use TGAIN.
- Creating and invoking gain sets: see SGSET, SGENB, TGAIN, TSGSET.

---

**DIFOLD      Current Foldback Enable**

Type	Drive Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DIFOLD<b>	GT6K	n/a
Units	b = enable bit	GV6K	6.0
Range	0 (disable) or 1 (enable)		
Default	0		
Response	DIFOLD: *DIFOLD0		(applicable only to servo axes)
See Also			

---

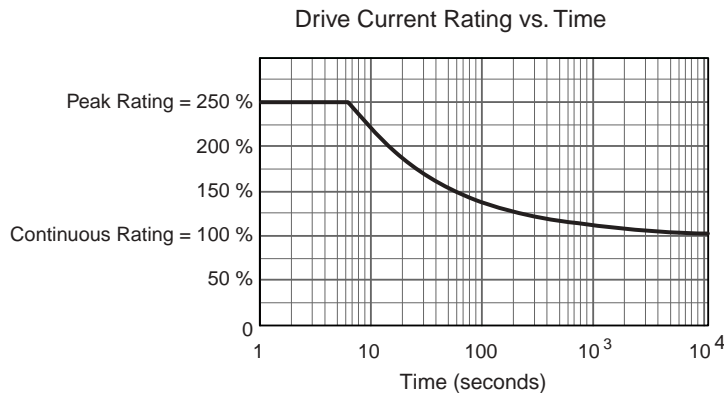
The DIFOLD command enables (1) or disables (0) the drive’s current foldback protection feature. The current foldback feature reduces the drive’s continuous current output by 20% when sustained current has the potential to overheat the drive.

Each drive has the following specifications. Note that current ratings are for the *drive*, not for the *motor*.

	Units	GV6K-L3	GV6K-U3	GV6K-U6	GV6K-U12	GV6K-H20	GV6K-H40
Drive Continuous Current Rating (100%)	amps *	3	3	6	12	20	40
Drive Peak Current Rating**	amps *	7.5	7.5	15	30	50	100
Maximum Time at Peak Current Rating	seconds	6	6	6	6	6	6

\* peak of the sinewave  
 \*\* peak rating is 250% of continuous rating

If your drive is operating above its continuous rating, use the figure below to predict the number of seconds until foldback will occur. For example, the figure shows that at the drive’s peak current rating (250% of continuous), foldback will occur after six seconds.



<b>DIGN</b>		<b>Current Loop Gain</b>	
Type	Tuning	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DIGNc<r>	GT6K	6.0
Units	c = gain identifier letter (required); r = gain value	GV6K	n/a
Range	c = A, B, C, or D; DIGNA, DIGNB, DIGNC : r = 0.000 to 15.000 : ±0.001 DIGND : r = 0.000 to 1.000 : ±0.001	(applicable only to stepper axes)	
Default	r = 0.000 (DIGNc of 0 results in motor config. warning)		
Response	DIGNA: *DIGNA2.306		
See Also	DMTR, TASX, TCS		

**AUTO-SETUP:** This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6). If you did not use the configuration utility or are not using a Parker Motor, this command is set to zero and you will have to manually set this parameter. (Refer to DMTR for a list of auto-configured commands.) If the drive is powered up when this command is set to zero (for instance, if RFS is executed), the drive reports a motor configuration warning with TASX bit 28, and writes a value to the TCS register (46 for DIGNA, 47 for DIGNB, 48 for DIGNC, or 49 for DIGND).

Use the DIGN command to set values for the gain terms in the stepper current loop, which optimizes drive performance for a specific motor.

#### Setting DIGN values for Non-Parker Motors:

##### NOTES

- The drive must be disabled (DRIVE0) before making any changes to the DIGN values.
- When making changes to DIGNA, DIGNB and DIGNC values, the ratio between the values must remain constant. That is, multiply or divide all three gain values by the same amount.
- Increasing the DIGN values can improve system performance; however, setting an excessive DIGN value will cause the motor to “sing” (emit a high-pitched squeal or screech) when it is at rest and to heat unnecessarily

Calculating initial values \* : 
$$DIGNA = \frac{\text{Inductance (in mH)}}{2}$$

$$DIGNB = DIGNA * 0.0896$$

$$DIGNC = DIGNA * 0.3578$$

$$DIGND = 0.98 \text{ (this value will not be changed again)}$$

\* If these initial values cause the motor to “sing,” immediately disable the drive (DRIVE0) and lower DIGNA, DIGNB and DIGNC by the same factor; repeat as necessary.

If motor performance is not as high as expected, increase DIGNA, DIGNB and DIGNC by the same factor (keeping the initial ratio) until system performance is acceptable.

For additional assistance in determining DIGNc values for your motor, please consult the factory.

<b>DJOG</b>		<b>Enable RP240 Jog Mode</b>	
Type	Display (RP240) Interface	<b>Product</b>	<b>Rev</b>
Syntax	<a><!>DJOG<b>	GT6K	6.0
Units	b = 0 or 1	GV6K	6.0
Range	0 = disable, 1 = enable		
Default	0		
Response	DJOG: *DJOG0		
See Also	JOG, JOGA, JOGAA, JOGADA, JOGADA, JOGVH, JOGVL		

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 axis, and then return to program execution.

The `DJOG1` command enables the RP240 jog mode. Once the RP240 jog mode is enabled, you can use the RP240 arrow keys to jog the axis. 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, 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 `!DJOG0` 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.

---

<b>[ DKEY ]</b>		<b>Value of RP240 Key</b>	
Type	Display (RP240) Interface; Assignment or Comparison	<b>Product</b>	<b>Rev</b>
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	DCLEAR, DPCUR, [ DREAD ], DREADF, DREADI, DVAR, DWRITE		

---

The `DKEY` operator allows you to read the current state of the RP240 key-pad and use it in comparison commands (e.g., `IF`, `WHILE`, etc.) or variable assignments. **NOTE:** If two or more keys are pressed simultaneously, `DKEY` will report -1.

**Syntax:** `VARn=DKEY` where “n” is the variable number,  
or `DKEY` can be used in an expression such as `IF (DKEY=-1)`

The value reported by the `DKEY` command is defined by the following table:

<u>Value of DKEY</u>	<u>Key currently active</u>
-1	None or multiple keys
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	.
11	+/-
12	C/E
13	ENTER
14	Menu Recall
15	STOP
16	PAUSE
17	CONTINUE
21	F1
22	F2
23	F3
24	F4
25	F5
26	F6

## DLED Turn RP240 Display LEDs On/Off

Type	Display (RP240) Interface	Product	Rev
Syntax	<a_><!>DLED<b><b><b><b><b><b><b><b>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	b = 0 (off) or 1 (on)		
Default	n/a		
Response	DLED: *DLED1101_0001		
See Also	DCLEAR, DPASS, DPCUR, DSTP, DVAR, DVARB, DVARI, DWRITE		

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:

```
DLED11XXXX01 ; Turn on LEDs 1, 2, and 8; turn off LED 7; leave LEDs 3,4,5,
               ; and 6 unchanged
VARB1=b10101010 ; 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
```

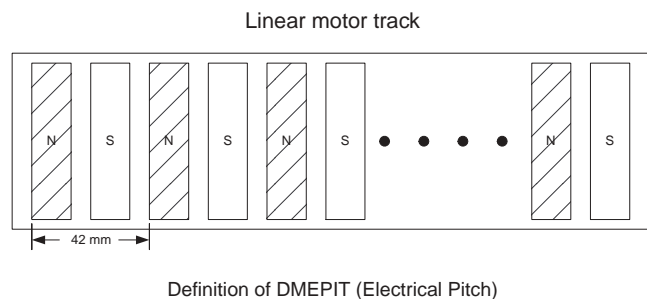
## DMEPIT Motor Electrical Pitch

Type	Motor (Linear only)	Product	Rev
Syntax	<a_><!>DMEPIT<r> (does not take effect until RESET, DRESET or cycle power)	GT6K	n/a
Units	r = millimeters	GV6K	6.0
Range	0 to 327.68 : ±0.01		(applicable only to servo axes)
Default	0		
Response	DMEPIT: *DMEPIT40.00		
See Also	D, DRES, ERES		

**NOTE:** This command is only applicable if you use the Motion Planner Setup Wizard, which uses the DMEPIT value to convert linear units to rotary units before downloading commands to the drive. The drive *always* expects rotary units. If you are *not* using the Motion Planner Setup Wizard, this command is not applicable, and you must convert your linear units to rotary units.

**AUTO-SETUP:** This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6). If you did not use the configuration utility or are not using a Parker linear servo motor, this command is set to 0 and assumes a rotary motor. Refer to DMTR for a list of auto-configured commands. For more information on linear motion, see the *Programmer's Guide*.

The DMEPIT command sets the electrical pitch of the magnets for use with permanent magnet brushless linear motors. The DMEPIT value is required to convert between linear units and rotary units. The electrical pitch can equate to one revolution in a rotary motor. Mechanically, the definition of the electrical pitch is the linear distance between two magnets comprising a full magnetic cycle. The illustration shows an example of an electrical pitch of 42mm (DMEPIT42).



For all rotary motors, DMEPIT is set to zero.

### Converting Between Rotary and Linear Values

The Gem6K drive operates in rotary units; therefore, it expects to receive commands in rotary units and reports operating conditions in rotary units. The setup wizard in Motion Planner (page 6) makes it easy to perform the setup in linear units. The setup wizard automatically converts your setup parameters (in linear units) to the appropriate Gem6K code in rotary units. You then download the generated code/file to the drive. If you are communicating to the Gem6K over a live serial link, you must convert certain command values from linear to rotary units before you send them to the drive. Likewise, when you query the drive for

certain conditions, or if you upload the configuration file from the drive, the command values are reported in rotary units.

Use the following table as a guide.

Command/Parameter	To convert rotary to linear, multiply by:	To convert linear to rotary, multiply by:
DMTKE (motor voltage constant)	$\frac{60}{DMEPIT}$	$\frac{DMEPIT}{60}$
DMTW (motor rated speed)	$\frac{DMEPIT}{1000}$	$\frac{1000}{DMEPIT}$
DMVLIM (velocity limit)	1000	DMEPIT
SMVER (max. allowable velocity error)		
TACC (display commanded accel)		
TVEL (display commanded velocity)		
TVELA (display actual velocity)		
TVE (display velocity error)		
Velocity & Accel/Decel (e.g., V, A, AD)		
DMTD (motor damping)	$\frac{(2 \cdot \pi \cdot 1000)^2}{(DMEPIT)^2}$	$\frac{(DMEPIT)^2}{(2 \cdot \pi \cdot 1000)^2}$
DMTJ (motor/forcer mass)		
LDAMP (load damping)		
DMTLIM (force limit)	$\frac{2 \cdot \pi \cdot 1000}{DMEPIT}$	$\frac{DMEPIT}{2 \cdot \pi \cdot 1000}$
DMTSCL (force scaling)		

If you are constructing your own motor data files, use the formulas from the table below. The key conversion parameter is  $r$  and is defined as:

$$r = \frac{DMEPIT (mm)}{2\pi \cdot 1000}$$

	Linear Motor	Convert to rotary units
<b>General Motion Equation</b>	$F = Ma + Dv$	$T = J\alpha + B\omega$
<b>Position</b>	$x = m$	$\theta = \frac{x}{r} = \text{radians}$
<b>Velocity</b>	$v = \frac{m}{\text{sec}}$	$\omega = \frac{v}{r} = \frac{\text{radians}}{\text{sec}}$
<b>Acceleration</b>	$a = \frac{m}{\text{sec}^2}$	$\alpha = \frac{a}{r} = \frac{\text{radians}}{\text{sec}^2}$
<b>Force / Torque</b>	$F = N$	$T = F \cdot r = Nm$
<b>Mass / Inertia</b>	$M = kg$	$J = m \cdot r^2 = kgm^2$
<b>Damping</b>	$D = \frac{N}{\frac{m}{\text{sec}}}$	$B = D \cdot r^2 = \frac{Nm}{\frac{\text{radians}}{\text{sec}}}$
<b>Flux constant</b>	$K_{e\_linear} = \frac{V_{peak,\phi-\phi}}{\frac{m}{\text{sec}}}$	$K_e = r \cdot K_{e\_Linear} \cdot \frac{2\pi 1000}{60} = \frac{V_{peak,\phi-\phi}}{krpm}$

---

## DMODE Drive Control Mode

Type	Drive Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DMODE<i>	GT6K	6.0
Units	i = control mode setting	GV6K	6.0
Range	1-17 (see table below)		
Default	12		
Response	DMODE: *DMODE12		
See Also	DRES, SRSET		

---

Use the `DMODE` command to select the drive control mode for your Gem6K drive. Refer to the table below for drive mode descriptions and drive compatibility.

DMODE	Mode	Description	GT6K	GV6K
1 – 10	RESERVED	-----	--	--
11	Feedback Alignment Mode	(requires a resolver card to be installed) This mode allows you to automatically set the resolver offset angle with the <code>SRSET</code> command.	--	<b>X</b>
<b>12</b>	Controller/Drive — <b>default for GT6K &amp; GV6K drives</b>	Programmed motion using internal trajectory generator.	<b>X</b>	<b>X</b>
13	Autorun	Rotates the motor at 1 rps/mps. Current is reduced by 10%. Refer also to the <i>Hardware Installation Guide</i> .	<b>X</b>	<b>X</b>
14	RESERVED	-----	--	--
15	Torque/Force Tuning Mode	10 Hz step input of 25% rated motor current for torque/force mode tuning. *	--	<b>X</b>
16	Velocity Tuning Mode	1 Hz $\pm 2$ rps/mps step velocity command for velocity mode tuning. *	--	<b>X</b>
17	Position Tuning Mode	1 Hz $\pm 1/4$ rev/epitch step position command for position mode tuning. *	--	<b>X</b>

---

\* Refer to the servo tuning procedures in the *Hardware Installation Guide* for details.

---

## DMONAS Analog Monitor Output A — Scaling

Type	Outputs	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DMONAS<i>	GT6K	6.0
Units	i = scale percentage (%)	GV6K	6.0
Range	-2000 to 2000		
Default	100 (no scaling)		
Response	DMONAS: *DMONAS100		
See Also	DMONAV, DMONBS, DMONBV		

---

The `DMONAS` command sets the amount by which the variable selected with the `DMONAV` command is scaled. For example, `DMONAS200` doubles the amplitude of the selected output signal. The maximum scaled output voltage is approximately  $\pm 10V$ .

Monitor waveform clipping will occur if `DMONAS` scaling results in an output greater than  $\pm 10V$ .

## DMONAV Analog Monitor Output A — Variable

Type	Outputs	Product	Rev
Syntax	<a_><!>DMONAV<i>	GT6K	6.0
Units	i = variable number	GV6K	6.0
Range	0-24 (0 = turn off output)		
Default	0 (turn off output)		
Response	DMONAV: *DMONAV0		
See Also	DMEPIT, DMONAS, DMONBS, DMONEV		

The DMONAV command selects the function (or signal) that will be presented at the 8-bit DAC “Analog Out A” terminal (pin 21 on the DRIVE I/O connector).

Number	Name	Description	GT6K	GV6K
0	Unused / Turn off output	No output function selected	X	X
1	Motor Temperature	(TMTEMP) Estimated motor winding temperature based on a second-order thermal model (see the <i>Hardware Installation Guide</i> for details). Normalized to $\pm 10$ volts equals $\pm 250$ °C.		X
2	Drive Temperature	(TDTEMP) Measured internal drive temperature. Normalized to $\pm 10$ volts equals $\pm 250$ °C.	X	X
3	Position Error	This value is normalized as follows (value clips at $\pm 1$ rev or $\pm 1$ electrical pitch, regardless of the DMONAS setting): <ul style="list-style-type: none"> <li>• Rotary motors: <math>\pm 10V = \pm 1</math> rev (based on <math>TPER \div ERES</math>)</li> <li>• Linear motors: <math>\pm 10V = \pm 1</math> epitch (based on <math>TPER \div DMEPIT</math>)</li> </ul>		X
4	Velocity Setpoint	(TVEL) User commanded velocity. Normalized to $\pm 10$ volts equals $\pm 200$ revs/sec (rotary) or $\pm 8.4$ meters/sec (linear).	X	X
5	Estimated Actual Velocity	(TVELA) Output of velocity estimator, based on encoder or resolver data. Normalized to $\pm 10$ volts equals $\pm 200$ revs/sec (rotary) or $\pm 8.4$ meters/sec (linear).		X
6	Acceleration Setpoint	(TACC) User commanded acceleration.	X	X
7	Torque/Force Setpoint	(TTRQ) User commanded torque/force.		X
8	Actual Torque/Force	(TTRQA) Calculated torque/force based on measured motor current and motor $K_e$ (DMTKE). Scaled as a % of DMTSCL.		X
9	Velocity Error	(TVE) Normalized to $\pm 10$ volts equals $\pm 200$ revs/sec (rotary) or $\pm 8.4$ meters/sec (linear).		X
10	Phase A Commanded Current	Instantaneous commanded current for phase A (Volts per Amp). *	X	
11	Phase A Actual Current	Instantaneous measured current for phase A (Volts per Amp). *	X	X



Number	Name	Description	GT6K	GV6K
13	Phase B Actual Current	Instantaneous measured current for phase B (Volts per Amp). *	X	X
14	RESERVED			
15	RESERVED			
16	d-axis Commanded Current	Commanded direct-axis current (usually 0). This current does not produce torque/force. *		X
17	d-axis Actual Current	Estimated direct-axis current from coordinate-conversion software. Based on measured phase currents. *		X
18	q-axis Commanded Current	Commanded quadrature-axis current. This current is proportional to commanded torque/force. *		X
19	q-axis Actual Current	Estimated quadrature-axis current from coordinate-conversion software. Based on measured phase currents. This current is proportional to actual torque/force. *		X
20	Phase A Applied Voltage Duty Cycle	A voltage representation of the PWM duty cycle for Phase A.	X	
21	Phase B Applied Voltage Duty Cycle	A voltage representation of the PWM duty cycle for Phase B.	X	
22	RESERVED			
23	Position Setpoint	This value is normalized as follows (value clips at $\pm 1$ rev or $\pm 1$ electrical pitch (epitch), regardless of the DMONAS setting): <ul style="list-style-type: none"> <li>• Rotary motors: <math>\pm 10V = \pm 1</math> rev (based on <math>TPC \div ERES</math>)</li> <li>• Linear motors: <math>\pm 10V = \pm 1</math> epitch (based on <math>TPC \div DMEPIT</math>)</li> </ul>		X
24	Actual Position	This value is normalized as follows (value clips at $\pm 1$ rev or $\pm 1$ electrical pitch, regardless of the DMONAS setting): <ul style="list-style-type: none"> <li>• Rotary motors: <math>\pm 10V = \pm 1</math> rev (based on <math>TPE \div ERES</math>)</li> <li>• Linear motors: <math>\pm 10V = \pm 1</math> epitch (based on <math>TPE \div DMEPIT</math>)</li> </ul>		X

\* The nominal Volts per Amp scaling is drive dependent, and is shown below:

Stepper Drive	Nominal (V/A) Current Scaling	Servo Drive	Nominal (V/A) Current Scaling
GT6K-L5	1.432	GV6K-L3	0.615
GT6K-L8	0.882	GV6K-U3	0.615
		GV6K-U6	0.308
		GV6K-U12	0.205
		GV6K-H20	0.123
		GV6K-H40	0.062

## DMONBS Analog Monitor Output B — Scaling

Type	Outputs	Product	Rev
Syntax	<a_><!>DMONBS<i>	GT6K	6.0
Units	i = scale percentage (%)	GV6K	6.0
Range	-2000 to 2000		
Default	100 (no scaling)		
Response	DMONBS: *DMONBS100		
See Also	DMONAS, DMONAV, DMONBV		

The DMONBS command sets the amount by which the variable selected with the DMONBV command is scaled. The maximum scaled output voltage is  $\pm 10V$ .

Monitor waveform clipping will occur if DMONBS scaling results in an output greater than  $\pm 10V$ .

## DMONBV Analog Monitor Output B — Variable

Type	Outputs	Product	Rev
Syntax	<a_><!>DMONBV<i>	GT6K	6.0
Units	i = variable number	GV6K	6.0
Range	0-24 (0 = turn off output)		
Default	0 (turn off output)		
Response	DMONBV: *DMONBV0		
See Also	DMONAS, DMONAV, DMONBS		

The DMONBV command selects the function (or signal) that will be presented at the 8-bit DAC “Analog Out B” terminal (pin 22 on the DRIVE I/O connector).

Number	Name	Description	GT6K	GV6K
0	Unused / Turn off output	No output function selected	X	X
1	Motor Temperature	(TMTEMP) Estimated motor winding temperature based on a second-order thermal model (see the <i>Hardware Installation Guide</i> for details). Normalized to $\pm 10$ volts equals $\pm 250$ °C.		X
2	Drive Temperature	(TDTEMP) Measured internal drive temperature. Normalized to $\pm 10$ volts equals $\pm 250$ °C.	X	X
3	Position Error	This value is normalized as follows (value clips at $\pm 1$ rev or $\pm 1$ electrical pitch, regardless of the DMONAS setting): <ul style="list-style-type: none"> <li>• Rotary motors: <math>\pm 10V = \pm 1</math> rev (based on <math>TPER \div ERES</math>)</li> <li>• Linear motors: <math>\pm 10V = \pm 1</math> epitch (based on <math>TPER \div DMEPIT</math>)</li> </ul>		X
4	Velocity Setpoint	(TVEL) User commanded velocity. Normalized to $\pm 10$ volts equals $\pm 200$ revs/sec (rotary) or $\pm 8.4$ meters/sec (linear).	X	X
5	Estimated Actual Velocity	(TVELA) Output of velocity estimator, based on encoder or resolver data. Normalized to $\pm 10$ volts equals $\pm 200$ revs/sec (rotary) or $\pm 8.4$ meters/sec (linear).		X
6	Acceleration Setpoint	(TACC) User commanded acceleration.	X	X
7	Torque/Force Setpoint	(TTRQ) User commanded torque/force.		X
8	Actual Torque/Force	(TTRQA) Calculated torque/force based on measured motor current and motor $K_e$ (DMTKE). Scaled as a % of DMTSCL.		X
9	Velocity Error	(TVE) Normalized to $\pm 10$ volts equals $\pm 200$ revs/sec (rotary) or $\pm 8.4$ meters/sec (linear).		X
10	Phase A Commanded Current	Instantaneous commanded current for phase A (Volts per Amp). *	X	
11	Phase A Actual Current	Instantaneous measured current for phase A (Volts per Amp). *	X	X
12	Phase B Commanded Current	Instantaneous commanded current for phase B (Volts per Amp). *	X	
13	Phase B Actual Current	Instantaneous measured current for phase B (Volts per Amp). *	X	X
14	RESERVED			
15	RESERVED			
16	d-axis Commanded Current	Commanded direct-axis current (usually 0). This current does not produce torque/force. *		X
17	d-axis Actual Current	Estimated direct-axis current from coordinate-conversion software. Based on measured phase currents. *		X
18	q-axis Commanded Current	Commanded quadrature-axis current. This current is proportional to commanded torque/force. *		X
19	q-axis Actual Current	Estimated quadrature-axis current from coordinate-conversion software. Based on measured phase currents. This current is proportional to actual torque/force. *		X
20	Phase A Applied Voltage Duty Cycle	A voltage representation of the PWM duty cycle for Phase A.	X	
21	Phase B Applied Voltage Duty Cycle	A voltage representation of the PWM duty cycle for Phase B.	X	
22	RESERVED			

(Continued on next page)

Number	Name	Description	GT6K	GV6K
23	Position Setpoint	This value is normalized as follows (value clips at $\pm 1$ rev or $\pm 1$ electrical pitch (epitch), regardless of the DMONAS setting): <ul style="list-style-type: none"> <li>• Rotary motors: <math>\pm 10V = \pm 1</math> rev (based on <math>TPC \div ERES</math>)</li> <li>• Linear motors: <math>\pm 10V = \pm 1</math> epitch (based on <math>TPC \div DMEPIT</math>)</li> </ul>		X
24	Actual Position	This value is normalized as follows (value clips at $\pm 1$ rev or $\pm 1$ electrical pitch, regardless of the DMONAS setting): <ul style="list-style-type: none"> <li>• Rotary motors: <math>\pm 10V = \pm 1</math> rev (based on <math>TPE \div ERES</math>)</li> <li>• Linear motors: <math>\pm 10V = \pm 1</math> epitch (based on <math>TPE \div DMEPIT</math>)</li> </ul>		X

\* The nominal Volts per Amp scaling is drive dependent, and is shown below:

Stepper Drive	Nominal (V/A) Current Scaling	Servo Drive	Nominal (V/A) Current Scaling
GT6K-L5	1.432	GV6K-L3	0.615
GT6K-L8	0.882	GV6K-U3	0.615
		GV6K-U6	0.308
		GV6K-U12	0.205
		GV6K-H20	0.123
		GV6K-H40	0.062

## DMTAMB Motor Ambient Temperature

Type	Motor	Product	Rev
Syntax	<a_><!>DMTAMB<r>	GT6K	n/a
Units	r = Degrees Celsius	GV6K	6.0
Range	-50.0 to 250.0 : $\pm 0.1$		
Default	40.0		(applicable only to servo axes)
Response	DMTAMB: *DMTAMB40.0		
See Also	DMTMAX, DMTRWC, DMTCM, DMTCW, TASX		

The DMTAMB command sets the motor ambient temperature used by the software motor thermal model. The DMTAMB value, in conjunction with the motor thermal time constant (DMTCM), the motor winding time constant (DMTCW), the motor thermal resistance (DMTRWC) and the continuous motor current (DMTIC), is used in a real-time estimation of the motor winding temperature. When the winding temperature exceeds DMTMAX, the drive faults and TASX bit #30 is set.

**NOTE:** If you cycle power, or issue a RESET or DRESET command, the drive resets its motor thermal model (in internal software). Be aware that your motor's winding temperature may still be high.

## DMTD Motor Damping

Type	Motor	Product	Rev
Syntax	<a_><!>DMTD<r> (does not take effect until RESET, DRESET or cycle power)	GT6K	n/a
Units	Rotary motor: r = Nm/rad/sec Linear motor: r = N/meter/sec	GV6K	6.0
Range	Rotary motor: 0.000000 to 0.010000 : $\pm 0.000001$ Linear motor: DMEPIT (electrical pitch) dependent		(applicable only to servo axes)
Default	0.000000		
Response	DMTD: *DMTD0.002000		
See Also	DMTR, LDAMP		

**NOTE:** This command does not take effect until you cycle power to the drive, or issue a RESET or DRESET command.

**AUTO-SETUP:** This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6). If you did not use the configuration utility or are not using a Parker Motor, this command is set to zero and you will have to manually set this parameter. (Refer to DMTR for a list of auto-configured commands.)

The DMTD command specifies the damping of the motor itself. This includes both magnetic losses and bearing losses. (The load damping is specified with the LDAMP command.)

---

## DMTIC                      Continuous Current

Type	Motor	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DMTIC<r>	GT6K	6.0
Units	r = Amps-RMS	GV6K	6.0
Range	0.00 to 100.00 : ±0.01		
Default	0.00 (DMTIC of 0 results in motor configuration warning)		
Response	DMTIC:     *DMTIC6.50		
See Also	DMTICD, TASX, TCS, TDICNT		

---

**GV6K Only:** This command does not take effect until you cycle power to the drive, or issue a RESET or DRESET command.

**GT6K Only:** This command takes effect immediately.

**AUTO-SETUP:** This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6). If you did not use the configuration utility or are not using a Parker Motor, this command is set to zero and you will have to manually set this parameter. (Refer to DMTR for a list of auto-configured commands.) If the drive is powered up when this command is set to zero (for instance, if RFS is executed), the drive reports a motor configuration warning with TASX bit 28, and writes 40 to TCS.

The DMTIC command sets the continuous operating current for a motor. For a servo drive operating a rotary motor continuously at this current, the internal winding temperature will reach 125°C with a specified heatsink (see the *Gemini Motor Reference Manual* for heatsink dimensions) in a 40°C ambient. For linear servo motors, the winding will reach 90°C in a 25°C ambient.

The continuous current of a motor that is designed to provide a long service life depends on many factors. Among them are motor ambient temperature, the degree of heatsinking provided by the motor mounting surface, and airflow over the motor. In a stepper, the continuous current is flowing in the motor continuously. In a servo, the continuous current is used in calculations to protect the motor from thermal damage, and has no immediate effect on performance.

**Example:**

DMTIC5                      ; Set the motor current to 5 amps rms (equates to 7.07 amps peak)

---

## DMTICD                      Continuous Current Derating

Type	Motor	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DMTICD<i> <i>(does not take effect until RESET, DRESET or cycle power)</i>	GT6K	n/a
Units	i = percent derating at rated speed	GV6K	6.0
Range	0.00 to 100.00 : ±0.01		
Default	0.00 (DMTICD of 0 results in no current derating)		(applicable only to servo axes)
Response	DMTICD:     *DMTICD5		
See Also	DMTIC, DMTIP, DMTW		

---

**NOTE:** This command does not take effect until you cycle power to the drive, or issue a RESET or DRESET command.

**AUTO-SETUP:** This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6). If you did not use the configuration utility or are not using a Parker Motor, this command is set to zero and you will have to manually set this parameter. (Refer to DMTR for a list of auto-configured commands.)

The DMTICD command sets the percentage current derating at rated speed (DMTW). This value sets the extent to which continuous current must be reduced at speed to compensate velocity-related losses in the motor.

For example, DMTICD3 sets the motor's continuous current derating to 3% (or 97% of continuous value DMTIC) at the motor's rated speed (DMTW). At half this speed, it will be reduced 1.5%.

---

<b>DMTIND</b>		<b>Motor Inductance</b>	
Type	Motor	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DMTIND<r> (does not take effect until RESET, DRESET or cycle power)	GT6K	6.0
Units	r = mH	GV6K	n/a
Range	0.0 to 200.0 : ±0.1		
Default	0.0 (DMTIND of 0 results in motor config. error)		(applicable only to stepper axes)
Response	DMTIND *DMTIND10		
See Also	DMTLMN, DMMLMX, DMTR, TASX, TCS		

---

**NOTE:** This command does not take effect until you cycle power to the drive, or issue a RESET or DRESET command.

**AUTO-SETUP:** This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6). If you did not use the configuration utility or are not using a Parker Motor, this command is set to zero and you will have to manually set this parameter. (Refer to DMTR for a list of auto-configured commands.) If the drive is powered up when this command is set to zero (for instance, if RFS is executed), the drive reports a motor configuration error with TASX bit 7, writes a value of -32726 to the TCS register, and shuts down the drive (DRIVE0).

The DMTIND command sets the motor phase inductance for stepper motors only (servo motor inductance is set with DMTLMN and DMMLMX). The motor inductance entered is the motor phase inductance you measure across one phase at the motor terminals of the drive. The inductance value is the "small signal inductance" as measured by a hand-held or bench-top inductance meter at 1 KHz.

A procedure for configuring motor inductance (for non-Parker motors) is provided in the *Configuration* chapter of the *Hardware Installation Guide*.

---

<b>DMTIP</b>		<b>Peak Current</b>	
Type	Motor	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DMTIP<r>	GT6K	n/a
Units	r = Amps-RMS	GV6K	6.0
Range	0.00 to 128.00 : ±0.01		
Default	0.00 (DMTIP of 0 results in motor config. warning)		(applicable only to servo axes)
Response	DMTIP: *DMTIP7.50		
See Also	DMTIC, DMTICD, DMTLIM, DMTR, TASX, TCS, TDIMAX		

---

**AUTO-SETUP:** This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6). If you did not use the configuration utility or are not using a Parker Motor, this command is set to zero and you will have to manually set this parameter. (Refer to DMTR for a list of auto-configured commands.) If the drive is powered up when this command is set to zero (for instance, if RFS is executed), the drive reports a motor configuration warning with TASX bit 28, and a value of 51 is written to the TCS configuration status register.

The DMTIP command sets a limit that the commanded current cannot exceed. This is typically set to three times the motor's continuous current rating (DMTIC) or less.

If DMTIP is set higher than the full-scale value calculated by DMTLIM (torque/force limit) the new DMTIP value will be ignored (but not overwritten), the configuration warning (TASX bit #28) will be set, a value of 51 is written to the TCS configuration status register, and the full-scale DMTLIM value will be used internally. The configuration warning is cleared with the RESET or DRESET command or by cycling power to the drive.

If the DMTIP value exceeds the drive's maximum output current (TDIMAX), the DMTIP value will be ignored and the maximum allowable value will be used (see table below).

	Units	GV6K-L3	GV6K-U3	GV6K-U6	GV6K-U12	GV6K-H20	GV6K-H40
Maximum Current Rating	amps *	7.5	7.5	15	30	50	100

\* peak of the sinewave

Note that the values in TDIMAX are amps (peak of the sine wave) and the value for DMTIP is in amps (rms). They are related by:

$$I_{rms} = \frac{I_{peak\ of\ the\ sine\ wave}}{\sqrt{2}}$$

## DMTJ Motor Rotor Inertia / Forcer Mass

Type	Motor	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DMTJ<r> (does not take effect until RESET, DRESET or cycle power)	GT6K	6.0
Units	Rotary motor: r = kgm <sup>2</sup> * 10 <sup>-6</sup> Linear motor: r = kg	GV6K	6.0
Range	Rotary motor: 0.000 to 1000000.000 : ±0.001 Linear motor: DMEPIT (electrical pitch) dependent		
Default	0.000 (DMTJ of 0 results in motor config. error)		
Response	DMTJ: *DMTJ200.600		
See Also	DMTR, TASX, TCS		

**NOTE:** This command does not take effect until you cycle power to the drive, or issue a RESET or DRESET command.

**AUTO-SETUP:** This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6). If you did not use the configuration utility or are not using a Parker Motor, this command is set to zero and you will have to manually set this parameter. (Refer to DMTR for a list of auto-configured commands.) If the drive is powered up when this command is set to zero (for instance, if RFS is executed), the drive reports a motor configuration error with TASX bit 7, writes a value of -32710 to the TCS register, and shuts down the drive (DRIVE0).

The DMTJ command sets the motor rotor inertia for rotary motors, or the forcer mass for linear motors.

## DMTKE Motor Ke

Type	Motor	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DMTKE<r> (does not take effect until RESET, DRESET or cycle power)	GT6K	n/a
Units	Rotary motor: r = volts (0-peak)/krpm (measured line-to-line) Linear motor: r = volts (0-peak)/meter/sec (measured line-to-line)	GV6K	6.0
Range	Rotary motor: 0.0 to 400.0 : ±0.1 Linear motor: DMEPIT (electrical pitch) dependent		(applicable only to servo axes)
Default	0.0 (DMTKE of 0 results in motor config. error)		
Response	DMTKE: *DMTKE15.0		
See Also	DMEPIT, DMONAV, DMONBV, DMTSCL, DMTR, TASX, TCS		

**NOTE:** This command does not take effect until you cycle power to the drive, or issue a RESET or DRESET command.

**AUTO-SETUP:** This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6). If you did not use the configuration utility or are not using a Parker Motor, this command is set to zero and you will have to manually set this parameter. (Refer to DMTR for a list of auto-configured commands.) If the drive is powered up when this command is set to zero (for instance, if RFS is executed), the drive reports a motor configuration error with TASX bit 7, writes a value of -32727 to the TCS register, and shuts down the drive (DRIVE0).

The DMTKE command specifies the motor voltage constant (Ke). This defaults to the nominal Ke of the motor selected with the DMTR command.

The motor's torque/force constant (Kt) is derived from the motor's voltage constant (Ke) by the following relationship:

**Rotary motors:** 
$$Kt(Nm / A^*) = \frac{3\sqrt{3}}{200\pi} * Ke(Volts^* / krpm)$$

\*  
peak value

**Linear motors:** 
$$Kt(N / A^*) = \frac{3\sqrt{3}}{200\pi} * Ke(Volts^* / (meter / sec))$$

\*  
peak value

**NOTE:** The Gem6K requires values in rotary units. Motion Planner does these conversions for you when you use the Setup Wizard. If you do not use the Setup Wizard, you must convert your linear units to rotary units. See the DMEPIT command for information on conversion.

---

## DMTLIM      Torque/Force Limit

Type	System	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DMTLIM<r>	GT6K	n/a
Units	Rotary motor: r = Nm Linear motor: r = N	GV6K	6.0
Range	Rotary motor: 0.0 to 500.0 (motor/drive dependent) : ±0.1 Linear motor: DMEPIT (electrical pitch) dependent	(applicable only to servo axes)	
Default	0		
Response	DMTLIM: *DMTLIM10.5		
See Also	DCLRLR, DMEPIT, DMODE, DMTIP, DMTKE, DMTR, DMTSCL, TASX, TGAIn, TSGSET, TTRQ, TTRQA		

---

**AUTO-SETUP:** This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6). If you did not use the configuration utility or are not using a Parker Motor, this command is set to zero and you will have to manually set this parameter. Refer to DMTR for a list of auto-configured commands.

The DMTLIM command sets a maximum torque/force limit for the system. Requests for higher torque/force will be clamped to this value. This command will default automatically to a value appropriate to the motor selection (DMTR) and the GV6K drive you are using, and no changes are required in many cases.

If your mechanical system has torque/force limitations (due, for example, to the limitations of a coupler or belt), you can use this command to limit system torque/force without affecting system scaling or gains.

During initial tuning, this command can be used to limit the torque/force produced if the system becomes unstable. This reduces the rate of motor heating, allows more reaction time for the person tuning the system, and lessens chances of damage to the mechanical system.

When the GV6K's commanded torque/force reaches the limit set by DMTLIM (TTRQ = DMTLIM), TASX bit #31 is set. TASX bit #31 remains set until you clear it with the DCLRLR command, cycle power, or issue a RESET or DRESET. This is not considered a fault condition.

If DMTLIM is set higher than the value allowed by the motor's peak current times the motor's Kt, or the drive's peak current times the motor's Kt (whichever is lower), the new DMTLIM value will be ignored (but

not overwritten), the status warning bit #28 in TASX will be set, and the maximum internal value will be used. This warning is cleared with the RESET or DRESET command or by cycling power to the drive.

The motor's torque/force constant (Kt) is derived from the motor's voltage constant (Ke, which is set by the DMTKE command) by the following relationship (note: Ke is set with the DMTKE command):

$$\text{Rotary motors: } Kt(Nm / A^*) = \frac{3\sqrt{3}}{200\pi} * Ke(Volts^* / krpm)$$

\*  
peak value

$$\text{Linear motors: } Kt(N / A^*) = \frac{3\sqrt{3}}{200\pi} * Ke(Volts^* / (meter / sec))$$

\*  
peak value

NOTE: The

Gem6K requires values in rotary units. Motion Planner does these conversions for you when you use the Setup Wizard. If you do not use the Setup Wizard, you must convert your linear units to rotary units. See the DMEPIT command for information on conversion.

### Working with servo gains.

- Servo tuning process: refer to the *Hardware Installation Guide*.
- Check the values of all active gains (DMTLIM is one of many servo gains): use TGAIN.
- Creating and invoking gain sets: see SGSET, SGENB, TGAIN, TSGSET.

---

## DMTLMN Minimum Motor Inductance

Type	Motor	Product	Rev
Syntax	<a_><!--DMTLMN<r> (does not take effect until RESET, DRESET or cycle power)	GT6K	n/a
Units	r = mH (measured line-to-line)	GV6K	6.0
Range	0.1 to 200.0 (motor dependent) : ±0.1		(applicable only to servo axes)
Default	0.0 (DMTLMN of 0 results in motor config. error)		
Response	DMTLMN: *DMTLMN10.0		
See Also	DMTLMX, DMTR, TASX, TCS		

---

**NOTE:** This command does not take effect until you cycle power to the drive, or issue a RESET or DRESET command.

**AUTO-SETUP:** This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6). If you did not use the configuration utility or are not using a Parker Motor, this command is set to zero and you will have to manually set this parameter. (Refer to DMTR for a list of auto-configured commands.) If the drive is powered up when this command is set to zero (for instance, if RFS is executed), the drive reports a motor configuration error with TASX bit 7, writes a value of -32715 to the TCS register, and shuts down the drive (DRIVE0).

The DMTLMN command specifies the minimum value of motor inductance. This will usually differ from the nominal nameplate value because actual inductance is usually position dependent.

If DMTLMN value is less than the “Lmin” value (see table below), the drive will fault (motor configuration error is reported in TASX bit #7).

	Units	GV6K-L3	GV6K-U3	GV6K-U6	GV6K-U12	GV6K-H20	GV6K-H40
Lmin (min. phase-phase inductance)	mH	0.5	5.0	5.0	5.0	5.0	5.0



---

## DMTLMX      Maximum Motor Inductance

Type	Motor	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DMTLMX<r>    (does not take effect until RESET, DRESET or cycle power)	GT6K	n/a
Units	r = mH (measured line-to-line)	GV6K	6.0
Range	0.1 to 200.0 (motor dependent) : ±0.1		
Default	0.0 (DMTLMX of 0 results in motor config. error)		(applicable only to servo axes)
Response	DMTLMX:    *DMTLMX10.0		
See Also	DMTLMN, DMTR, TASX, TCS		

---

**NOTE:** This command does not take effect until you cycle power to the drive, or issue a RESET or DRESET command.

**AUTO-SETUP:** This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6) . If you did not use the configuration utility or are not using a Parker Motor, this command is set to zero and you will have to manually set this parameter. (Refer to DMTR for a list of auto-configured commands.) If the drive is powered up when this command is set to zero (for instance, if RFS is executed), the drive reports a motor configuration error with TASX bit 7, writes a value of -32714 to the TCS register, and shuts down the drive (DRIVE0).

The DMTLMX command specifies the maximum value of motor inductance. This will usually differ from the nominal nameplate value since actual inductance is usually position dependent.

---

## DMTMAX      Maximum Motor Winding Temperature

Type	Motor	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DMTMAX<r>	GT6K	n/a
Units	r = Degrees Celsius	GV6K	6.0
Range	-50.0 to 250.0 : ±0.1		
Default	125.0		(applicable only to servo axes)
Response	DMTMAX:    *DMTMAX125.0		
See Also	DMTAMB, DMTIC, DMTRWC, DMTTTCM, DMTTTCW, TASX		

---

**AUTO-SETUP:** This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6). If you did not use the configuration utility or are not using a Parker Motor, you will have to manually set this parameter. (Refer to DMTR for a list of auto-configured commands.)

The DMTMAX command sets the maximum motor winding temperature allowed. The DMTMAX value, in conjunction with the motor thermal time constant (DMTTTCM), the motor winding time constant (DMTTTCW), the motor thermal resistance (DMTRWC) and the continuous motor current (DMTIC), is used in a real-time estimation of the motor winding temperature. When the winding temperature exceeds DMTMAX, the drive faults and TASX bit #30 is set.

**NOTE:** If you cycle power, or issue a RESET or DRESET command, the drive resets its motor thermal model (in internal software). Be aware that your motor's winding temperature may still be high.

## DMTR Identify (and Load) Motor

Type	Drive Configuration	Product	Rev
Syntax	<a_><!>DMTR<i>	GT6K	6.0
Units	i = Parker motor identification number	GV6K	6.0
Range	0-3000 (0 = non-Parker motor; 1-3000 = Parker motor)		
Default	-1 (motor config. Parameters are not configured)		
Response	DMTR: *DMTR50		
See Also	(see parameter table below), DRIVE, RESET, RFS, TASX, TCS		

The purpose of the DMTR command is to record and report the identification number of the Parker motor you selected in the setup tool in Motion Planner (see page 6).

When you select a specific Parker motor using the Motion Planner setup tool, the DMTR setting and various motor parameters (see list below) are automatically configured for the associated motor and saved in a configuration file. After you download the configuration file to the Gem6K drive, you must cycle drive power, or issue a RESET or DRESET command for the DMTR and all the motor parameter commands to take effect. (**NOTE:** If you do not select a Parker motor, the default setting, DMTR-1, is used and you must set all relevant motor parameters manually.) Avoid using the DMTR command to change the motor number, because the new DMTR value may not represent the actual motor parameters that are currently loaded in the drive. Instead, use the Motion Planner setup tool to change the motor number.

Stepper Motor Data Parameters	Servo Motor Data Parameters	
DIGNA .....Current Loop Gain A	DIBW .....Current Loop Bandwidth	DMTSC ..... Torque/Force Scaling
DIGNB .....Current Loop Gain B	DMEPIT ..... Motor Electrical Pitch	DMTTCM ..... Motor Thermal Time Constant
DIGNC .....Current Loop Gain C	DMTD ..... Motor Damping	DMTTCW ..... Motor Winding Time Constant
DIGND .....Current Loop Gain D	DMTIC ..... Continuous Current	DMTW ..... Motor Rated Speed
DMTIC .....Continuous Current	DMTICD ..... Continuous Current Derating	DMVLIM ..... Velocity Limit
DMTIND .....Motor Inductance	DMTIP ..... Peak Current	DPBW ..... Position Loop Bandwidth
DMTJ .....Motor Rotor Inertia	DMTJ ..... Motor Rotor Inertia	DPOLE ..... Number of Motor Pole Pairs
DMTRES .....Motor Winding Resistance	DMTKE ..... Motor Ke	DRES ..... Drive Resolution
DMTST .....Motor Static Torque	DMTLIM ..... Torque/Force Limit	ERES ..... Encoder Resolution
DMVLIM .....Velocity Limit	DMTLMN ..... Minimum Motor Inductance	ORES ..... Encoder Output Resolution
DPOLE .....Number of Motor Pole Pairs	DMTLMX ..... Maximum Motor Inductance	SFB ..... Feedback Source Selection
	DMTMAX ..... Maximum Motor Winding Temp.	SRSET ..... Resolver Offset Angle
	DMTRES ..... Motor Winding Resistance	
	DMTRWC ..... Motor Winding Thermal Resistance	

Although these command values are auto-configured when you select a Parker motor (using the setup tool in Motion Planner), you may individually set the command values with the respective configuration command.

### Motor Configuration Error

For many of the above motor parameters, if they are not configured (i.e., a command is left at its factory default value, or an RFS command is executed) when the Gem6K drive is powered up, a motor configuration “error” or “warning” is reported in TASX bit #7 or bit #28 (an “error” also disables the drive – DRIVE0). To resolve the error or warning condition, you must select a Parker motor with Motion Planner (or configure each motor parameter command with a value other than zero – using a Gem6K terminal emulator), download the resulting configuration information and then cycle power, or issue the RESET or DRESET command.

**Updating the motor data table.** The motor information for the parameters list above is located in a file named “GEM\_motors.mtr” and is used by Motion Planner to configure the Gem6K drive. Updates to this file are maintained on the Compumotor web site (<http://www.compumotor.com>) — search for “GEM\_motors.mtr”. If you need to update the information, download the latest motor table file. If you are using Motion Planner, place the updated file in the Motion Planner directory (default location is \Program Files\Compumotor\Motion Planner).

---

## DMTRES Motor Winding Resistance

Type	Motor	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!--DMTRES<r> (does not take effect until RESET, DRESET or cycle power)	GT6K	6.0
Units	r = Ohm (measured line-to-line)	GV6K	6.0
Range	0.00 to 50.00 : ±0.01		
Default	0.00 (DMTRES of 0 results in motor config. error)		
Response	DMTRES: *DMTRES7.50		
See Also	DMTR, TASX, TCS		

---

**NOTE:** This command does not take effect until you cycle power to the drive, or issue a RESET or DRESET command.

**AUTO-SETUP:** This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6). If you did not use the configuration utility or are not using a Parker Motor, this command is set to zero and you will have to manually set this parameter. (Refer to DMTR for a list of auto-configured commands.) If the drive is powered up when this command is set to zero (for instance, if RFS is executed), the drive reports a motor configuration error with TASX bit 7, writes a value of -32725 to the TCS register, and shuts down the drive (DRIVE0).

The DMTRES command sets the motor winding resistance. This resistance value is measured at 25 °C at the drive end of the motor cable (motor cable included). For steppers, this resistance is a phase resistance measured at terminals A+ to A-, or B+ to B-. For servos, this resistance is the phase-to-phase resistance measured at terminals U to V, V to W, or W to U.

**NOTE:** Disconnect the motor cable from the drive before attempting to make this measurement. For best accuracy, and to avoid injury, this measurement must be made with the motor cable disconnected from the drive.

---

## DMTRWC Motor Winding Thermal Resistance

Type	Motor	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!--DMTRWC<r> (does not take effect until RESET, DRESET or cycle power)	GT6K	n/a
Units	r = Degrees Celsius/Watt (°C/W)	GV6K	6.0
Range	0.00 to 50.00 : ±0.01		(applicable only to servo axes)
Default	0.00		
Response	DMTRWC: *DMTRWC23.60		
See Also	DMTR, DMTCM, DMTCW, TASX, TCS		

---

**NOTE:** This command does not take effect until you cycle power to the drive, or issue a RESET or DRESET command.

**AUTO-SETUP:** This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6). If you did not use the configuration utility or are not using a Parker Motor, this command is set to zero and you will have to manually set this parameter. (Refer to DMTR for a list of auto-configured commands.)

DMTRWC specifies the temperature rise of the motor winding above motor case temperature per watt of winding power dissipation. Motor heatsinking does not affect this value. See the DMTMAX command for more information on how the drive estimates motor temperature.

---

## DMTSCL Torque/Force Scaling

Type	Drive Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DMTSCL<r>	GT6K	n/a
Units	Rotary motor: r = Nm Linear motor: r = N	GV6K	6.0
Range	Rotary motor: 0.0 to 500.0 (motor/drive dependent) : ±0.1 Linear motor: DMEPIT (electrical pitch) dependent		(applicable only to servo axes)
Default	0		
Response	DMTSCL: *DMTSCL20.0		
See Also	DMEPIT, DMONAV, DMONBV, DMTKE, DMTLIM, TTRQ, TTRQA		

---

**AUTO-SETUP:** This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6). If you did not use the configuration utility or are not using a Parker Motor, this command is set to zero and you will have to manually set this parameter. (Refer to DMTR for a list of auto-configured commands.)

The DMTSCL command scales the torque/force (TTRQ and TTRQA). Motion Planner’s configuration utility will set the DMTSCL value according to the Parker motor you select. If you use a non-Parker motor, set the full-scale DMTSCL value according to the peak torque/force your motor can produce (in newton-meters for rotary motors, or newtons for linear motors).

DMTLIM may limit torque/force to less than this full-scale value.

**NOTE:** The Gem6K requires values in rotary units. Motion Planner does these conversions for you when you use the Setup Wizard. If you do not use the Setup Wizard, you must convert your linear units to rotary units. See the DMEPIT command for information on conversion.

---

## DMTSTT Motor Static Torque

Type	Drive Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DMTSTT<i> (does not take effect until RESET, DRESET or cycle power)	GT6K	6.0
Units	i = oz-in	GV6K	n/a
Range	0 to 5000		(applicable only to stepper axes)
Default	0 (DMTSTT of 0 results in motor config. error)		
Response	DMTSTT *DMTSTT410		
See Also	DMTR, TASX, TCS		

---

**NOTE:** This command does not take effect until you cycle power to the drive, or issue a RESET or DRESET command.

**AUTO-SETUP:** This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6). If you did not use the configuration utility or are not using a Parker Motor, this command is set to zero and you will have to manually set this parameter. (Refer to DMTR for a list of auto-configured commands.) If the drive is powered up when this command is set to zero (for instance, if RFS is executed), the drive reports a motor configuration error with TASX bit 7, writes a value of -32729 to the TCS register, and shuts down the drive (DRIVE0).

The DMTSTT command sets the motor static torque. The motor static torque is the motor holding torque measured in the “one phase on” condition. The “one phase on” condition is measured with the peak motor current in one phase.

---

## DMTTCM Motor Thermal Time Constant

Type	Motor	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DMTTCM<r> (does not take effect until RESET, DRESET or cycle power)	GT6K	n/a
Units	r = minutes	GV6K	6.0
Range	0.0 to 300.0 : ±0.1		(applicable only to servo axes)
Default	0.0		
Response	DMTTCM: *DMTTCM30.4		
See Also	DMTR, DMTCW		

---

**NOTE:** This command does not take effect until you cycle power to the drive, or issue a RESET or DRESET command.

**AUTO-SETUP:** This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6). If you did not use the configuration utility or are not using a Parker Motor, this command is set to zero and you will have to manually set this parameter. (Refer to DMTR for a list of auto-configured commands.)

The DMTTCM command specifies the thermal time constant of the motor and its mounting. This value is used to help protect the motor from thermal damage. It describes the length of time the motor takes to reach 63% of it's final temperature, given constant power. Note that motor mounting will affect this.

Continuous current ratings and published time constants for Parker motors are specified when mounted to a 10" x 10" x 1/4" aluminum plate in 25 °C free air. If your mounting surface provides heatsinking or thermal mass significantly different than this, a different value may be appropriate to your application. Note also that the time constant of the motor winding itself (DMTTCW) is much faster than this; therefore, the rise in winding temperature will initially be much faster than DMTTCM would suggest.

---

## DMTTCW Motor Winding Time Constant

Type	Motor	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DMTTCW<r> (does not take effect until RESET, DRESET or cycle power)	GT6K	n/a
Units	r = minutes	GV6K	6.0
Range	0.00 to 100.00 : ±0.01		(applicable only to servo axes)
Default	0.00		
Response	DMTTCW: *DMTTCW28.40		
See Also	DMTMAX, DMTR, DMTTCM, DMTRWC		

---

**NOTE:** This command does not take effect until you cycle power to the drive, or issue a RESET or DRESET command.

**AUTO-SETUP:** This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6). If you did not use the configuration utility or are not using a Parker Motor, this command is set to zero and you will have to manually set this parameter. (Refer to DMTR for a list of auto-configured commands.)

The DMTTCW command specifies the time constant of the motor winding alone. This is the time for the winding to reach 63% of it's final temperature rise above the rest of the motor, given constant power. Note that this is NOT the time constant usually specified in motor data sheets (see DMTTCM); the DMTTCW value is typically much faster.

---

## DMTW Motor Rated Speed

Type	Motor	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DMTW<r> (does not take effect until RESET, DRESET or cycle power)	GT6K	n/a
Units	Rotary motor: r = revs/sec Linear motor: r = meters/sec	GV6K	6.0
Range	Rotary motor: 0.0 to 200.0 : ±0.1 Linear motor: DMEPIT (electrical pitch) dependent	(applicable only to servo axes)	
Default	0.0 (DMTW of 0 results in motor config. error)		
Response	DMTW: *DMTW150.0		
See Also	DMEPIT, DMTICD, DMTR, TASX, TCS		

---

**NOTE:** This command does not take effect until you cycle power to the drive, or issue a RESET or DRESET command.

**AUTO-SETUP:** This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6). If you did not use the configuration utility or are not using a Parker Motor, this command is set to zero and you will have to manually set this parameter. (Refer to DMTR for a list of auto-configured commands.) If the drive is powered up when this command is set to zero (for instance, if RFS is executed), the drive reports a motor configuration error with TASX bit 7, writes a value of -32718 to the TCS register, and shuts down the drive (DRIVE0).

The DMTW command specifies the rated speed of the motor. This is the lesser of:

- (rotary motor) The motor mechanical limit of 7500 RPM (125 rps)
- (rotary motor) The encoder limit of 6000 rpm (100 rps) for a 1000-line encoder
- Linear motor speed limitations include encoder resolution and track length.
- The corner of the continuous speed/torque or speed/force curve (the point where the continuous torque/force breaks downward).

The DMTW value is used in conjunction with DMTICD to protect the motor from thermal damage.

**NOTE:** The Gem6K requires values in rotary units. Motion Planner does these conversions for you when you use the Setup Wizard. If you do not use the Setup Wizard, you must convert your linear units to rotary units. See the DMEPIT command for information on conversion.

---

## DMVLIM Velocity Limit

Type	System	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DMVLIM<r>	GT6K	6.0
Units	Rotary motor: r = revs/sec Linear motor: r = meters/sec	GV6K	6.0
Range	Rotary motor: GT6K: 0.000000 to 60.000000 : ±0.000001 GV6K: 0.000000 to 200.000000 : ±0.000001 Linear motor: DMEPIT (electrical pitch) dependent		
Default	GT6K: 50.000000 GV6K: 200.000000		
Response	DMVLIM: *DMVLIM50.000000		
See Also	DCLRLR, DMODE, DMTR, SGVRAT, TASX, TGAIn, TSGSET		

---

**AUTO-SETUP:** This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6). If you did not use the configuration utility or are not using a Parker Motor, this command is set to zero and you will have to manually set this parameter. Refer to DMTR for a list of auto-configured commands.

The DMVLIM command sets a limit that commanded velocity cannot exceed, without affecting gains or scaling. This is typically used to protect parts of the mechanical system.

In controller/drive mode (DMODE12), DMVLIM clamps the command to the specified value.

If the velocity demand from the internal Gem6K control loops exceeds the limits set by DMVLIM, then TASX bit 18 will be set. The Gem6k responds to this condition by invoking the “Override Mode,” in which the

drive software clamps the maximum allowable velocity command to the value set by DMVLIM. Bit 18 stays latched until a DCLRLR, RESET or DRESET is issued.

Changes to the DMVLIM command are not allowed while motion is in progress.

**NOTE:** The Gem6K requires values in rotary units. Motion Planner does these conversions for you when you use the Setup Wizard. If you do not use the Setup Wizard, you must convert your linear units to rotary units. See the DMEPIT command for information on conversion.

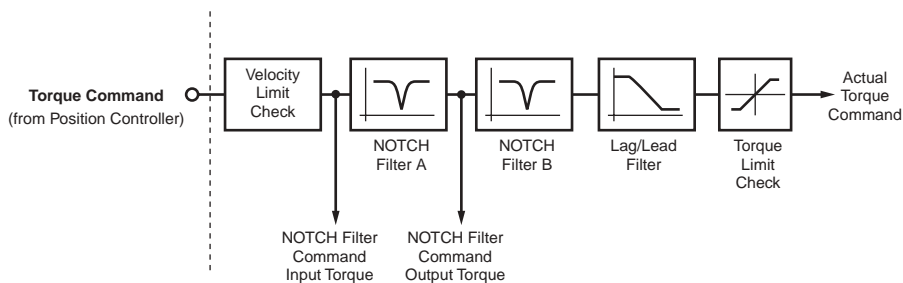
**Working with servo gains.**

- Servo tuning process: refer to the *Hardware Installation Guide*.
- Check the values of all active gains (DMVLIM is one of many servo gains): use TGAIN.
- Creating and invoking gain sets: see SGSET, SGENB, TGAIN, TSGSET.

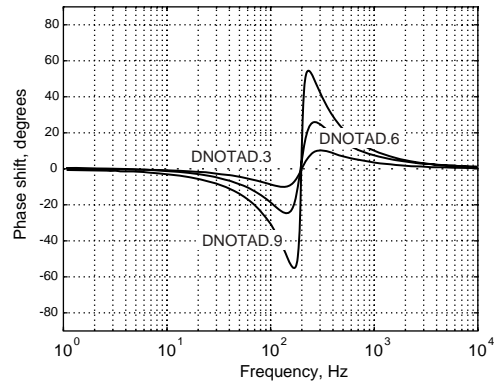
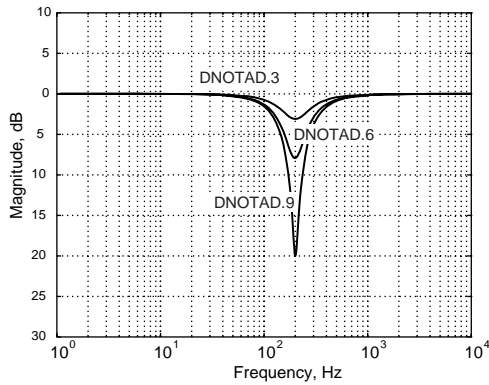
<b>DNOTAD</b>		<b>Notch Filter A Depth</b>	
Type	Tuning	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DNOTAD<i>	GT6K	n/a
Units	n/a	GV6K	6.0
Range	0.0000 - 1.0000		
Default	0.0000 (depth is zero)		(applicable only to servo axes)
Response	DNOTAD: *DNOTAD.5		
See Also	DMODE, DNOTAF, DNOTAQ, DNOTBD, DNOTBF, DNOTBQ, DNOTLD, DNOTLG, DPBW, TGAIN, TSGSET		

The DNOTAD command sets the depth for the commanded torque/force notch filter A. Setting this to 0 disables the filter. This command is useful in adjusting the maximum allowable attenuation and phase shift through the filter. The deeper the notch depth, the more attenuation and phase shift. In general, the notch depth is increased until the resonance is diminished. Increasing the depth further might increase the phase shift to an unacceptable level and decrease the overall system performance.

There are two cascaded notch filters labeled “A” and “B”. Both filters operate in exactly the same way. The diagram below shows the topology of these filters.



The graphs below illustrate the transfer function for the magnitude and phase of the notch filter command output torque/force vs. the notch filter command input torque/force. In this example, the notch depths are set to .3, .6, and .9 (DNOTAD . 3, DNOTAD . 6, DNOTAD . 9). The notch center frequency is set to 200 Hz (DNOTAF200) and the “Q” is set to 1 (DNOTAQ1).



These filters operate in all DMODE settings, except Autorun (DMODE13) and Torque/Force Tuning mode (DMODE15).

### Working with servo gains.

- Servo tuning process: refer to the *Hardware Installation Guide*.
- Check the values of all active gains (DNOTAD is one of many servo gains): use TGAIN.
- Creating and invoking gain sets: see SGSET, SGENB, TGAIN, TSGSET.

---

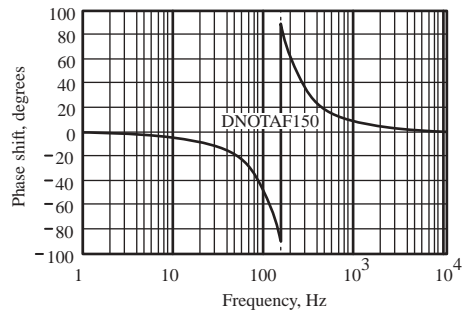
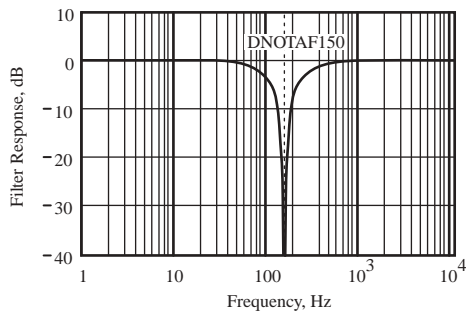
## DNOTAF Notch Filter A Frequency

Type	Tuning	Product	Rev
Syntax	<a_><!>DNOTAF<i>	GT6K	n/a
Units	i = Hz	GV6K	6.0
Range	0 (disable), or 60-1000		
Default	0 (filter is disabled)		(applicable only to servo axes)
Response	DNOTAF: *DNOTAF200		
See Also	DNOTAD, DNOTAQ, DNOTBD, DNOTBF, DNOTBQ, DNOTLD, DNOTLG, TASX, TCS, TGAIN, TSGSET		

---

The DNOTAF command sets the center frequency for the commanded torque/force notch filter A. Setting this to 0 disables the filter. If setting a value results in an internal calculation error, the last valid value is used, TASX bit #28 is set, and a value of 551 is written to the TCS register.

There are two cascaded notch filters labeled “A” and “B”. Both filters operate in exactly the same way. The graphs below illustrate the transfer function (magnitude and phase) of the input commanded torque/force vs. the output commanded torque/force. In this example, the notch frequency is set to 150 Hz (DNOTAF150) and the “Q” is set to 1 (DNOTAQ1).



These filters operate in all DMODE settings, except Autorun (DMODE13) and Torque/Force Tuning mode (DMODE15).

### Working with servo gains.

- Servo tuning process: refer to the *Hardware Installation Guide*.
- Check the values of all active gains (DNOTAF is one of many servo gains): use TGAIN.
- Creating and invoking gain sets: see SGSET, SGENB, TGAIN, TSGSET.



---

## **DNOTAQ**      **Notch Filter A Quality Factor**

Type	Tuning	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DNOTAQ<r>	GT6K	n/a
Units	r = quality factor	GV6K	6.0
Range	0.5 to 2.5		
Default	1		
Response	DNOTAQ: *DNOTAQ1.5		(applicable only to servo axes)
See Also	DNOTAD, DNOTAF, DNOTBD, DNOTBF, DNOTBQ, DNOTLD, DNOTLG, TGAIN, TSGSET		

---

The DNOTAQ command sets the quality factor (Q) for notch filter A. For a description of the filter's transfer function characteristics, refer to the DNOTAF command description.

### **Working with servo gains.**

- Servo tuning process: refer to the *Hardware Installation Guide*.
- Check the values of all active gains (DNOTAQ is one of many servo gains): use TGAIN.
- Creating and invoking gain sets: see SGSET, SGENB, TGAIN, TSGSET.

---

## **DNOTBD**      **Notch Filter B Depth**

Type	Tuning	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DNOTBD<i>	GT6K	n/a
Units	n/a	GV6K	6.0
Range	0.0000 - 1.0000		
Default	0.0000 (depth is zero)		
Response	DNOTBD: *DNOTBD.5		(applicable only to servo axes)
See Also	DNOTAD, DNOTAF, DNOTAQ, DNOTBF, DNOTBQ, DNOTLD, DNOTLG, TGAIN, TSGSET		

---

The DNOTBD command sets the depth for the commanded torque/force notch filter B. Refer to DNOTAD for a complete description of the notch filter depth.

### **Working with servo gains.**

- Servo tuning process: refer to the *Hardware Installation Guide*.
- Check the values of all active gains (DNOTBD is one of many servo gains): use TGAIN.
- Creating and invoking gain sets: see SGSET, SGENB, TGAIN, TSGSET.

---

## **DNOTBF**      **Notch Filter B Frequency**

Type	Tuning	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DNOTBF<i>	GT6K	n/a
Units	i = Hz	GV6K	6.0
Range	0 (disable), or 60-1000		
Default	0 (filter is disabled)		
Response	DNOTBF: *DNOTBF200		(applicable only to servo axes)
See Also	DNOTAD, DNOTAF, DNOTAQ, DNOTBD, DNOTBQ, DNOTLD, DNOTLG, TGAIN, TSGSET		

---

The DNOTBF command sets the center frequency for notch filter B. Setting this to 0 disables the filter. For a description of the filter's transfer function characteristics, refer to the DNOTAF command description.

### **Working with servo gains.**

- Servo tuning process: refer to the *Hardware Installation Guide*.
- Check the values of all active gains (DNOTBF is one of many servo gains): use TGAIN.
- Creating and invoking gain sets: see SGSET, SGENB, TGAIN, TSGSET.

---

## DNOTBQ Notch Filter B Quality Factor

Type	Tuning	Product	Rev
Syntax	<a_><!>DNOTBQ<r>	GT6K	n/a
Units	r = quality factor	GV6K	6.0
Range	0.5 to 2.5		
Default	1		
Response	DNOTBQ: *DNOTBQ1.5		(applicable only to servo axes)
See Also	DNOTAF, DNOTAQ, DNOTBD, DNOTBF, DNOTLD, DNOTLG, TGAIN, TSGSET		

---

The DNOTBQ command sets the quality factor (Q) for notch filter B. For a description of the filter's transfer function characteristics, refer to the DNOTAF command description.

### Working with servo gains.

- Servo tuning process: refer to the *Hardware Installation Guide*.
- Check the values of all active gains (DNOTBQ is one of many servo gains): use TGAIN.
- Creating and invoking gain sets: see SGSET, SGENB, TGAIN, TSGSET.

---

## DNOTLD Notch Lead Filter Break Frequency

Type	Tuning	Product	Rev
Syntax	<a_><!>DNOTLD<i>	GT6K	n/a
Units	i = Hz	GV6K	6.0
Range	0 (disable), or 80-1000		
Default	0 (filter is disabled)		
Response	DNOTLD: *DNOTLD200		(applicable only to servo axes)
See Also	DNOTAD, DNOTAF, DNOTAQ, DNOTBD, DNOTBF, DNOTLG, TASX, TGAIN, TSGSET		

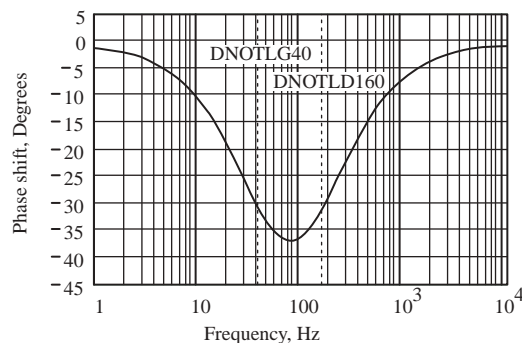
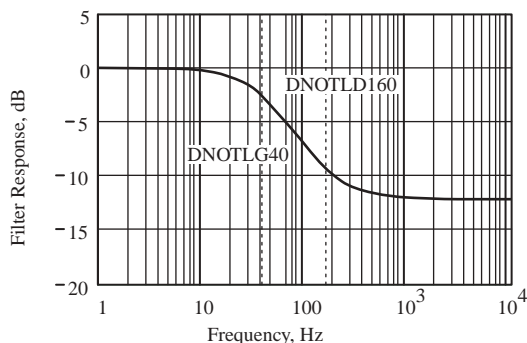
---

The DNOTLD command sets the break frequency of the lead filter. This filter cannot be used alone, but must be used in conjunction with the DNOTLG lag filter. The DNOTLG lag filter must be configured before the DNOTLD lead filter is configured.

The DNOTLD value must be less than or equal to 4 times the DNOTLG (notch lag frequency) value; otherwise, the new DNOTLD value will be ignored (but not overwritten), the configuration warning bit (TASX bit #28) will be set, and the last valid DNOTLD value will be used internally.

This filter operates in all DMODE settings, except Autorun (DMODE13) and Torque/Force Tuning mode (DMODE15).

In the graphs below, the transfer function is shown relating the internal commanded torque/force vs. the user commanded torque/force. In this example, the lag frequency was set first to 40 Hz (DNOTLG40) and then the lead filter was set to 160 Hz (DNOTLD).



### Working with servo gains.

- Servo tuning process: refer to the *Hardware Installation Guide*.
- Check the values of all active gains (DNOTLD is one of many servo gains): use TGAIN.
- Creating and invoking gain sets: see SGSET, SGENB, TGAIN, TSGSET.

---

## **DNOTLG**      **Notch Lag Filter Break Frequency**

Type	Tuning	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DNOTLG<i>	GT6K	n/a
Units	i = Hz	GV6K	6.0
Range	0 (disable), or 20-1000		
Default	0 (filter is disabled)		
Response	DNOTLG: *DNOTLG400		(applicable only to servo axes)
See Also	DNOTAD, DNOTAF, DNOTAQ, DNOTBD, DNOTBF, DNOTBQ, DNOTLD, TGAIN, TSGSET		

---

The DNOTLG command sets the break frequency of the lag filter. This filter can be used alone, or in conjunction with lead filter (DNOTLD) to improve the phase response of the notch filters. In this case, the lag value (DNOTLG) must be greater than or equal to ¼ of the lead value (DNOTLD), but not greater than the DNOTLD value.

If DNOTLG is lower than ¼ the value of DNOTLD, the new DNOTLG value will be ignored (but not overwritten), the configuration warning bit (TASX bit #28) will be set, and the last valid DNOTLG value will be used internally.

This filter operates in all DMODE settings, except Autorun (DMODE13) and Torque/Force Tuning mode (DMODE15).

### **Working with servo gains.**

- Servo tuning process: refer to the *Hardware Installation Guide*.
- Check the values of all active gains (DNOTLG is one of many servo gains): use TGAIN.
- Creating and invoking gain sets: see SGSET, SGENB, TGAIN, TSGSET.

---

## **DPASS**      **Change RP240 Password**

Type	Display (RP240) Interface	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DPASS<i>	GT6K	6.0
Units	i = integer of up to 4 characters	GV6K	6.0
Range	1 - 9999		
Default	For the Gx6K: DPASS6000		
Response	DPASS: * DPASS6000		
See Also	DCLEAR, DLED, DPCUR, DSTP, DVAR, DVARB, DVARI, 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:**

```
DPASS1234                    ; New password = 1234
```

---

## **DPBW**      **Position Loop Bandwidth**

Type	Tuning	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DPBW<r>	GT6K	n/a
Units	r = Hz	GV6K	6.0
Range	1.00 to 100.00 : ±0.01		
Default	5.00		
Response	DPBW: *DPBW10.00		(applicable only to servo axes)
See Also	DIBW, DMTR, SGPRAT, LJRAT, TGAIN, TSGSET		

---

**AUTO-SETUP:** This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6). If you did not use the configuration utility or are not using a Parker Motor, this command is set to zero and you will have to manually set this parameter. Refer to DMTR for a list of auto-configured commands.

The DPBW command sets the bandwidth of the position loop. Higher values will increase responsiveness and dynamic stiffness. Excessive position loop bandwidth can result in reduced stability, causing long settling times; this is particularly true in applications with resonant mechanics.

**Working with servo gains.**

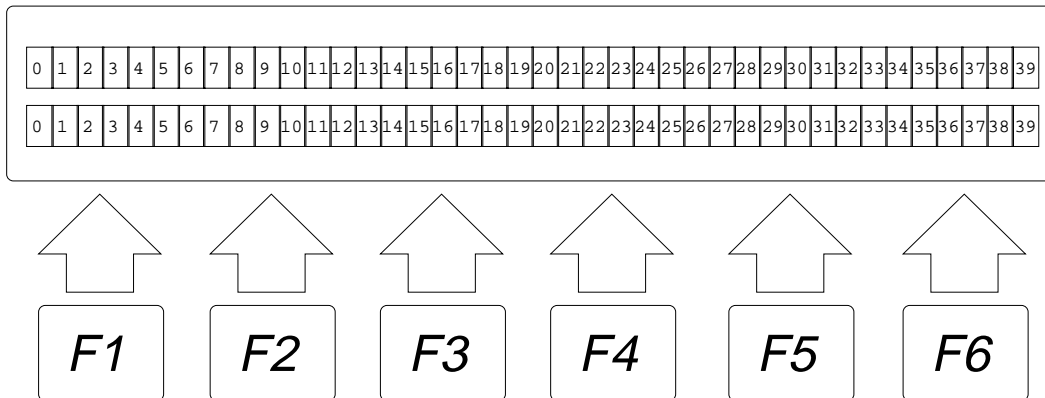
- Servo tuning process: refer to the *Hardware Installation Guide*.
- Check the values of all active gains (DPBW is one of many servo gains): use TGAIn.
- Creating and invoking gain sets: see SGSET, SGENB, TGAIn, TSGSET.

---

<b>DPCUR</b>	<b>Position Cursor</b>	<b>Product</b>	<b>Rev</b>
Type	Display (RP240) Interface	GT6K	6.0
Syntax	<a_><!>DPCURi,i	GV6K	6.0
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, DSTP, DVAR, DVARI, DVARB, DWRITE		

---

The 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:**  
DPCUR2,15 ; Position cursor on line 2, column 15

---

<b>DPHBAL</b>	<b>Phase Balance</b>	<b>Product</b>	<b>Rev</b>
Type	Drive Configuration	GT6K	6.0
Syntax	<a_><!>DPHBAL<r>	GV6K	n/a
Units	r = percent of (peak) nominal commanded current		
Range	90.0 to 110.0 : ±0.1		
Default	100.0		(applicable only to stepper axes)
Response	DPHBAL: *DPHBAL100.0		
See Also	DPHOFA, DPHOFB, DWAVEF		

---

The DPHBAL command adjusts the current amplitude of phase B with respect to phase A. A procedure for configuring Phase Balance is provided in the *Configuration* chapter of the *Hardware Installation Guide*.

**Example:**  
DPHBAL110 ; sets the current amplitude of phase B to 110%  
; of the current in phase A.

---

## DPHOFA Phase A Current Offset

Type	Drive Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DPHOFA<r>	GT6K	6.0
Units	r = no units	GV6K	n/a
Range	-10.000 to 10.000 : ±0.001		
Default	0.000		
Response	DPHOFA: *DPHOFA5.000		(applicable only to stepper axes)
See Also	DPHBAL, DPHOFB, DWAVEF		

---

The DPHOFA command adjusts the current offset of phase A.

A procedure for configuring phase offset is provided in the *Configuration* chapter of the *Hardware Installation Guide*.

---

## DPHOFB Phase B Current Offset

Type	Drive Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DPHOFB<r>	GT6K	6.0
Units	r = no units	GV6K	n/a
Range	-10.000 to 10.000 : ±0.001		
Default	0.000		
Response	DPHOFB: *DPHOFB5.000		(applicable only to stepper axes)
See Also	DPHBAL, DPHOFA, DWAVEF		

---

The DPHOFB command adjusts the current offset of phase B.

A procedure for configuring phase offset is provided in the *Configuration* chapter of the *Hardware Installation Guide*.

---

## DPOLE Number of Motor Pole Pairs

Type	Motor	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DPOLE<i> (does not take effect until RESET, DRESET or cycle power)	GT6K	6.0
Units	i = pole pairs	GV6K	6.0
Range	1-200		
Default	0 (DPOLE of 0 results in motor config. error)		
Response	DPOLE: *DPOLE50		
See Also	DMTR, TASX, TCS		

---

**NOTE:** This command does not take effect until you cycle power to the drive, or issue a RESET or DRESET command.

**AUTO-SETUP:** This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6). If you did not use the configuration utility or are not using a Parker Motor, this command is set to zero and you will have to manually set this parameter. (Refer to DMTR for a list of auto-configured commands.) If the drive is powered up when this command is set to zero (for instance, if RFS is executed), the drive reports a motor configuration error with TASX bit 7, writes a value of -32723 to the TCS register, and shuts down the drive (DRIVE0).

The DPOLE command sets the number of motor pole pairs. The number of pole pairs is defined as the number of poles (P), divided by 2 (or, P/2). The electrical frequency of the current ( $\omega_e$ ) is related to the mechanical speed ( $\omega_m$ ) of the motor by the pole pairs. The equation (right) shows this relationship.

$$\omega_e = \left(\frac{P}{2}\right) * \omega_m$$

**NOTES:**

- A 1.8° step motor will have 50 pole pairs, a 0.9° step motor will have 100 pole pairs.
- All linear motors, regardless of the number of stator poles, are considered to be one pole pair (DPOLE1) machines.

## [ DPTR ] Data Pointer Location

Type	Data Storage; Assignment or Comparison	Product	Rev
Syntax	see below	GT6K	6.0
Units	n/a	GV6K	6.0
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:**

```
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.
```

## DPWM Drive PWM Frequency

Type	Drive Configuration	Product	Rev
Syntax	<a_><!>DPWM <i> (does not take effect until RESET, DRESET or cycle power)	GT6K	n/a
Units	kHz	GV6K	6.0
Range	0, 8, 16, 20 or 40		
Default	0 (use the drive's default frequency): GV6K-U3, GV6K-U6, GV6K-U12, GV6K-H20, GV6K-H40: 8 KHz GV6K-L3: 40 KHz		(applicable only to servo axes)
Response	DPWM8		
See Also			

**NOTE:** This command does not take effect until you cycle power to the drive, or issue a RESET or DRESET command.

Use the DPWM command to select the drive's PWM frequency. This value is the internal PWM frequency as seen at the motor windings; the motor ripple current is twice this frequency. In general, for a given drive power level, the higher the switching frequency, the lower the motor ripple current heating and the lower both the peak and continuous current ratings.

The table below lists the drive's continuous current and peak current ratings for different PWM frequencies.

Drive	Input Voltage	CONTINUOUS CURRENT				PEAK CURRENT			
		DPWM8	DPWM16	DPWM20	DPWM40	DPWM8	DPWM16	DPWM20	DPWM40
GV6K-L3	120VAC, 1 phase	n/a	n/a	n/a	3.0A	n/a	n/a	n/a	7.5A
GV6K-U3*	240VAC, 1 phase	3.0A	n/a	n/a	n/a	7.5A	n/a	n/a	n/a
GV6K-U6*	240VAC, 1 phase	6.0A	4.5A	3.6A	n/a	15.0A	11.2A	9.0A	n/a
GV6K-U12*	240VAC, 1 phase	12.0A	9.0A	7.2A	n/a	20.0A	22.5A	18.0A	n/a
GV6K-H20	240VAC, 1 phase	16.0A	13.0A	11.0A	n/a	50.0A	32.5A	27.5A	n/a
GV6K-H20	240VAC, 3 phase	20.0A	13.0A	11.0A	n/a	50.0A	32.5A	27.5A	n/a
GV6K-H40	240VAC, 1 phase	23.1A	23.1A	22.0A	n/a	100.0A	65.0A	55.0A	n/a
GV6K-H40	240VAC, 3 phase	40.0A	26.0A	22.0A	n/a	100.0A	65.0A	55.0A	n/a

\* Current Ratings are the same for 120VAC operation

Note: All currents are peak of the sine wave values

---

## [ DREAD ] Read RP240 Data

Type	Display (RP240) Interface	<b>Product</b>	<b>Rev</b>
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	[ DREADF ], DREADI, DVAR, DWRITE, [ SS ], TSS, VAR		

---

The DREAD command allows you to store numeric data entered 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 (Rule of Thumb: If the command syntax shows that the command field requires a real number (denoted by <r>) or an integer value (denoted by <i>), you can use the DREAD substitution.):

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

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

---

## [ DREADF ] Read RP240 Function Key

Type	Display (RP240) Interface	<b>Product</b>	<b>Rev</b>
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	[ DREAD ], DREADI, DVAR, DWRITE, [ SS ], TSS, VAR, VARI		

---

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

Rule of Thumb for command value substitutions: If the command syntax shows that the command field requires a real number (denoted by <r>) or an integer value (denoted by <i>), you can use the DREADF substitution (e.g., V(DREADF)).

**Example:**

```
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 ...
GO1              ; Start motion
NIF              ; End if statement
```

---

## DREADI      RP240 Data Read Immediate Mode

Type	Display (RP240) Interface	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DREADI<b>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	1 (enable) or 0 (disable)		
Default	0		
Response	DREADI: *DREADI0		
See Also	DPCUR, [ DREAD ], [ DREADF ], VAR, VARI		

---

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.

<b>NOTES</b>
--------------

- While in the Data Read Immediate Mode (DREADI1), data is read into VAR and VARI variables only (e.g., A(DREAD) or V(DREAD) substitutions are not valid).
  - This feature is not designed to be used in conjunction with the RP240's standard menus (see *Programmer's Guide* for menu structure); the **RUN** and **JOG** menus will disable the DREADI mode.
  - After the RP240's ENTER key is pressed (to enter numeric data), the value is displayed on the RP240 display at the 1,30 location (showing 10 significant digits).
  - Do not assign the same variable to read numeric data **and** function key data—pick only one.
- 

### Simple Numeric Data Entry (example):

```
VAR1=25000      ; Initialize variable #1
DCLEAR0        ; 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
LN           ; End loop
DREADI0       ; 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 Gem6K drive is ready to accept new data.
```

### Numeric Data & Function Key Entry (example):

```
VAR1=25000      ; Initialize variable #1
VAR2=1          ; Initialize variable #2
DCLEAR0        ; Clear the RP240 display
DPCUR2,0       ; 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
DPCUR1,0       ; Place RP240 cursor on line 1, column 0 (top left corner)
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
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 (example):**

```

VAR2=0         ; Initialize variable #2 (VAR2)
VAR3=99        ; Initialize variable #3 (VAR3)
VAR4=10        ; Initialize variable #4 (VAR4)
VAR5=25000     ; Initialize variable #5 (VAR5)
DCLEAR0       ; Clear the entire RP240 display
DPCUR2,0      ; Place RP240 cursor on line 2, column 0 (bottom left corner)
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<>0)   ; 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=0        ; 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
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

Type	Drive Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DRES<i>	GT6K	6.0
Units	i = counts/rev	GV6K	6.0
Range	200-128000		
Default	25000		
Response	DRES: *DRES25000		
See Also	DMEPIT, DMODE, DRIVE, ERES, SCALE, TSTAT		

---

Use the DRES command to set the Gemini6K's resolution appropriately for your application. This resolution determines how many distinct positions the drive will command the motor to travel within one revolution.

### Example:

```
DRES25000 ; Set drive resolution to 25000 counts/rev
```

**NOTE:** This command does not take effect until you cycle power to the drive, or issue a RESET or DRESET command.

---

## DRESET Drive Reset

Type	Communication Interface	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DRESET	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	DRESET: (no response; parameters listed below are reset)		
See Also	RESET, RFS, TASX		

---

The DRESET command performs a limited “software” reset of the parameters listed below. Unlike the RESET command, which affects the drive the same as cycling power or activating the hardware Reset input, the DRESET command affects only the parameters below.

### Resolution Commands

DRES  
ERES  
ORES

### Status Registers (certain bits cleared by DRESET)

ASX TASX TASXF

### Motor Configuration Parameters

DMEPIT  
DMTD  
DMTIC  
DMTICD  
DMTIND  
DMTJ  
DMTKE  
DMTLMN  
DMTLMX  
DMTR  
DMTRES  
DMTRWC  
DMTSTT  
DMTTCM  
DMTTCW  
DMTW  
DPOLE  
DPWM  
SFB  
SHALL

**NOTE:** The DRESET command does not disconnect an Ethernet connection.  
The DRESET command does not clear the ERRORP assignment or run the STARTP program.

---

## DRIVE

### Drive Enable

Type	Drive Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DRIVE<b>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	b = 0 (disable), 1 (enable)		
Default	0 (disable)		
Response	DRIVE *DRIVE1		
See Also	[ AS ], [ ASX ], ESK, FLTDSB, KDRIVE, TAS, TASX, TER		

---

Use the DRIVE command to enable or disable (shut down) the drive. By default, the drive is disabled (DRIVE0) upon power up. The enable input (pin 1 on the 50 pin DRIVE I/O connector) must be connected to ground before a DRIVE1 will enable the drive.

If the Disable Drive on Kill (KDRIVE) mode is enabled, the drive will be disabled in the event of a kill command or kill input.

**NOTE:** The DRIVE0 command will not de-energize a motor drive during motion.

DRIVE1 sets the commanded position (TPC) equal to the actual position (TPE).

The conditions below automatically disable the drive (DRIVE0), activate any outputs you have configured as “fault” outputs, and open the dry contact relay (“RELAY N.O.”):

- Certain axis “fault” conditions – refer to the status bits denoted with an asterisk (\*) in the TAS and TASX descriptions.
- If operating in the FLTDSB1 mode and the drive received a DRIVE0 command or the hardware enable input interlock was opened.

**Example:**

```
DRIVE1 ; Enable the drive
```

---

## DRPCHK

### RP240 Check

Type	Communication Interface; Display (RP240) Interface	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DRPCHK<i>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	0-3		
Default	0 for port COM1 and COM2 (PORT command setting determines which COM port's DRPCHK setting is checked)		
Response	DRPCHK *DRPCHK0		
See Also	LOCK, PORT, XONOFF		

---

The DRPCHK command is used to indicate whether a port is to be used with an RP240 or for Gem6K language commands. The DRPCHK command affects the COM port selected with the last PORT command. The DRPCHK command value is automatically saved in battery-backed RAM.

**NOTE:** COM1 is the connector labeled “RS-232/485” and COM2 is the connector labeled “RS-232.”

DRPCHK0..... The serial port will be used for Gem6K language commands. This is the default setting for COM1 and COM2. Power-up messages appear on all ports set to DRPCHK0.

DRPCHK1..... A check for the presence of an RP240 will be performed at power-up/reset. If an RP240 is present and energized, the Gem6K will initialize the RP240. If an RP240 is not present, the port may be used for Gem6K language commands. Note that RP240 commands will be sent at power-up and reset.

DRPCHK2..... A status check for the presence of an RP240 will be periodically performed (every 5-6 seconds). If an RP240 is plugged in and energized, the Gem6K will initialize the RP240.

Press F6 on the RP240 periodically until the Gem6K recognizes the RP240. (The RP240 indicates that it has been recognized by beeping when F6 is pressed.)

DRPCHK3 ..... A status check for the presence of an RP240 will be performed at power-up/reset. If an RP240 is present, the Gem6K will initialize the RP240. If an RP240 is not present, no commands except DWRITE will have any effect for that port and the COM port will ignore received characters.

Each port has its own DRPCHK value, but only one may be set to DRPCHK2 or DRPCHK3 at any time.

Default values are used until DRPCHK is set for the first time. DRPCHK values are automatically saved in non-volatile memory. They do not change until you set new values or issue an RFS command. It may be advisable to include the DRPCHK command in your start-up program to ensure that it powers up in the correct setting for your current application.

---

## DSTALL Drive Stall Detect Sensitivity

Type	Drive Configuration	Product	Rev
Syntax	<a_><!>DSTALL<i>	GT6K	6.0
Units	i = sensitivity level selection	GV6K	n/a
Range	0-50 (0 disables the stall detect function)		
Default	0 (disabled)		(applicable only to stepper axes)
Response	DSTALL: *DSTALL20		
See Also	ERROR, ESK, ESTALL, KDRIVE, TAS, TASX, TER		

---

The DSTALL command sets the sensitivity of the *Drive Stall Detection* function. An encoder is not required for this function. (An encoder is required for *Encoder Stall Detection*; refer to the ESTALL command.)

A setting of 0 disables the drive stall detect function. Stall “sensitivity” is an empirical metric; therefore, you must iteratively try different settings to identify the sensitivity setting that works best for your application.

When a Drive Stall is detected, the GT6K responds as follows:

- The stall is reported with Extended Axis Status bit #17 (reported with TASX, TASXF, and ASX).
- If error-checking bit #1 is enabled (ERROR.1-1):
  - The stall is reported with Error Status bit #1 (reported with TER, TERF, and ER).
  - The GT6K branches to the assigned ERRORP program.
- If the Kill-on-Stall feature is enabled (ESK1), the GT6K immediately stops pulses from being sent to the motor. If Disable Drive on Kill is enabled (KDRIVE1) it disables the drive.

### Example:

```
SCALE0           ; Disable scaling
DEL proga        ; Delete program called proga
DEF proga        ; Begin definition of program called proga
DSTALL20         ; Enable drive stall detect, with sensitivity of 20
ESK1             ; Enable kill on stall
MA0              ; Incremental positioning mode
MCO              ; Preset positioning mode
A10              ; Set the acceleration to 10 units/sec/sec
V1               ; Set the velocity to 1 unit/sec
D100000         ; Set the distance to 100000 units
GO1              ; Initiate motion
                 ; The motor will move 100000 commanded counts (4 revs)
                 ; (If, at any time during the move the drive detects a stall,
                 ; a stall condition will be flagged, and motion will stop.)
END              ; End definition of proga
```

---

## DSTP                    Enable/Disable RP240 Stop Key

Type	Display (RP240) Interface	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DSTP<b>	GT6K	6.0
Units	b = enable bit	GV6K	6.0
Range	b = 1 (enable) or 0 (disable)		
Default	1 (enable)		
Response	DSTP        *DSTP1		
See Also	DLED, [ DREAD ], [ DREADF ]		

---

Use the DSTP command to enable or disable the stop key on the RP240 panel.

---

## DVAR                    Display Variable on RP240

Type	Display (RP240) Interface	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DVARi,<i>,<i>,<i>	GT6K	6.0
Units	See below	GV6K	6.0
Range	n/a		
Default	See below		
Response	n/a		
See Also	[ DREAD ], [ DREADF ], DVARB, DVARI, DWRITE, VAR		

---

Use the DVAR command to display a numeric variable on the RP240's LCD at the current cursor location:

1<sup>st</sup> i = Variable number [Range 1-225]

2<sup>nd</sup> i = Number of whole digits displayed (left of decimal point) [Range 0-9]

3<sup>rd</sup> i = Number of fractional digits displayed (right of decimal point) [Range 0-8]

4<sup>th</sup> i = Sign bit: 0 = no sign displayed, 1 = display + or -

### Example:

```
VAR2=542.14            ; Assign the value 542.14 to variable #2
DVAR2,6,3,1            ; Display variable #2 as +000542.140
DVAR2,3,1,0            ; Display variable #2 as 542.1
DVAR2,3,,1             ; Display variable #2 as +542
```

---

## DVARB                  Display Binary Variable on RP240

Type	Display (RP240) Interface	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DVARBi,<i>	GT6K	6.0
Units	See below	GV6K	6.0
Range	n/a		
Default	See below		
Response	n/a		
See Also	DVAR, DVARI, DWRITE, VARB		

---

Use the DVARB command to display a binary variable on the RP240's LCD at the current cursor location:

1<sup>st</sup> i = Variable number [Range 1-125]

2<sup>nd</sup> i = Number of bits displayed [Range 1-32]

### Example

```
VARB2=b11001X11       ; Assign the value 11001X11 to binary variable #2
DVARB2,6               ; Display binary variable #2 as 1100_1X
DVARB2,3               ; Display binary variable #2 as 110
DVARB2,1               ; Display binary variable #2 as 1
```

---

## DVARI Display Integer Variable on RP240

Type	Display (RP240) Interface	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DVARIi,<i>,<i>	GT6K	6.0
Units	See below	GV6K	6.0
Range	n/a		
Default	See below		
Response	n/a		
See Also	DVAR, DVARB, DWRITE, VARI		

---

Use the DVARI command to display an integer variable on the RP240's LCD at the current cursor location:

- 1<sup>st</sup> i = Variable number [Range 1-225]
- 2<sup>nd</sup> i = Number of whole digits displayed [Range 0-9]
- 3<sup>rd</sup> i = Sign bit: 0=no sign displayed, 1=display + or - sign

### Example

```
VARI2=542 ; Assign the value 542 to integer variable #2
DVARI2,6,1 ; Display integer variable #2 as +000542
DVARI2,3,1 ; Display integer variable #2 as +=542
```

---

## DWAVEF Waveform

Type	Drive Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DWAVEF<r>	GT6K	6.0
Units	r = percent 3 <sup>rd</sup> harmonic	GV6K	n/a
Range	-20.00 to 10.00 : ±0.01		
Default	-4.00 (-4% 3 <sup>rd</sup> harmonic injection)		(applicable only to stepper axes)
Response	DWAVEF *DWAVEF-4		
See Also	DPHBAL, DPHOFA, DPHOFB		

---

The DWAVEF command sets the percentage of 3<sup>rd</sup> harmonic included in the commanded motor current waveform. A procedure for configuring the waveform is provided in the *Hardware Installation Guide*; refer to the section entitled *Motor Control Settings*.

---

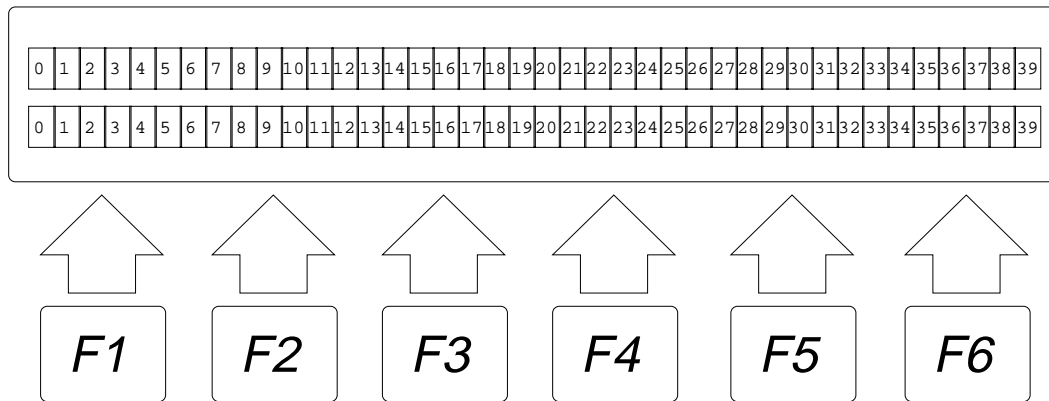
## DWRITE Write Text on RP240

Type	Display (RP240) Interface	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DWRITE"message"	GT6K	6.0
Units	n/a	GV6K	6.0
Range	Message can be ≤ 80 characters (may not use characters ", \, * or :)		
Default	See below		
Response	n/a		
See Also	DCLEAR, DLED, DPASS, DPCUR, DVAR, DVARB, DVARI, PORT		

---

The 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 80 characters in length. The characters within the string may be any characters except quote ("), backslash (\), asterisk (\*), colon (:), and tilde (~). 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.



**HINT:** If you do not have an RP240 and wish to send (only) characters out the serial port to another serial device, you may use this command. Place a backslash (\) before non-alphanumeric characters.  
 Example: `DWRITE"HOMING SUCCESSFUL\13"` = send message plus <CR>

**Example:**

```
DCLEAR0           ; 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
```

---

<b>E</b>	<b>Enable Communication</b>		
Type	Communication Interface	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>E<b>	GT6K	6.0
Units	b = enable bit	GV6K	6.0
Range	b = 0 (serial communication off) or 1 (serial communication on)		
Default	b = 1		
Response	0_E: *E1		
See Also	ADDR, BAUD, DRPCHK, ECHO, LOCK, PORT, XONOFF		

---

The **E** command allows you to enable and disable serial ports on the Gem6K drive. To enable all units in the RS-232 daisy-chain or RS-485 multi-drop at one time, you can use the **E1** command. The **PORT** command determines which COM port is affected by the **E** command.

**Example:**

```
PORT1           ; Next command affects the COM1 serial port on the Gem6K
0_E1           ; Enable serial port for unit with device address 0
```

---

<b>ECHO</b>	<b>Communication Echo Enable</b>		
Type	Communication Interface	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>ECHO<b>	GT6K	6.0
Units	b = enable bit	GV6K	6.0
Range	b = 0 (disable), 1 (enable), 2 (echo through other COM port), 3 (echo through both COM ports), or X (don't change)		
Default	1		
Response	ECHO: *ECHO0		
See Also	], [, BAUD, EOL, EOT, ERRVL, PORT, [ 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.

The `PORT` command determines which COM port is affected by the `ECHO` command.

The purpose of the `ECHO2` and `ECHO3` options is to accommodate an RS-485 multi-drop configuration in which a host computer communicates to the “master” Gem6K drive over RS-232 (COM2 port) and the master Gem6K drive communicates over RS-485 (COM1 port) to the rest of the units on the multi-drop. For this configuration, the echo setup should be configured by sending to the master the following commands executed in the order shown. In this example, it is assumed that the master's device address is set to 1. Hence, each command is prefixed with “1\_” to address only the master unit.

- 1\_PORT1..... Subsequent command affects COM1, the RS-485 port
- 1\_ECHO2..... Echo characters back through the other port, COM2
- 1\_PORT2..... Subsequent command affects COM2, the RS-232 port
- 1\_ECHO3..... Echo characters back through both ports, COM1 and COM2

---

<b>ELSE</b>	<b>Else Condition of IF Statement</b>		
Type	Program Flow Control	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>ELSE	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	IF, NIF		

---

ELSE 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, and 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 and 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:**

```
IF(IN.1=b1)      ; Specify condition: if onboard input #1 is on
T5              ; If condition evaluates true, wait 5 seconds
ELSE            ; Else part of IF condition
TPE             ; If condition does not evaluate true transfer encoder position
NIF             ; End IF statement
```

---

<b>ENCCNT</b>	<b>Encoder Count Reference Enable</b>		
Type	Encoder; Drive Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>ENCCNT<b>	GT6K	6.0
Units	b = enable bit	GV6K	n/a
Range	0 (reference the commanded position), 1 (reference the encoder position) or X (don't change)		(applicable only to stepper axes)
Default	0 (reference the commanded position)		
Response	ENCCNT *ENCCNT0		
See Also	ESTALL, INFNC, LIMFNC, OUTP, [ PCC ], [ PCE ], [ PCME ], [ PE ], TPCC, TPCE, TPCME, TPE, TVELA, [ VELA ]		

---

Use ENCCNT to configure a GT6K to reference either the encoder position or the commanded position when capturing the position (see INFNCi-H) and checking the encoder position (PE and TPE). When checking the actual velocity (VELA and TVELA), ENCCNT determines whether the velocity, in units of revs/sec, is derived



with the encoder resolution (ERES) or the drive resolution (DRES). The default setting (ENCCNT0) references the commanded position.

- NOTES:**
- ENCCNT must be enabled (ENCCNT1) before encoder stall detect (ESTALL) can be used.
  - When operating in ENCCNT1 mode, the encoder position (PCE, PE, TPCE, TPE) is always measured in encoder counts and is not scaled, regardless of SCALE and SCLD.

**Example:**

```
INFNC1-H      ; Configure trigger A as a position capture input
ENCCNT1      ; Capture encoder position when trigger A is activated,
```

---

<b>END</b>	<b>End Program or Subroutine Definition</b>		
Type	Program or Subroutine Definition	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>END	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	\$, DEF, DEL, ERASE, GOBUF, GOSUB, GOTO, RUN, PRUN		

---

The END command marks the ending point of a program or subroutine definition. All commands between the DEF and the END statement will be considered in a program or subroutine.

**Example:**

```
DEL pick      ; Delete program named pick
DEF pick      ; Begin definition of program named pick
G01           ; Initiate motion
END           ; End program definition
pick         ; Execute program named pick
```

---

<b>EOL</b>	<b>End of Line Terminating Characters</b>		
Type	Communication Interface	<b>Product</b>	<b>Rev</b>
Syntax	<!>EOL<i>,<i>,<i>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	i = 0 - 256		
Default	13,10,0		
Response	EOL: *EOL13,10,0		
See Also	], [, BOT, EOT, ERRVLV, PORT, WRITE, XONOFF		

---

The 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:** ASCII 256 transmits nothing. ASCII 256 transmits a null ASCII zero character, which is typically used to create a null terminated string in a C language.

The PORT command determines which COM port is affected by the EOL command.

**NOTE:** Although you may issue a single command, like TERRLG, 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

For a more complete list of ASCII Equivalents, refer to the ASCII Table in [Appendix B](#).

**Example:**

```
EOL13,0,0    ; Place a carriage return after each line of a response
```

---

## EOT End of Transmission Characters

Type	Communication Interface	<b>Product</b>	<b>Rev</b>
Syntax	<!>EOT<i>,<i>,<i>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	i = 0 – 256		
Default	13,0,0		
Response	EOT: *EOT13,0,0		
See Also	], [, BOT, EOL, ERRVLV, PORT, WRITE		

---

The 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, and no terminating character is designated with a zero.

**NOTE:** ASCII 256 transmits nothing. ASCII 256 transmits a null ASCII zero character, which is typically used to create a null terminated string in a C language.

The PORT command determines which COM port is affected by the EOT command.

**NOTE:** Although you may issue a single command, like TERRLG, 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

For a more complete list of ASCII Equivalents, refer to the ASCII Table in [Appendix B](#).

### Example:

```
EOT13,10,0 ; Place a carriage return and a line feed after the last line  
           ; of a multi-line response, and after all single line responses
```

---

## [ ER ] Error Status

Type	Assignment or Comparison	<b>Product</b>	<b>Rev</b>
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	[ ASX ], DSTALL, ERROR, ERRORP, ESTALL, GOWHEN, INFNC, K, LH, LIMFNC, LS, S, SMPER, STRGTT, TASX, TER, TERF		

---

The 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:** VARn=<i%>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=b1) 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.

## MULTI-TASKING

If you are operating multiple tasks, be aware that you must enable error conditions (ERROR) and specify an error program (ERRORP) for each task (e.g., `2%ERROR.2-1` and `2%ERRORP.FIX` for Task 2). Each task has its own error status register (reported with ER, TER, and TERF).

The function of each error status bit is shown below.

Bit #	Function (1 = Yes; 0 = No)
1*	Stall Detected: Functions when stall detection has been enabled (ESTALL or DSTALL), and ESK1 is set.
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: Detected only if the drive is enabled (DRIVE1).
5	RESERVED (see note at end of table; refer to ERROR command)
6	Kill Input: When an input is defined as a Kill input (INFNCi-C), and that input becomes active.
7	User Fault Input: When an input is defined as a User Fault input (INFNCi-F), and that input becomes active.
8	Stop Input: When an input is defined as a Stop input (INFNCi-D), and that input becomes active.
9	Enable input is activated (not grounded).
10	Pre-emptive (on-the-fly) GO or registration move profile not possible.
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	RESERVED
14	Position relationship in GOWHEN already true when GO, FSHFC, or FSHFD was executed.
15	RESERVED
16	Bad command detected (bit is cleared with TCMDEP command).
17	RESERVED
18	Cable to an expansion I/O brick is disconnected, or power to the I/O brick is lost; error is cleared by reconnecting the I/O brick and issuing the ERROR.18-0 command and then the ERROR.18-1 command. (Multi-tasking: This condition also kills all tasks.)
19	RESERVED
20	RESERVED
21	RESERVED
22	RESERVED
23	Ethernet Client connection error.
24	Ethernet Client polling error.
25-32	RESERVED

\* Stepper axes only

\*\* Servo axes only

**NOTE:** 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 drive to GOSUB or GOTO to the error program (ERRORP). Within the error program the cause of the error will need to be determined. The ER or 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.

**Example:**

```

ERROR111101101      ; 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
IF(ER=b1110X11X1)   ; 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=hF600)        ; 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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>ERASE	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	[ DATP ], DEF, DEL, RESET, RFS		

---

The 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. See the RESET command to determine which values are reset to factory defaults.

---

## ERES Encoder Resolution

Type	Encoder Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>ERES<i>>:(does not take effect until RESET, DRESET or cycle power)	GT6K	6.0
Units	Rotary motor: i = counts/rev Linear motor: i = counts/electrical pitch	GV6K	6.0
Range	0 - 65535 (stepper axes); 200 - 1073741823 (servo axes)		
Default	4000 (If SFB4, the default is 4096 counts/rev.)		
Response	ERES: *ERES4000		
See Also	DMEPIT, DMTR, DRES, ESTALL, ORES, SFB, SMPER, TPE, TPER, TSTAT		

---

Use the ERES command to establish the encoder or resolver resolution (post quadrature) in counts/rev or counts/electrical pitch. (To set a linear motor's electrical pitch, refer to the DMEPIT command).

**Stepper Axes:** The 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.

**Servo Axes:** The servo system's resolution is determined by the resolution of the encoder used with the servo motor. The ERES command establishes the number of counts (post quadrature), per unit of travel. For example, Compumotor's SM and NeoMetric Series motors with the "E" encoder option use 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 GV6K's resolution (ERES value) must match the encoder's resolution.

**NOTE:** This command does not take effect until you cycle power to the drive, or issue a RESET or DRESET command.

**AUTO-SETUP for GV6K Drives only:** This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6). If you did not use the configuration utility or are not using a Parker Motor, this command is set to zero and you will have to manually set this parameter (if SFB1) or this command is set to 4096 counts/rev (if SFB4). Refer to DMTR for a list of auto-configured commands.

## Resolutions for Compumotor Encoders:

### Stepper axes:

- -RE, -RC, -EC, and -E Series Encoders ....ERES4000
- -HJ Series Encoders .....ERES2048

### Servo axes:

- BE Series Servo Motors .....BExxxxJ-xxxx: ERES8000
- SM, N or J Series Servo Motors.....SM/N/JxxxxD-xxxx: ERES2000  
SM/N/JxxxxE-xxxx: ERES4000
- G Series Servo Motors .....GxxxxK-xxxx: ERES8192
- Resolution for Parker-supplied resolvers ..SFB4: ERES4096 (BE, SM, N, J and G Series motors)

### Daedal Positioning Tables (encoder options):

- -E2.....ERES42000
- -E3.....ERES84000
- -E4.....ERES420000
- -E5.....ERES8400
- For linear servo motors, use the following equation to determine the proper ERES, based on both the encoder resolution and the motor's electrical (or magnetic) pitch (DMEPIT).

$$ERES = \frac{DMEPIT \text{ (mm)}}{\text{Encoder\_resolution (mm/count)}}$$

**Example:** Linear encoder resolution (post quad) is 1 µm and the electrical pitch is 42mm (DMEPIT42). ERES is calculated as:

$$ERES = \frac{42 \text{ (mm)}}{1 \cdot 10^{-3} \text{ (mm/count)}} = 42000$$

### Dynaserv (stepper or servo):

- DR10xxB .....ERES507904
- DR1xxxE.....ERES614400
- DR1xxxA .....ERES819200
- DR5xxxB .....ERES278528
- DR5xxxA .....ERES425894
- DM10xxB .....ERES655360
- DM1xxxA .....ERES1024000
- DM1004x .....ERES655360

### Example (for stepper axis):

```
SCALE0          ; Disable scaling
DEL proga       ; Delete program called proga
DEF proga       ; Begin definition of program called proga
DRES25000       ; Motor/drive resolution set to 25000 steps/rev
                ; (DRES is used for stepper axes only)
ERES4000        ; Encoder resolution set to 4000 post-quadrature counts/rev
                ; (encoder is for stall detection only)
ESTALL1        ; Enable encoder stall detection
MA0            ; Incremental mode
MC0            ; Preset mode
A10            ; Set the acceleration to 10 units/sec/sec
V1             ; Set the velocity to 1 unit/sec
D100000        ; Set the distance to 100000 units
GO1            ; Initiate motion
                ; Axis will move 100,000 commanded counts (4 revs)
END            ; End definition of proga
```

---

## ERRBAD Error Prompt

Type	Communication Interface	<b>Product</b>	<b>Rev</b>
Syntax	<!>ERRBAD<i>,<i>,<i>,<i>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	i = 0 - 256		
Default	13,10,63,32		
Response	ERRBAD: *ERRBAD13,10,63,32		
See Also	BOT, EOT, ERRDEF, ERRLVL, ERROK, PORT, TCMDER		

---

The 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 B](#).

**NOTE:** ASCII 256 transmits nothing. ASCII 256 transmits a null ASCII zero character, which is typically used to create a null terminated string in a C language.

The PORT command determines which COM port is affected by the ERRBAD command.

**Example:**

```
ERRBAD13,0,0,0 ; Place a carriage return only in the output buffer after  
                ; processing an erroneous command
```

---

## ERRDEF Program Definition Prompt

Type	Communication Interface	<b>Product</b>	<b>Rev</b>
Syntax	<!>ERRDEF<i>,<i>,<i>,<i>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	i = 0 - 256		
Default	13,10,45,32		
Response	ERRDEF: *ERRDEF13,10,45,32		
See Also	BOT, END, EOT, ERRBAD, ERRLVL, ERROK, PORT, XONOFF		

---

The ERRDEF command designates the characters to be placed into the output buffer after a DEF has been entered. Up to 4 characters can be placed in the output buffer. These characters will continue to be placed into the output buffer after each command, until the END command is processed. 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 B](#).

**NOTE:** ASCII 256 transmits nothing. ASCII 256 transmits a null ASCII zero character, which is typically used to create a null terminated string in a C language.

The PORT command determines which COM port is affected by the ERRDEF command.

**Example:**

```
ERRDEF13,0,0,0 ; Place a carriage return only in the output buffer after each  
                ; command in the program definition
```

---

## ERRLVL      Error Detection Level

Type	Error Handling	Product	Rev
Syntax	<a_><!>ERRLVL<i>	GT6K	6.0
Units	i - error level settings	GV6K	6.0
Range	i = 0, 1, 2, 3, or 4		
Default	4 if COM port is set up for RS-232C; 0 if COM port is set up for RS-485		
Response	ERRLVL: *ERRLVL4		
See Also	BOT, EOL, EOT, ERBAD, ERRDEF, ERROK, PORT		

---

The `ERRLVL` command has options which allow you to include or exclude certain elements of the response from the Gem6K (see table below). The `PORT` command determines which COM port is affected by the `ERRLVL` command.

### Overview:

#### ◆ When you enter or send a command:

Under factory default conditions (error detection level 4, selected with the `ERRLVL4` command), after you type a command and press the ENTER key, or send a command via the `COM6SRVR` communications server, the Gem6K responds as follows:

- If an error is not detected, the `ERROK` characters are transmitted. The default `ERROK` transmission characters are (in order of transmission): carriage return, line feed, “>” prompt, and space.
- If an error is detected, the following is transmitted (in this order):
  1. `BOT` (beginning of transmission) characters. The default `BOT` transmission is no characters.
  2. Carriage return
  3. Line feed
  4. Asterisk (\*)
  5. Error message (if there is one corresponding to the error) — see list on page 11.
  6. `EOT` (end of transmission) characters. The default `EOT` transmission is a carriage return.
  7. `ERBAD` characters. The default `ERBAD` transmission characters are (in order of transmission): carriage return, line feed, “?” prompt, and space.

#### ◆ Soliciting information and checking responses:

In each command description, there is a “Response” field that identifies the potential response(s) given when the command is executed. For example, parameter-based commands may be executed without parameters to report the current configuration; *transfer* commands may be executed to report certain status conditions. Under factory default conditions (`ERRLVL4`), and if no error is detected, the response characters are transmitted in the following order:

1. `BOT` (beginning of transmission) characters. The default `BOT` transmission is no characters.
2. Asterisk (\*) immediately preceding the response text.
3. Response text.
4. `EOT` (end of transmission) characters. The default `EOT` transmission is a carriage return. If the response comprises more than one line (e.g., `TASF` has a multi-line response), the `EOL` (end of line) characters are placed at the end of each line, except for the last line—`EOT` characters are always placed at the end of the last line. The default `EOL` characters are carriage return and line feed.
5. `ERROK` characters. Default characters are: carriage return, line feed, “>” prompt, and space.

## Customizing the Gem6K response:

Using the factory default error level, ERRLVL4, the Gem6K uses all available response characters. The ERRLVL command has additional options which allow you to include or exclude certain elements of the response (refer also to the table below):

- BOT, EOL, and EOT characters (these characters are always transmitted when no error is detected).
- An asterisk (\*), transmitted at the beginning of each line of the Gem6K's response (e.g., \*ERRLVL4).
- The query or status command that was entered (echoed back as a confirmation). For example, to check the error level setting, type in the ERRLVL command with no parameters; the Gem6K responds with \*ERRLVL4 if it is set to error level 4.
- ERROK character(s) transmitted when no error is detected, and whenever you press the ENTER key.
- ERRBAD character(s) transmitted when an error is detected. Default is the "?" character.
- Error message (see page 11).

	Characters when no error is detected						Characters when an error is detected				
	BOT	EOT	EOL	*	Command	ERROK	BOT	EOT	EOL	ERRBAD	Error Msg.
ERRLVL4	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
ERRLVL3	✓	✓	✓	✓	✓	✓			✓		
ERRLVL2	✓	✓	✓	✓	✓						
ERRLVL1	✓	✓	✓	✓							
ERRLVL0 *	✓	✓	✓								

## Example:

The examples below show the Gem6K's response to the TAS status command under all five ERRLVL settings. The factory default BOT, EOT, EOL, and ERROK characters (noted above) are used. The printed characters are shown in **bold text**, and the non-printing characters are represented as follows:

- Carriage return ..... <cr>
- Line feed ..... <lf>
- Space ..... <sp>

### ◆ ERRLVL4 and ERRLVL3 (no difference between these two settings):

```
*TAS0000_0000_0000_1000_0000_0001_0000_0000<cr><lf>
* 0000_0000_0000_1000_0000_0001_0000_0000<cr>
<cr><lf>
> <sp>
```

### ◆ ERRLVL2:

```
*TAS0000_0000_0000_1000_0000_0001_0000_0000<cr><lf>
* 0000_0000_0000_1000_0000_0001_0000_0000<cr>
```

### ◆ ERRLVL1:

```
*0000_0000_0000_1000_0000_0001_0000_0000<cr><lf>
*0000_0000_0000_1000_0000_0001_0000_0000<cr>
```

### ◆ ERRLVL0:

```
0000_0000_0000_1000_0000_0001_0000_0000<cr><lf>
0000_0000_0000_1000_0000_0001_0000_0000<cr>
```





Bit #	Function (Error bits 13, 15, 17, 19 – 22, and 25 – 32 are reserved.)	Branch Type
1*	Stall Detected: Functions when stall detection is enabled with <code>ESTALL</code> (encoder stall detection) or <code>DSTALL</code> (drive stall detection), and Kill on Stall is enabled ( <code>ESK1</code> ).	GOSUB
2	Hard Limit Hit: When an input is defined as an end-of-travel input ( <code>LIMFNCi-R</code> or <code>INFNCi-R</code> ; <code>LIMFNCi-S</code> or <code>INFNCi-S</code> ), and that input becomes active. Hard limits must be enabled ( <code>LH3</code> ).	GOTO if <code>COMEXL0</code> ; GOSUB if <code>COMEXL1</code>
3	Soft Limit Hit: Functions when soft limits are enabled ( <code>LS3</code> ).	GOTO if <code>COMEXL0</code> ; GOSUB if <code>COMEXL1</code>
4	Drive Fault: Detected only if the drive is enabled ( <code>DRIVE1</code> ).	GOTO
5	Commanded Kill or Commanded Stop (a <code>K</code> , <code>!K</code> , <code>&lt;ctrl&gt;K</code> , <code>S</code> , or <code>!S</code> command is sent).	<code>!K</code> = GOTO; <code>!S</code> = GOTO if <code>COMEXS0</code> ; <code>!S</code> = GOSUB if <code>COMEXS1</code> , but need <code>!C</code>
	<b>NOTE</b> If you want the program to stop, you must issue the <code>!HALT</code> command.	
6	Kill Input: When an input is defined as a KILL input ( <code>INFNCi-C</code> ), and that input becomes active.	GOTO
7	User Fault Input: When an input is defined as a user fault input ( <code>INFNCi-F</code> ), and that input becomes active.	GOTO
8	Stop Input: When an input is defined as a stop input ( <code>INFNCi-D</code> ), and that input becomes active.	GOTO
9	Enable input is activated (not grounded). See also <code>TINO</code> bit #6.	GOTO
10	Pre-emptive (on-the-fly) <code>GO</code> or registration move profile not possible at the time of attempted execution.	GOSUB
11**	Target Zone Settling Timeout Period (set with the <code>STRGTT</code> command) is exceeded.	GOSUB
12**	Maximum Position Error (set with the <code>SMPER</code> command) is exceeded.	GOSUB
14	<code>GOWHEN</code> condition already true when a subsequent <code>GO</code> , <code>FSHFC</code> , or <code>FSHFD</code> command is executed.	GOSUB
16	Bad command detected (bit is cleared with <code>TCMDER</code> command).	GOSUB
18	Cable to an expansion I/O brick is disconnected, or power to the I/O brick is lost; error is cleared by reconnecting the I/O brick (or restore power to the I/O brick) and issuing the <code>ERROR.18-0</code> command and then the <code>ERROR.18-1</code> command. (Multi-tasking: This condition also kills all tasks.)	GOTO if <code>KIOEN1</code> GOSUB if <code>KIOEN0</code>
23	Ethernet Client connection error.	GOSUB
24	Ethernet Client polling error.	GOSUB

\* Stepper axes only; \*\* Servo axes only

**Example:** Refer to the Error Program Assignment (`ERRORP`) command example.

## ERRORL Error Log Selection

Type	Error Handling	<b>Product</b>	<b>Rev</b>
Syntax	<code>&lt;a_&gt;&lt;!&gt;&lt;%&gt;ERRORL&lt;b&gt;&lt;b&gt;...&lt;b&gt;&lt;b&gt; (32 bits)</code>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	<code>b = 0</code> (disable), <code>1</code> (enable), or <code>X</code> (don't change)		
Default	0		
Response	ERRORL: *ERRORL0000_0000_0000_0000_0000_0000_0000_0000 <div style="display: flex; justify-content: space-between; width: 100%;"> <span>bit 1 </span> <span> bit</span> </div> 32		
See Also	[ <code>ASX</code> ], <code>CERRLG</code> , <code>DSTALL</code> , [ <code>ER</code> ], <code>ERROR</code> , <code>ESTALL</code> , <code>GOWHEN</code> , <code>INFNC</code> , <code>K</code> , <code>KIOEN</code> , <code>LH</code> , <code>LS</code> , <code>S</code> , <code>TASX</code> , <code>TER</code> , <code>TERF</code> , <code>TERRLG</code> , <code>TRGFN</code>		

Use the `ERRORL` command to choose the conditions that will be included in the error log. When an error log bit is enabled (`ERRORL11...11`), the operating system will respond to a specific execution error by making an entry in the error log. Each bit corresponds to a different error condition (see table below). To enable or disable a specific bit, the syntax is `ERRORL.n-b`, where “n” is the error bit number and “b” is either 1 to enable or 0 to disable.

Use the `TERRLG` command to view the error log. Use the `CERRLG` command to clear the error log.

**NOTE:** After an error event occurs and the drive updates the error log, bits may still be recorded in `TAS` or `TASX` for another 120 milliseconds. To find all possible causes of an error event, examine `TAS` and `TASX` in addition to viewing the error log.

Bit #	Function
1*	Stall Detected: Functions when stall detection is enabled with <code>ESTALL</code> (encoder stall detection) or <code>DSTALL</code> (drive stall detection), and Kill on Stall is enabled ( <code>ESK1</code> ).
2	Hard Limit Hit: When an input is defined as an end-of-travel input ( <code>LIMFNCi-R</code> or <code>INFNCi-R</code> ; <code>LIMFNCi-S</code> or <code>INFNCi-S</code> ), and that input becomes active. Hard limits must be enabled ( <code>LH3</code> ).
3	Soft Limit Hit: Functions when soft limits are enabled ( <code>LS3</code> ).
4	Drive Fault: Detected only if the drive is enabled ( <code>DRIVE1</code> ).
5	RESERVED
6	RESERVED
7	User Fault Input: When an input is defined as a user fault input ( <code>INFNCi-F</code> ), and that input becomes active.
8	RESERVED
9	RESERVED
10	Pre-emptive (on-the-fly) <code>GO</code> or registration move profile not possible at the time of attempted execution.
11**	Target Zone Settling Timeout Period (set with the <code>STRGTT</code> command) is exceeded.
12**	Maximum Position Error (set with the <code>SMPER</code> command) is exceeded.
13	RESERVED
14	<code>GOWHEN</code> condition already true when a subsequent <code>GO</code> , <code>FSHFC</code> , or <code>FSHFD</code> command is executed.
15-32	RESERVED

\* Stepper axes only; \*\* Servo axes only

**NOTE:** `ERRORL` bits 5, 6, 8, 9, 13, 15 – 32 are reserved.

ERRORP		Error Program Assignment	
Type	Error Handling	Product	Rev
Syntax	<a_><!><%>ERRORP<t>	GT6K	6.0
Units	t = text (name of error program)	GV6K	6.0
Range	Text name of 6 characters or less		
Default	n/a		
Response	ERRORP: *ERRORPerr1		
See Also	[ ER ], <code>ERRLVL</code> , <code>ERROR</code> , <code>TER</code>		

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. If you later decide not to have an error program, issue the `ERRORP CLR` command; after the `ERRORP CLR` command, no error program will be called until you assign a new one.

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, 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=b10X)`). An error program set-up example is provided in the *Programmer's Guide*.

When an error condition occurs and the associated error-checking bit has been enabled with the `ERROR` command, the Gem6K will branch to the error program. Depending on the error condition, the branch may 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.

#### MULTI-TASKING

If you are operating multiple tasks, be aware that you must enable error conditions (`ERROR`) and specify an error program (`ERRORP`) for each task (e.g., `2%ERROR.2-1` and `2%ERRORP.FIX` for Task 2). Each task has its own error status register (reported with `ER`, `TER`, and `TERF`).

The `ERRORP` assignment is not saved in battery-backed RAM. To ensure that the `ERRORP` assignment is retained when you cycle power or issue a `RESET` command, put the `ERRORP` command in the *startup* program assigned with the `STARTP` command.

#### 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 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, enable input activated, in which case the error program is called only once.) There are three ways to cancel the branch:

- Disable the error-checking bit with the `ERROR.n-0` 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.6-0` command. To re-enable the error-checking bit, issue the `ERROR.n-1` command.
- 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 <code>ERRORP</code>	How to Remedy the Error
1	Stepper axes only: Stall detected (Stall Detection and Kill On Stall must be enabled first—see <code>ESTALL</code> or <code>DSTALL</code> , respectively—and <code>ESK1</code> enabled)	Gosub	Issue a <code>GO</code> command.
2	Hard Limit Hit. An input defined as an end-of-travel input ( <code>LIMFNC</code> or <code>INFNC</code> ) became active. Hard limits must be enabled (see <code>LH</code> ).	If <code>COMEXL0</code> , then Goto; If <code>COMEXL1</code> , then Gosub	Change direction & issue <code>GO</code> command; or issue <code>LH0</code> .
3	Soft Limit Hit (soft limits must be enabled first—see <code>LS</code> )	If <code>COMEXL0</code> , then Goto; If <code>COMEXL1</code> , then Gosub	Change direction & issue <code>GO</code> command; or issue <code>LS0</code> .
4	Drive Fault (Detected only if drive enabled – <code>DRIVE1</code> ).	Goto	Check the cause with <code>TASX</code> , remedy the fault condition, then issue a <code>DRIVE1</code> command.

5	Commanded Stop or Kill (whenever a K, !K, <ctrl>K, S, or !S command is sent) — See “Commanded Kill or Stop” note below.	If !K, then Goto; If !S & COMEXS0, then Goto; If !S & COMEXS1, then Gosub, but need !C	No fault condition is present—there is no error to clear <div style="border: 1px solid black; padding: 2px; width: fit-content;">If you want the program to stop, you must issue the !HALT command.</div>
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 INFNC function.
8	Stop Input Activated (see INFNCi-D)	Goto	Deactivate the stop input, or disable it by assigning it a different INFNC function.
9	Enable input not grounded	Goto	Re-ground the enable input, and issue a DRIVE1 command.
10	Pre-emptive (on-the-fly) GO or registration move profile not possible at the time of attempted execution.	Gosub	Issue another GO command.
11	GV6K Only: Target Zone Timeout (STRGTT value has been exceeded).	Gosub	Issue these commands in this order: STRGTE0, D0, GO, STRGTE1
12	GV6K Only: Exceeded Max. Allowable Position Error (set with the SMPER command).	Gosub	Issue a DRIVE1 command. Verify that feedback device is working properly.
14	GOWHEN condition already true when GO, FSHFC, or FSHFD executed.	Goto	Issue another GOWHEN command; or issue a !K command and check the program logic (use the TRACE and STEP features if necessary).
16	Bad command detected.	Gosub	Issue the TCMDEP command.
18	Expansion I/O brick disconnected, or lost power. (Multi-tasking: This condition also kills all tasks.)	Goto if KIOEN1 Gosub if KIOENO	Reconnect I/O brick or restore power. Then issue ERROR.18-0 and then ERROR.18-1.
23	Ethernet Client connection error.	Gosub	Clear the error bit (ERROR.23-0), then re-establish the Ethernet connection, and then issue ERROR.23-1.
24	Ethernet Client polling failed.	Gosub	Clear the error bit (ERROR.24-0), then re-establish the Ethernet connection, and then issue ERROR.24-1.

**Reserved Bits:** Bits 13, 15, 17, 19-22, and 25-32 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.

**Commanded Kill or Stop:** When ERROR bit 5 is enabled (ERROR.5-1), a Stop (S or !S) or a Kill (K, !K or <ctrl>K) command will cause the Gem6K to branch to the error program. Note, however, that this error condition does not set an error bit (ER), because there is no way to clear the error condition upon leaving the error program. Therefore, you should use the `IF (ER=b00000000000000000000000000000000)`

statement in your error program to determine if the cause of the error was a commanded kill or stop (i.e., if no error bits are set).

**Example:**

```

DEF err1      ; Define error program err1
IF(ER=b01)   ; 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=b0X1)  ; If error is soft limit, back off soft limit, reset position, & continue
D~           ; Change direction in preparation to back off the soft limit
D1           ; Set distance to 1 step (just far enough to back off the soft limit)
GO1          ; Initiate the 1-step move to back off the soft limit
PSET0        ; 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 hitting a hard or soft limit
ERROR01100000 ; Set error condition bits to look for hard limit or a soft limit

```

---

**ESDB Stall Backlash Deadband**

Type	Encoder Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>ESDB<i>	GT6K	6.0
Units	i = motor steps	GV6K	n/a
Range	0 - 99,999,999		
Default	0		
Response	ESDB: *ESDB0		(applicable only to stepper axes)
See Also	[ AS ], DRES, ERES, ESK, ESTALL, TAS		

The ESDB command establishes the number of motor steps that a move will travel 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. To use the stall backlash deadband, encoder-based 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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>ESK<b>	GT6K	6.0
Units	b = enable bit	GV6K	n/a
Range	b = 0 (disable) or 1 (enable)		
Default	0		
Response	ESK: *ESK0		(applicable only to stepper axes)
See Also	DRES, DRIVE, DSTALL, ERES, ESDB, ESTALL, TAS, TASX, TER		

The ESK command enables/disables the Kill-on-Stall feature. With Kill-on-Stall enabled, if the Gem6K detects an encoder stall (ESTALL) or a drive stall (DSTALL), it will immediately stops pulses from being sent to the motor. Stall detection must be enabled (with the ESTALL command or the DSTALL command) before the ESK command will have any affect.

For further details and programming examples on *Encoder Stall Detection*, refer to ESTALL.

For further details and programming examples on *Drive Stall Detection*, refer to DSTALL.

---

## ESTALL Encoder-Based Stall Detection

Type	Encoder Configuration	Product	Rev
Syntax	<a_><!>ESTALL<b>	GT6K	6.0
Units	n/a	GV6K	n/a
Range	b = 0 (disable) or 1 (enable)		
Default	0		(applicable only to stepper axes)
Response	ESTALL: *ESTALL0		
See Also	[ AS ], DRES, DSTALL, ENCCNT, [ ER ], ERES, ESDB, ESK, TAS, TER		

---

The ESTALL command determines if encoder-based stall conditions will be checked. (For information about *Drive Stall Detection*; which does not require an encoder, refer to the DSTALL command.)

**NOTE:** Encoder count reference must be enabled (ENCCNT1) before encoder stall detect (ESTALL) can be used.

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.

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 drive checks to see if it received 40 encoder counts. If it did, then everything is O.K. If not, then a stall condition exists.

To accurately detect a stall, the drive resolution (DRES) and the encoder resolution (ERES) must be properly set.

When a stall is detected, the GT6K responds as follows:

- The stall is reported with Axis Status bit #12 (reported with TAS, TASF, and AS).
- The stall is reported with Extended Axis Status bit #17 (reported with TASX, TASXF, and ASX).
- If error-checking bit #1 is enabled (ERROR.1-1):
  - The stall is reported with Error Status bit #1 (reported with TER, TERF, and ER).
  - The GT6K branches to the assigned ERRORP program.
- If the Kill-on-Stall feature is enabled (ESK1), the GT6K immediately stops pulses from being sent to the motor. If Disable Drive on Kill is enabled (KDRIVE1) it disables the drive.

### Example:

```
SCALE0          ; Disable scaling
DEL proga       ; Delete program called proga
DEF proga       ; Begin definition of program called proga
DRES25000       ; Drive resolution set to 25000 steps/rev
ERES4000        ; Encoder resolution is 4000 post-quadrature counts/rev
ESDB10          ; Stall backlash set to 10 commanded counts
ENCCNT1         ; Enable encoder count reference mode
ESTALL1         ; Enable stall detection
ESK1            ; Enable kill on stall
MA0             ; Incremental positioning mode
MCO             ; Preset positioning mode
A10             ; Set the acceleration to 10 units/sec/sec
V1              ; Set the velocity to 1 unit/sec
D100000         ; Set the distance to 100000 units
GO1             ; Initiate motion
                ; The motor will move 100000 commanded counts (4 revs)
                ; (If, at any time during the move the actual encoder count falls
                ; behind, a stall condition will be flagged, and motion will stop.)
END             ; End definition of proga
```

---

## EXE Execute a Program From a Compiled Program

Type	PLC Scan Program	<b>Product</b>	<b>Rev</b>
Syntax	i%EXEt	GT6K	6.0
Units	i = Task Number t = Program Name (6 characters or less)	GV6K	6.0
Range	i = 1-10		
Default	n/a		
Response	n/a		
See Also	INSELP, PCOMP, PEXE, PLCP, SCANP		

---

Use the EXE command to start a standard (non-compiled) program from within a compiled PLCP program. The EXE command specifies the name of the program, and the task in which it will be launched. The program named in the EXE command need not be defined at the time the PLCP program is compiled; however, the program must be defined before the SCANP or PRUN is issued. If no task number is assigned with a % prefix, then the task in which the PLCP program is compiled (PCOMP) will be the task that runs the program. Note, however, that the EXE program cannot be executed in the Task Supervisor (task 0).

When a PLCP program performs an EXE command, SS bit 28 ("PROGRAM PENDING") will be set in the task specified by EXE command. When background task swapping allows the program to actually start running, SS.28 will be cleared and SS bit 3 ("PROGRAM EXECUTING") will be set. If a PLCP program performs an EXE command with either of these bits set, the new program will not run, and SS bit 32 ("EXE FAILED") will be set. SS.32 will be cleared upon the next successful EXE directed to that task. Because the PLCP program ignores the EXE command if a currently running program is detected within the specified task, the EXE command can only be used to initiate a new program if the task is not already running a program.

Like the INSELP command, the program launched by the EXE command will not interrupt a currently running program, nor will it interrupt a WAIT or T command.

**CAUTION:** Using the SCANP command to run a PLCP program in Scan mode will cause the PLCP program to execute as often as every system update period (2 ms). An EXE command used within a PLCP program running in Scan mode could therefore attempt to launch a program in the specified task as often as every 2 ms. This may not allow enough time for the program launched in the specified task by the EXE command to complete before the same EXE command is issued again. As stated, the PLCP program will ignore the EXE command if a currently running program is detected, so timing must be considered when launching programs with the EXE command.

To execute a compiled program from within a compiled PLCP program, use the PEXE command.

### Example:

```
DEF PLCP1          ; Define PLC program PLCP1
IF(IN.1=b1)        ; If input 1 is active
3%EXE PROG1        ; Launch program PROG1 in Task 3
ELSE
2%EXE PROG2        ; Otherwise launch program PROG2 in Task 2
NIF
END

PCOMP PLCP1        ; Compile PLCP1
SCANP PLCP1        ; Scan with program PLCP1
```



<b>[ FB ] Value of Current Feedback Device</b>		<b>Product</b>	<b>Rev</b>
Type	Servo; Assignment or Comparison		
Syntax	See below	GT6K	6.0
Units	See below	GV6K	6.0
Range	See below		
Default	n/a		
Response	n/a		
See Also	GOWHEN, [ PE ], PSET, SCALE, SCLD, SFB, TFB,		

Use the FB operator to assign the value of the selected feedback device to a variable or to make a comparison. Depending on the configuration of the SFB command, the feedback device could be an encoder or a resolver.

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 position values returned will be encoder or resolver counts. If scaling is enabled (SCALE1), the encoder or resolver values will be scaled by the SCLD value. For more information on scaling, refer to page 16.

**Syntax:** VARn=FB where n is the variable number, or FB can be used in an expression such as IF(FB<6).

**Example:**

```
SFB1           ; Feedback is encoder
VAR6=FB       ; Assign position (scalable) of encoder to variable #6
IF(FB<500)    ; If position (scalable) of encoder is less than 500,
              ; do the commands following the IF statement until the NIF command
VAR4=FB+1000 ; Set variable #4 equal to current position of encoder plus 1,000
NIF          ; End of IF statement
```

<b>FFILT Following Filter</b>		<b>Product</b>	<b>Rev</b>
Type	Following		
Syntax	<a_><!>FFILT<i>	GT6K	6.0
Units	i = filtering level	GV6K	6.0
Range	i = 0, 1, 2, 3, or 4		
Default	0		
Response	FFILT *FFILT0		
See Also	FMAXA, FMAXV, FPPEN		

The FFILT command specifies the bandwidth of the low pass filter applied to the measurements of master position. This command is to be used in these situations:

- Measurement of master position is contaminated by either electrical noise (when analog input is the master) or mechanical vibration.
- Measurement noise is minimal, but the motion that occurs on the master input is oscillatory. In this case, using the filter can prevent the oscillatory signal from propagating into the follower axis (i.e., ensuring smoother motion on the follower axis).

The table below shows how the value of the FFILT command specifies the low pass filter's bandwidth:

FFILT Setting	Low pass Filter Bandwidth
0	$\infty$ (no filtering) – <i>default setting</i>
1	120 Hz
2	80 Hz
3	50 Hz
4	20 Hz

**Example:**

```
FFILT1       ; Set filtering bandwidth to 120 Hz
```

---

## FGADV Following Geared Advance

Type	Following	Product	Rev
Syntax	<a_><!>FGADV<r>	GT6K	6.0
Units	r = advance distance (scalable)	GV6K	6.0
Range	0.00000-999,999,999 (scalable with SCLD)		
Default	n/a		
Response	n/a		
See Also	FOLMD, FOLRD, FOLRN, [ FS ], FSHFD, GOWHEN, SCLD, TFS		

---

The FGADV command provides the ability to superimpose an advance or retard on Following motion. This is the same ability provided by the FSHFD command, except that the superimposed motion is also geared to master motion. The FGADV command has the positive or negative “advance” distance as a parameter, but it initiates motion instead of simply setting up the distance. The shape of the superimposed profile is determined by the FOLMD, FOLRN, and FOLRD commands (just as a normal preset Following move).

The FGADV command profile may be delayed with the GOWHEN command.

A FGADV move may be performed only while the conditions below exist (Following status bit #23, reported with the FS, TFS, and TFSF commands, indicates that it is “OK to do FGADV move”):

- Master is specified with a FOLMAS command
- Following is enabled with the FOLEN command
- The follower axis is either not moving, or moving at constant ratio in continuous mode (MC1)

A FGADV move may not be performed:

- During a preset (MC0) move
- In a compiled profile or program

Following Status (FS, TFS, and TFSF) bit #24 reports if a “FGADV move is underway”.

### Example:

```
COMEXC1          ; All command processing during motion
FOLRN25          ; Set numerator of follower-to-master Following ratio
FOLRD10          ; Set denominator of follower-to-master Following ratio
FOLMD1000       ; Set master distance to 1000 units
MC1              ; Enable continuous positioning mode
D+              ; Set direction to positive
FOLEN1          ; Enable Following
GO               ; Ramp up to a 2.5 to 1 ratio over 1000 master distance units
FOLMD500        ; Set master distance to 500 units
FOLRN13         ; Superimposed ratio will be 1.3 (added to 2.5 = 3.8 total)
WAIT(FS.23=B1)  ; Wait for OK to do geared advance
                 ; (in this case, ramp is complete)
FGAVD400        ; Advance the follower axis 400 counts over a distance
                 ; of 500 master counts
WAIT (FS.23=B1) ; Wait for OK to do geared advance
                 ; (in this case, FGADV400 superimposed profile is complete)
FGADV-400       ; Retard the follower axis 400 counts over a distance of
                 ; 500 master counts (2.5 - 1.3 = 1.2 net ratio)
```

---

## FLTDSB Fault on Drive Disable (DRIVE0)

Type	Drive Configuration	Product	Rev
Syntax	<a_><!>FLTDSB<b>	GT6K	6.0
Units	b = enable bit	GV6K	6.0
Range	0 (disable) or 1 (enable)		
Default	1		
Response	FLTDSB: *FLTDSB1		
See Also	DRIVE, TAS, TASX, TER		

---

Use the FLTDSB command to enable/disable the Fault on Drive Disable mode. If Fault on Drive Disable is enabled (FLTDSB1 – the default setting), and the drive is disabled via the DRIVE0 command or the “Enable” input (pin 1 on the DRIVE I/O connector), any output configured as a fault output is activated and the dry contact relay (labeled “RELAY COM” and “RELAY N.O.” on the 4-pin removable connector) is opened.

---

## **FMAXA**      **Follower Axis Maximum Acceleration**

Type	Following	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>FMAXA<r>	GT6K	6.0
Units	r = units/sec/sec	GV6K	n/a
Range	r = 0.00001 - 39,999,998 (scalable with SCLA)		
Default	0.00 (no limit imposed)		(applicable only to stepper axes)
Response	FMAXA    *FMAXA0.0000		
See Also	[ FS ], FFILT, FMAXV, FPPEN, SCLA, TFS, TFSF		

---

The FMAXA command sets the maximum acceleration for follower axes. The FMAXA command is scaled by the SCLA parameter.

As part of a ramp to new ratio, or simply following an accelerating master at constant ratio, a follower may be required to accelerate. If the required acceleration is larger than FMAXA, the follower will begin falling behind its commanded position. The Gem6K drive will attempt to make up this position error as soon as the commanded accel falls below FMAXA. In controlling stepper motors, the Gem6K adds an error correction velocity to that implied by the commanded ratio.

As with FMAXV, FMAXA should be determined and defined early in the development stage of an application to prevent any damage to the load on the follower axis when unexpectedly high accelerations are commanded. The torque available from the follower motor will also be a determining factor in this parameter in order to prevent motor stalls.

Following status bit #26 (see TAS, TASF, FS) is set while FMAXA or FMAXV is limiting the present Following profile.

**Example:**

```
FMAXA75            ;Set maximum follower acceleration to 75 user units.
```

---

## **FMAXV**      **Follower Axis Maximum Velocity**

Type	Following	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>FMAXV<r>	GT6K	6.0
Units	r = units/sec (scalable with SCLV)	GV6K	n/a
Range	r = 0.000000-1600000.000000		
Default	0.00 (no limit imposed)		(applicable only to stepper axes)
Response	FMAXV    *FMAXV0.0000		
See Also	[ FS ], FFILT, FMAXA, FPPEN, SCLV, TFS, TFSF		

---

The FMAXV command sets the maximum velocity at which follower axes may travel. The FMAXV command accepts numeric variables (VAR) as an argument and is scaled by the SCLV parameter.

Normally in a Following application, the follower velocities will be known based on the normal speed of the master and the commanded Following ratios (FOLRN and FOLRD). In some cases, however, the master speed may be higher than normal, the follower may be commanded to perform a shift move, or some other event may occur which will cause the follower to travel at a velocity higher than expected. In these cases, the Gem6K drive will increase the speed of the follower as necessary to perform the required move, but only up to the FMAXV value.

If the commanded speed is higher than FMAXV, the follower axis will start falling behind its commanded position. The Gem6K drive will attempt to make up this position error as soon as the commanded speed falls below FMAXV. In controlling stepper motors, the Gem6K automatically adds an error correction velocity to that implied by the commanded ratio.

The FMAXV value should be determined and defined early in the development stage of an application to prevent any damage to the load on the follower axis when unexpectedly high velocities are commanded.

Following status bit #26 (see TAS, TASF, FS) is set while FMAXA or FMAXV is limiting the present Following profile.

**Example:**

```
FMAXV15            ;Set the follower maximum velocity to 15 user units
```

---

## FMCLLEN Master Cycle Length

Type	Following	Product	Rev
Syntax	<a_><!>FMCLLEN<r>	GT6K	6.0
Units	r = master distance units (scalable)	GV6K	6.0
Range	r = 0-999,999,999 (scalable with SCLMAS)		
Default	0		
Response	FMCLLEN *FMCLLEN0		
See Also	FMCNEW, FMCP, FOLEN, [ FS ], GOWHEN, [ PMAS ], SCLMAS, TFS, TPMAS, WAIT		

---

The FMCLLEN command defines the length of the master cycle in user units. This value is scaled by the SCLMAS parameter. Numeric variables (VAR) can be used with this command. The initial value for FMCLLEN is zero (FMCLLEN0), which means that the default master cycle length is the maximum internal size (4,294,967,296).

The concept of a master cycle may be useful when moves or other events must be initiated at certain master positions in a repetitive cycle. By specifying a master cycle length, periodic actions may be programmed in a loop or with subroutines which refer to cycle positions, even if the master runs continuously. You can program the Gem6K drive to suspend program operation or delay moves until specified master cycle positions. The master cycle length, FMCLLEN, should be defined before the functions which wait for periodic master cycle positions are used. An axis need not be in Following mode (FOLEN1) to utilize the concept of a master cycle. However, **master positions will not be measured until a master has been assigned with the FOLMAS command.**

**Example** (refer also to FOLEN example #2):

```
SCLMAS4000      ; Set the master scale factor to 4000
FMCLLEN3       ; Set master cycle length to 3 user units
```

---

## FMCNEW Restart Master Cycle Counting

Type	Following	Product	Rev
Syntax	<a_><!>FMCNEW<b>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	b = 0 (do not restart) or 1 (restart immediately)		
Default	n/a		
Response	n/a		
See Also	FMCLLEN, FMCP, GOWHEN, [ NMCY ], [ PMAS ], TPMAS, TRGFNCx1, WAIT		

---

The FMCNEW1 command restarts master cycle counting. This sets the master cycle position (PMAS) to the value most recently specified with FMCP, and sets the master cycle number (NMCY) to zero. The master cycle position and the master cycle number are set immediately, and program flow continues normally.

The function of the FMCNEW1 command can be initiated with a trigger input by specifying a TRGFNCx1 command. If the FMCNEW1 command is used, master cycle counting is restarted immediately; if TRGFNCx1 is used, the Gem6K drive will record the instruction to set the master cycle position when the specified trigger occurs. In this case, the master cycle counting is restarted when the specified trigger is activated, even though commands continue to execute and the master cycle counting continues.

FMCNEW0 or FMCNEW1 will remove the status of master cycle restart pending a trigger input (TRGFNCx1). In the case of FMCNEW0, no restart will occur, and the specified trigger will not cause a new cycle restart.

A new cycle automatically occurs (i.e., the master cycle position is set to zero, not the FMCP value), when the master cycle length (FMCLLEN) is reached, even if no FMCNEW command has been executed.

**Example:**

```
TPMAS          ; Display master position: response is *TPMAS12.2
FMCNEW1       ; Start new master cycle
TPMAS          ; Display master position: response is *0
```

---

## FMCP

### Initial Master Cycle Position

Type	Following	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>FMCP<r>	GT6K	6.0
Units	r = master position in scalable steps	GV6K	6.0
Range	r = ±999,999,999 (scalable with SCLMAS)		
Default	0		
Response	FMCP *FMCP+0		
See Also	FMCNEW, FOLMAS, [ FS ], GOWHEN, SCLMAS, TFS, WAIT		

---

The FMCP command defines the initial master cycle position in user units. The initial master cycle position is assigned as the current master cycle position each time master cycle counting is restarted with the FMCNEW or TRGFNCx1 command. This value is scaled by the SCLMAS parameter. Numeric variables (VAR) can be used with this command. The default value for FMCP is zero (FMCPØ), which means that the master cycle position will be zero when master cycle counting is restarted (see FMCNEW).

The concept of an initial master cycle position may be useful if new master cycle position counting must be restarted at a master position which is different from what needs to be considered the “zero position” of a periodic cycle. The initial position defined with FMCP applies to the first cycle only. When a master cycle is complete, the master cycle position rolls over to zero. A negative value would be used if some master travel were desired before master cycle position was zero. A positive value would be used if it was necessary to enter the first master cycle at a position greater than zero.

For example, suppose FMCLEN was set to 20 and FMCP was set to 7. When master cycle position counting is restarted, either via FMCNEW1 or the specified trigger (TRGFNCx1), the initial master cycle position will be 7. Rollover will occur after the master travels 13 more units, and the master cycle position would go to zero.

**Example:**

```
FMCP-2          ; Set the initial master cycle position to -2
```

---

## FOLEN

### Following Mode Enable

Type	Following	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>FOLEN<b>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	b = 0 (disable) or 1 (enable)		
Default	0		
Response	FOLEN: *FOLEN0		
See Also	FGADV, FOLK, FOLMAS, FOLRD, FOLRN, FOLRNF, [ FS ], FSHFC, FSHFD, GOWHEN, JOG, JOY, TFS		

---

The FOLEN command indicates whether subsequent moves on the specified axes will be following a master (FOLEN1) or normal time-based moves (FOLEN0). The term *Following mode* means that FOLEN1 has been given, and that the motion of the follower is dependent on the motion of the master at all times. If FOLEN0 is given, the motion of the master is still monitored, but the motion of the follower is independent of the master.

**To move in the Following mode, the master must be previously specified with the FOLMAS command.**

Enabling the Following mode (FOLEN1) will set the net position shift value (reported by TPSHF and PSHF) to zero. This is true even if the follower is already in Following mode.

S-Curve profiling is not operational during Following moves.

<b>RESTRICTIONS ON USING FOLEN</b>
------------------------------------

The FOLEN command may not be executed during certain conditions (results in the error message “NOT VALID DURING RAMP”).

- You may not enable Following (FOLEN1) on an axis that is in motion, waiting for a GOWHEN condition, or operating in the Joystick mode (JOY1) or Jog mode (JOG1).
  - You may not disable Following (FOLEN0) on an axis that is in motion (unless moving at ratio in continuous mode, MC1, and not shifting) or waiting for a GOWHEN condition.
-

## FOLEN Examples

### Example #1:

The Gem6K drive is controlling a rotary motor, the master is a 1000-line incremental encoder mounted on the back of an externally controlled motor, and programming units are to be revs/second (rps).

#### Stepper Products:

The follower will start ramping to a ratio of 1:1 when trigger #1 (TRG-1A) goes active. This means the actual step ratio of follower to master is 25000 to 4000, or 6.25 follower steps for every master. After 25 master revolutions, the follower will decelerate to a 0.5:1 ratio (3.125 follower steps for every master). After a total of 75 master revolutions, the follower will ramp to zero ratio (i.e., stop) and repeat the cycle when trigger #1 is activated. All ramps to new ratios, including zero ratio, take place over one master revolution.

##### Scaling Set Up: (prior to defining program)

```
SCALE1          ; Enable scaling
SCLD25000       ; Set follower distance scale factor to 25,000 steps/rev
                ; (assumes a resolution of 25,000 steps/rev)
SCLMAS4000      ; Set master scale factor to 4000 steps/rev
```

#### Servo Products:

The follower will start ramping to a ratio of 1:1 when trigger #1 (TRG-1A) goes active. This means the actual step ratio of follower to master is 4000 to 4000, or 1 follower steps for every master. After 25 master revolutions, the follower will decelerate to a 0.5:1 ratio (0.5 follower steps for every master). After a total of 75 master revolutions, the follower will ramp to zero ratio (i.e., stop) and repeat the cycle when trigger #1 is activated. All ramps to new ratios, including zero ratio, take place over one master revolution.

##### Scaling Set Up: (prior to defining program)

```
SCALE1          ; Enable scaling
SCLD4000        ; Set follower distance scale factor to 4,000 steps/rev
                ; (assumes an encoder resolution of 4,000 steps/rev)
SCLMAS4000      ; Set master scale factor to 4000 steps/rev
```

The application program is defined as follows:

```
DEL FOLTST      ; Delete program called FOLTST
DEF FOLTST      ; Begin definition of program called FOLTST
INFNC1-H       ; Set input #1 (TRG-1A) to be "trigger interrupt" (used with GOWHEN later)
COMEXC1        ; Select continuous command processing mode
MCL            ; Select continuous positioning mode
FOLMAS1        ; Assign master encoder input as master
FOLMD1         ; Follower should change ratios over 1 master revolution
FMCLLEN100     ; Set master cycle length to 100 revs
FOLRD1         ; Set follower-to-master Following ratio denominator to 1
                ; (applies to all subsequent FOLRN commands)
FOLEN1         ; Enable Following
D+             ; Set motion to the positive- direction
$STRMV         ; Label to repeat move
TRGFNA1        ; Suspend execution of next move until trigger (TRG-1A) is active
TRGFNAx1       ; Begin new master cycle (counter at 0) when trigger (TRG-1A) is active
FOLRN1         ; Set follower-to-master Following ratio numerator to 1 (ratio set to 1:1)
GO1            ; Start continuous Following move (when TRG-1A is active)
WAIT(AS.26=b0 AND FS.4=b1) ; Wait for profile to actually start
                ; (when TRG-1A is active) and be at ratio
GOWHEN(PMAS>=25) ; Suspend execution of next move until master position >= 25
FOLRN0.5       ; Set Following ratio numerator to 0.5 (ratio set to 0.5:1)
GO1            ; Initiate new move according to new Following ratio
                ; (when master position >= 25)
WAIT(AS.26=b0 AND FS.4=b1) ; Wait for profile to actually start
                ; (when master position >= 25) and be at ratio
GOWHEN(PMAS>=75) ; Suspend execution of next move until master position >= 75
FOLRN0         ; Set Following ratio numerator to zero
                ; (ratio causes follower to ramp to stop)
GO1            ; Initiate new move with new Following ratio (when master pos. >= 75)
WAIT(AS.26=b0 AND FS.1=b0) ; Wait for profile to actually start
                ; (when master position >= 75) and the follower is not moving
JUMP STRMV     ; Repeat the cycle
END            ; End of program
```

## Example #2:

### Stepper Axes:

The master is an encoder mounted to gearing on a conveyor line. The gearing results in 16,000 encoder steps per conveyor inch. The follower, controlled by the Gem6K, is a 25,000 step/rev microstepper on a 36" long, 4-pitch leadscrew. The follower waits for the product to be sensed on the conveyor, accelerates to a 1-to-1 ratio, waits for a safe location to actuate the stamping equipment, then applies an inked stamp to the product at the correct location. After the stamp is placed, the follower quickly moves back to the starting position and waits for the next product. Note that this example illustrates how the WAIT command can be used to wait for master cycle positions in order to coordinate motion.

### Scaling Set Up: (prior to defining program)

```
SCALE1          ; Enable scaling
SCLA100000      ; Set accel scaling: 100,000 steps/inch
SCLV100000      ; Set velocity scaling: 100,000 steps/inch
SCLD100000      ; Set follower distance scaling: 100,000 steps/inch
SCLMAS16000     ; Set master scale factor to 16000 steps/inch to program in inches
```

### Servo Axes:

The master is an encoder mounted to gearing on a conveyor line. The gearing results in 16,000 encoder steps per conveyor inch. The follower is a 4,000 step/rev servo on a 36" long, 4-pitch leadscrew. The follower waits for the product to be sensed on the conveyor, accelerates to a 1-to-1 ratio, waits for a safe location to actuate the stamping equipment, then applies an inked stamp to the product at the correct location. After the stamp is placed, the follower quickly moves back to the starting position and waits for the next product. Note that this example illustrates how the WAIT command can be used to wait for master cycle positions in order to coordinate motion.

### Scaling Set Up: (prior to defining program)

```
SCALE1          ; Enable scaling
SCLA16000       ; Set accel scaling: 16,000 steps/inch
SCLV16000       ; Set velocity scaling: 16,000 steps/inch
SCLD16000       ; Set follower distance scaling: 16,000 steps/inch
SCLMAS16000     ; Set master scale factor to 16,000 steps/inch to program in inches
```

The application program is defined as follows:

```
DEF STAMPR      ; Begin definition of program called STAMPR
COMEXS1         ; Continue command execution after Stop
COMEXC1         ; Continue command execution during motion
SCALE1         ; Enable parameter scaling
1OUTFNC1-A      ; Configure onboard output #1 as a general-purpose prog. output
1INFNC2-H       ; Define TRG-1B as trigger interrupt (use as GOWHEN input)
A10            ; Acceleration = 10 inches/sec/sec
V5             ; Velocity = 5 inches/sec (non-Following moves)
MA1            ; Enable absolute positioning mode
FOLMAS1        ; Assign Master encoder input as master
FOLRN1         ; Set follower-to-master Following ratio numerator to 1
FOLRD1         ; Set follower-to-master Following ratio denominator to 1 (ratio is 1:1)
FOLMD1         ; Accel the follower over 1 master inch for Following moves
FMCLN40        ; Master cycle length is 40 inches
$INKON         ; Label to repeat inking process
FOLEN1         ; Enable Following on axis #1
1TRGFNBx1      ; Begin new master cycle when TRG-1B goes active
                ; (product sensed on conveyor)
1TRGFNB1       ; Start next move when TRG-1B is active
D+             ; Set to positive-direction
MC1           ; Select continuous positioning mode
GO1           ; Start continuous follower move on trigger #2
WAIT(PMAS>=10.5) ; Wait until master position is 10.5 inches - this is when the
                ; stamping device can be actuated without mechanical damage
                ; to the leadscrew assembly
1OUT.1-1       ; Turn on actuator (output #1) to place ink stamp on product
T.1           ; Wait for the ink stamp to be pressed in place by a
                ; stationary stamper
1OUT.1-0       ; Turn off actuator (output #1)
S1            ; Stop follower move
WAIT(1AS.1=b0) ; Wait until the axis is not moving
FOLENO        ; Disable Following
D0            ; Set distance (position) to zero
```

```

MCO          ; Select preset positioning mode
GO1          ; Move back to zero (the home position)
WAIT(MOV=b0) ; Wait until the axis is not moving
JUMP INKON   ; Begin cycle again on trigger #2
END          ; End of program

```

---

## FOLK      Following Kill

Type	Following	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>FOLK<b>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	b= 0 (disable) or 1 (enable)		
Default	0		
Response	FOLK      *FOLK0		
See Also	DRIVE, [ ER ], ERROR, FOLEN, FOLRD, FOLRN, FOLMAS, FOLMD, FSHFC, FSHFD, INFNC, K, [ PSHF ], SMPER, TER		

---

Under default operation (FOLK0), certain error conditions (i.e., drive fault input active, or max. position error limit exceeded) will cause the Gem6K to disable the drive and kill the Following profile (follower's commanded position loses synchronization with the master).

If you enable Following Kill (FOLK1), these error conditions will still disable the drive, but will not kill the Following profile. Because the Following profile is still running, the drive keeps track of what the follower's position should be in the Following trajectory. To resume Following operation, resolve the error condition (drive fault, excessive position error), enable the drive (DRIVE1), and command the drive to impose a shift to compensate for the lapse/shift that occurred while the drive was disabled and the follower was not moving. To impose the shift, assign the negative of the internally monitored shift value (PSHF) to a variable (e.g., VAR1 = -1 \* PSHF) and command the shift using a variable substitution in the FSHFD command (e.g., FSHFD(VAR1)).

The FOLK command only preserves Following profiles; normal velocity-based profiles will be killed regardless of the FOLK command.

---

## FOLMAS      Assignment of Master to Follower

Type	Following	<b>Product</b>	<b>Rev</b>
Syntax	<a_><±i>FOLMAS<±ii >	GT6K	6.0
Units	1st i = master #; 2nd i = master count source; ± sets direction of master counts relative to direction of actual master count source	GV6K	6.0
Range	1st i = 1-8 if 2nd i = 8; otherwise 1st i = 1 2nd i = 1 (encoder), 2 (analog input), 4 (commanded position), 5 (internal count source), 6 (internal sine wave source), 7 (RESERVED), or 8 (VARI variable). NOTE: "1", by itself, selects the master encoder. "0", by itself, disables the axis from being a follower		
Default	+0 (disable from being a follower axis)		
Response	FOLMAS      *FOLMAS+0		
See Also	ANIMAS, FGADV, FOLEN, FOLK, FOLMD, FOLRD, FOLRN, FOLRNF, [ FS ], FVMACC, FVMFRQ, SINAMP, SINANG, SINGO, TFS		

---

Use FOLMAS to assign or un-assign a master to a follower axis. The data field (±ii) configures the axis as a follower following the specified master count source:

Exceptions to the syntax:

- If a one (1) is placed in the data field (±ii), the axis will follow the counts from the Master Encoder (the separate encoder labeled "MASTER ENCODER").
- If a zero (0) is placed in the data field (±ii), the axis becomes a normal non-Following axis.



**Following a Virtual Master.** There are two “Virtual Master” options (an internal count source and an internal sine wave) for applications that require the synchronization features of Following, but have no external master. For a detailed description of virtual master features, see *Virtual Master* in the *Programmer’s Guide*.

- Master Source Option 5 (e.g., FOLMAS±15) selects the internal count source as master. The frequency and acceleration of the internal count source are established with the FVMACC and FVMFRQ commands, respectively.
- Master Source Option 6 (e.g., FOLMAS±16) selects the internal sine wave as master. The angle and amplitude of the sine wave are established with the SINANG and SINAMP commands, respectively. To start and stop the internal sine wave generator, use the SINGO command.

**Following an Integer Variable** (Option 8: syntax is FOLMASn8). This option allows an axis to follow the integer variable (VARI) specified with “n”; that is, VARIn. The range for “n” is 1-8 (VARI1 – VARI8). For example, FOLMAS48 assigns the axis to follow VARI4.

This option is particularly useful in conjunction with the INVARI (Map Inputs to a VARI Variable) feature. INVARI continuously updates a specified VARI variable with the value of a specified group of digital inputs, allowing the axis to follow a binary input pattern. Another useful way to update the value of the VARI variable is to calculate its value in a PLCP program (launched with the SCANP command).

The INVARI and SCANP options for updating VARI are good choices, because both are performed every system update, thus facilitating smooth Following motion. It is also possible to use an extra task (multi-tasking) to calculate VARI values, but the resulting updates will not be as fast (not perfectly periodic); consequently, Follow motion will be less smooth.

If scaling is enabled (SCALE1), the measurement of the master is scaled by the SCLMAS value. For more information on scaling, refer to page 16 or to the SCLMAS command description.

#### NOTES

- A follower axis cannot use its own feedback device or commanded position as the master input.
- Before you can use an analog input as a master count source, you must first use the ANIMAS command to assign the analog input to a master axis number. Then you can use the FOLMAS command to assign the analog input as a master counting source for the follower axis.

As an example, the FOLMAS+1 command sets up these parameters:

- Follower axis #1 is set up as follows (+1): the master encoder is assigned as the master to follower axis #1. The positive sign bit indicates that master counts will count in the same direction as the master encoder.

#### NOTE

The FOLMAS command configures an axis to be a follower, but *does not* automatically enable Following. To enable Following use the FOLEN1 command. To enable follower motion, enable Following (FOLEN1), issue a ratio (FOLRN and FOLRD), and issue the GO command.

As soon as the master is specified with the FOLMAS command, a continuously updated relationship is maintained between the follower’s position and the master’s position. Also, master velocity is continuously measured.

There are several applications in which a minus sign in the FOLMAS command is used. A minus sign should be used whenever the master is moving in the desired positive direction and yet the Gem6K drive actually perceives the master to be moving in the negative direction. For example, this can occur when the master input device is mounted on the opposite side of a conveyor. Putting a minus sign in front of the master

parameter specification in the FOLMAS command causes the incoming master signal to be negated before it is used by the follower. The term *master count* refers to the count after negation, if any.

For preset follower moves, the direction the follower travels depends on the mode of operation (absolute or incremental) and the commanded position. However, once a preset follower move is commanded, it will only start moving if the master is moving in the positive direction. This is true no matter the commanded direction of the follower move.

For continuous follower moves, the master count direction has a different effect. If the commanded move is positive in direction and the master is counting up, the actual follower travel direction will be positive. If the commanded move is positive in direction and the master is counting down, the actual follower travel direction will be negative. Similar cases exist for follower moves commanded in the negative direction.

**Example:** (refer to the FOLEN examples)

---

<b>FOLMD</b>		<b>Master Distance</b>	
Type	Following	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>FOLMD<r>	GT6K	6.0
Units	r = distance in counts	GV6K	6.0
Range	0 - 999,999,999 (scalable by SCLMAS)		
Default	0		
Response	FOLMD *FOLMD0		
See Also	ANIMAS, FGADV, FOLEN, FOLK, FOLRN, FOLRNF, FOLRD, MC, [ PMAS ], SCLMAS, TPMAS		

---

If a follower is in continuous positioning mode (MC1), FOLMD is the master distance over which acceleration or deceleration from the current ratio to the new ratio takes place. Or, if a follower is in preset positioning mode (MCØ), the FOLMD command indicates the master distance over which the next preset move will take place.

If scaling is enabled (SCALE1), the FOLMD value is specified in user units and is scaled by the SCLMAS parameter (for more detail on scaling, refer to page 16 or to the SCLMAS command description). Numeric variables (VAR) can be used with this command (e.g., FOLMD(VAR6)).

By carefully specifying accurate master distances for each ramp of a follower's move profile, a precise position relationship between master and follower will be maintained during all phases of the profile. The *Master and Follower Distance Calculation* section in the *Following* chapter of the *Programmer's Guide* discusses the relationship between ratio changes and the corresponding master and follower distances.

**HINT:** If a follower is in continuous mode (MC1) and the master is starting from rest, setting FOLMD to Ø will ensure precise tracking of the master's acceleration ramp.

**Examples:** (refer also to the FOLEN examples)

```
SCALE1      ; Enable parameter scaling
SCLMAS4000  ; Master scale factor is 4000 steps/rev
SCLD4000    ; Follower scale factor is 4000 steps/rev
DEL progx   ; Delete program called progx
DEF progx   ; Begin definition of program called progx
FOLMAS1     ; Master encoder is the master
FOLMD0     ; Assign Following acceleration distance to 0 master revs
            ; (i.e., instantaneous)
FOLRN1     ; Set follower-to-master Following ratio numerator to 1
FOLRD1     ; Set follower-to-master Following ratio denominator to 1
            ; Ratio set to 1:1
FOLEN1     ; Enable Following
D-         ; Set direction to opposite direction of the master
GO1        ; Begin following master. If the master is not moving, follower
            ; will remain at rest until master moves, at which time the
            ; follower will track the master precisely, but in the opposite
            ; direction as the master.
END        ; End definition of progx
```

---

## FOLRD Denominator of Follower-to-Master Ratio

Type	Following	Product	Rev
Syntax	<a_><!>FOLRD<r>	GT6K	6.0
Units	r = master distance in counts	GV6K	6.0
Range	r = 1.00000 - 999,999,999 (scalable by SCLMAS)		
Default	1		
Response	FOLRD *FOLRD1		
See Also	COMEXC, FGADV, FOLEN, FOLK, FOLMAS, FOLRN, FOLRNF, SCLMAS		

---

The FOLRD command establishes the denominator of a ratio between follower and master travel. (Ratios are always specified as positive, similar to velocity.) For a preset move (MCØ), it is the maximum allowed ratio, and for a continuous move (MC1), it is the final ratio reached by the follower. The actual follower direction will depend on commanded moves (D+ or D-) and master direction.

If no FOLRD parameter is specified, it is assumed to be 1.

If scaling is enabled (SCALE1), the FOLRD value is scaled by the SCLMAS value. For more detail on scaling, refer to page 16 or to the SCLMAS command description.

Numeric variables (VAR) can be used with this command for master parameters (e.g., FOLRD(VAR5)).

Each time FOLRN or FOLRD are given, the Gem6K drive divides the scaled numerator and denominator to calculate the ratio, but round-off errors are eliminated by measuring both master and follower over a large distance. After scaling, the maximum magnitude of the ratio is 127 follower steps for every master step.

**ON-THE-FLY CHANGES:** You can change Following ratio *on the fly* (while motion is in progress) in two ways. One way is to send an immediate command (!FOLRD) followed by an immediate go command (!GO). The other way is to enable the continuous command execution mode (COMEXC1) and execute a buffered command (FOLRD) followed by a buffered go command (GO).

**Example:** (refer also to the FOLEN examples)

```
SCLD25000      ; Set follower scaling factor to 25,000
SCLMAS4000     ; Set master scaling factor to 4,000
SCALE1        ; Enable scaling
FOLRN5        ; Set ratio numerator to 5 (5 * 25,000 = 125,000)
FOLRD3        ; Set ratio denominator to 3 (3 * 4,000 = 12,000)
               ; (Resulting ratio is 125 follower steps to every 12 master steps.)
```

---

## FOLRN Numerator of Follower-to-Master Ratio

Type	Following	Product	Rev
Syntax	<a_><!>FOLRN<r>	GT6K	6.0
Units	r = follower distance in steps	GV6K	6.0
Range	r = 0.00000 - 999,999,999.99999 (scalable by SCLD)		
Default	1		
Response	FOLRN *FOLRN1		
See Also	FGADV, FOLEN, FOLK, FOLMAS, FOLRNF, FOLRD, SCLD		

---

The FOLRN command establishes the numerator of a ratio between follower and master travel. (Ratios are always specified as positive, similar to velocity.) For a preset move (MCØ), it is the maximum allowed ratio, and for a continuous move (MC1), it is the final ratio reached by the follower. The actual follower direction will depend on commanded moves (D+ or D-) and master direction.

If no FOLRN parameter is specified, it is assumed to be 1.

If scaling is enabled (SCALE1), the FOLRN value is scaled by the SCLD value. For more detail on scaling, refer to page 16 or to the SCLD command description.

Numeric variables (VAR) can be used with this command for follower parameters (e.g., FOLRN(VAR2)).

Each time FOLRN or FOLRD are given, the Gem6K drive divides the scaled numerator and denominator to calculate the ratio, but round-off errors are eliminated by measuring both master and follower over a large distance. After scaling, the maximum magnitude of the ratio is 127 follower steps for every master step.

**ON-THE-FLY CHANGES:** You can change Following ratio *on the fly* (while motion is in progress) in two ways. One way is to send an immediate command (!FOLRN) followed by an immediate go command (!GO). The other way is to enable the continuous command execution mode (COMEXC1) and execute a buffered command (FOLRN) followed by a buffered go command (GO).

**Example:** refer to the FOLRD and FOLEN examples

---

<b>FOLRNF</b>		<b>Numerator of Final Follower-to-Master Ratio, Preset Moves</b>	
Type	Following; Compiled Motion	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>FOLRNF<r>	GT6K	6.0
Units	r = follower distance in steps	GV6K	6.0
Range	0.00000		
Default	0		
Response	FOLRNF *FOLRNF0		
See Also	FGADV, FOLEN, FOLRD, FOLRN, FOLMD, SCLD		

---

The FOLRNF command establishes the numerator of the final ratio between follower and master travel. (Ratios are always specified as positive, similar to velocity.) The FOLRNF command designates that the motor will move the load the distance designated in a preset GOBUF segment, completing the move at a final ratio of zero. FOLRNF applies only to the first subsequent GOBUF, which marks an intermediate “end of move” within a following profile. FOLRNF is used only in conjunction with the GOBUF command. Normal preset GO moves always finish with zero FOLRNF.

If scaling is enabled (SCALE1), the FOLRNF value is scaled by the SCLD value. For more detail on scaling, refer to page 16 or to the SCLD command description.

**NOTE:** The only allowable value for FOLRNF is 0, and it may only be used with compiled preset Following moves (a non-zero FOLRNF value will result in an immediate error message). FOLRNF is allowed for a segment only if the starting ratio is also zero (i.e., it must be the first segment, or the previous segment must have ended in zero ratio).

With compiled preset Following moves where FOLRNF has not been given, the final ratio is given with FOLRN, and the shape of the intermediate profile will be constrained to be within the starting and ending ratios.

For more information on using the FOLRNF command, refer to *Compiled Following Profiles* in the *Custom Profiling* chapter in the *Programmer's Guide*.

---

<b>FPPEN</b>		<b>Master Position Prediction Enable</b>	
Type	Following	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>FPPEN<b>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	b = 0 (disable) or 1 (enable)		
Default	1		
Response	FPPEN *FPPEN1		
See Also	[ FS ], TFS		

---

The FPPEN command enables or disables Master Position Prediction in the Gem6K drive Following algorithm. Master Position Prediction is enabled by default, but can be disabled as desired with the FPPEN0 command.

The Gem6K drive measures master position once per *position sample period* and calculates a corresponding follower commanded position. This calculation, and achieving the subsequent follower commanded position, requires 2 sample periods (4 milliseconds).

Enabling Master Position Prediction (FPPEN1) eliminates any lag in follower position which would be dependent on master speed. It may be desirable to disable Master Position Prediction (FPPEN0) when

maximum follower smoothness is important and minor phase delays can be accommodated. A detailed discussion of Master Position Prediction is given in the *Following* chapter of the *Programmer's Guide*.

**Example:**

```
FPPEN1          ; Enable Master Position Prediction.
```

**[ FS ]**

**Following Status**

Type	Following; Assignment or Comparison	<b>Product</b>	<b>Rev</b>
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	FGADV, FMAXA, FMAXV, FMCLen, FMCP, FOLEN, FOLMAS, FPPEN, FSHFC, FSHFD, MC, [ NMCY ], [ PMAS ], TFS, TFSF, VARB		

The FS command is used to assign the Following status bits to a binary variable, or to make a comparison against a binary or hexadecimal value. The function of each status bit is shown below.

Bit Assignment (left to right)	Function (YES = 1; NO = 0)	
1	Follower in Ratio Move	A Following move is in progress.
2	Ratio is Negative	The current ratio is negative (i.e., the follower counts are counting in the opposite direction from the master counts).
3	Follower Ratio Changing	The follower is ramping from one ratio to another (including a ramp to or from zero ratio).
4	Follower At Ratio	The follower is at constant non-zero ratio.
* 5	FOLMAS Active	A master is specified with the FOLMAS command.
* 6	FOLEN Active	Following has been enabled with the FOLEN command.
* 7	Master is Moving	The specified master is currently in motion.
8	Master Dir Neg	The current master direction is negative. (bit must be cleared to allow Following move in preset mode-MC0).
9	OK to Shift	Conditions are valid to issue shift commands (FSHFD or FSHFC).
10	Shifting now	A shift move is in progress.
11	Shift is Continuous	An FSHFC-based shift move is in progress.
12	Shift Dir is Neg	The direction of the shift move in progress is negative.
13	Master Cyc Trig Pend	A master cycle restart is pending the occurrence of the specified trigger.
14	Mas Cyc Len Given	A non-zero master cycle length has been specified with the FMCLen command.
15	Master Cyc Pos Neg	The current master cycle position (PMAS) is negative. This could be caused by a negative initial master cycle position (FMCP), or if the master is moving in the negative direction.
16	Master Cyc Num > 0	The master position (PMAS) has exceeded the master cycle length (FMCLen) at least once, causing the master cycle number (NMCY) to increment.
17	Mas Pos Prediction On	Master position prediction has been enabled (FPPEN).
18	Master Filtering On	A non-zero value for master position filtering (FFILT) is in effect.
19	RESERVED	
20	RESERVED	

(continued on next page)

21	RESERVED	
22	RESERVED	
23	OK to do FGADV move	OK to do Geared Advance move (master assigned with FOLMAS, Following enabled with FOLEN, and follower axis is either not moving, or moving at constant ratio in continuous mode).
24	FGADV move underway	Geared Advance move profile is in progress.
25	RESERVED	
26	FMAXA/FMAXV limited	The present Following move profile is being limited by FMAXA or FMAXV.
27-32	RESERVED	

\* All these conditions must be true before Following motion will occur.

**Syntax:** VARBn=FS where n is the binary variable number, or FS can be used in an expression such as IF(FS=b1101), or IF(FS=h7F).

To make a comparison against a binary value, place the letter b (b or B) in front of the value that the Following status is being compared against. 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 that the Following status is being compared against. The hexadecimal value itself must only contain the letters A through F, and the numbers 0 through 9.

If you wish to assign only one bit of the Following status to a binary variable, instead of all 32, use the bit select (.) operator. The bit select, in conjunction with the bit number, is used to specify a specific Following status bit (e.g., VARB1=FS.12 assigns status bit 12 to binary variable 1).

**Example:**

```

VARB1=FS           ; Following status assigned to binary variable 1
VARB2=1FS.12      ; Following status bit 12 assigned to binary variable 2
VARB2              ; Response if bit 12 is set to 1 should be:
                  ; *VARB2=XXXX_XXXX_XXX1_XXXX_XXXX_XXXX_XXXX_XXXX
IF(FS=b111011X11) ; If the Following status contains 1's for bit locations
                  ; 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(FS=h7F00)      ; If the Following status contains 1's for bits 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

```

---

## FSHFC Continuous Shift

Type	Following	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>FSHFC<i>	GT6K	6.0
Units	i = shift feature to implement	GV6K	6.0
Range	i = 0 (stop), 1 (positive-direction), 2 (negative-direction), or 3 (kill)		
Default	n/a		
Response	n/a		
See Also	FGADV, FOLEN, FOLK, FOLRN, FOLRNF, FOLRD, [ FS ], FSHFD, MC, [ PSHF ], TFS, TPSHF		

---

The FSHFC command allows time-based follower moves to be superimposed on continuous Following moves. This results in a *shift* (change in phase) between the master position and the follower position. Continuous shift moves in the positive- or negative-direction may be commanded only while the follower is in the Following mode (FOLEN1).

**Steppers only:** An FSHFC move may be performed only when the follower is in the continuous positioning mode (MC1) and performing a Following move at a constant ratio.

The most recently commanded velocity (V) and acceleration (A) for the follower axis will determine the speed at which the FSHFC move takes place. The velocity and direction of the FSHFC shift is independently superimposed on whatever velocity and direction results from the ratio and motion of the master. The FSHFC shift is not a change in ratio; rather, it is a velocity added to a ratio. The velocity commanded is added to the present speed at which the follower is moving, up to the velocity limit of the product. For example, assume a follower is traveling at 1 rps in the positive direction while following a master. If a FSHFC move is commanded in the positive direction at 2 rps, the follower's actual velocity (after acceleration) will be 3 rps.

The FSHFC parameters stop (Ø) and kill (3) can be used to halt a continuous FSHFC move (positive-direction or negative-direction). The example below shows how to stop a FSHFC continuous move.

An FSHFC move may be needed to adjust the relative follower position on the fly during the continuous Following move. For example, suppose an operator is visually inspecting the follower's motion with respect to the master. If he notices that the master and follower are out of synchronization, it may be desirable to have an interrupt programmed (e.g., activated with a push-button switch) that will allow the operator to advance or retard the follower at a superimposed correction speed until the operator chooses to have the follower start tracking the master again. The example below shows this.

**FSHFC Example:**

Assume all scale factors and set-up parameters have been entered for the master and follower. In this example, the follower is continually following the master at a 1:1 ratio. If the operator notices some misalignment between master and follower, he can press 1 of 2 pushbuttons (connected to onboard trigger inputs #1 and #2, which are also referred to as TRG-1A and TRG-1B) to shift the follower in the positive- or negative-direction at 0.1 user scaled units until the button is released. After the adjustment, the program continues on as before.

**Example:**

```

DEF SHIFT          ; Begin definition of program called SHIFT
V.1                ; Add or subtract 0.1 user scaled units from the follower velocity
                  ; when shifting
COMEXS1           ; Continue command execution after stop
COMEXC1           ; Continue command execution during motion
FOLMAS1           ; Master encoder input is the master
FOLRN1            ; Set follower-to-master Following ratio numerator to 1
FOLRD1            ; Set follower-to-master Following ratio denominator to 1
                  ; (ratio set to 1:1)
FOLEN1            ; Enable Following mode
D+                ; Set to positive-direction
MC1               ; Select continuous positioning mode
GO1               ; Start following master continuously
VARB1=b10         ; Define onboard input pattern #1 and assign to VARB1
VARB2=b01         ; Define onboard input pattern #2 and assign to VARB2
$TESTIN           ; Define label called TESTIN
IF(IN=VARB1)      ; IF statement (if onboard input #1 is activated, do the jump)
  JUMP SHIFTP     ; Jump to shift follower in the positive-direction when pattern 1
                  ; active
  NIF             ; End of IF statement
IF(IN=VARB2)      ; IF statement (if onboard input #2 is activated, do the jump)
  JUMP SHIFTN     ; Jump to shift follower in the negative-direction when pattern 2
                  ; active
  NIF             ; End of IF statement
JUMP TESTIN       ; Return to main program loop
$SHIFTP           ; Define label called SHIFTP (subroutine to shift in the
                  ; positive direction)
FSHFC1            ; Start continuous follower shift move in positive-direction
WAIT(IN.1=b0)    ; Continue shift until onboard input #1 is deactivated
FSHFC0            ; Stop shift move
JUMP TESTIN       ; Return to main program loop
$SHIFTN           ; Define label called SHIFTN (subroutine to shift in the

```

```

; negative-direction)
FSHFC2      ; Start continuous follower shift move in the negative-direction
WAIT(IN.2=b0) ; Continue shift until onboard input #2 is deactivated
FSHFC0      ; Stop shift move
JUMP TESTIN ; Return to main program loop
END          ; End definition of program called SHIFT

```

---

## FSHFD Preset Shift

		Product	Rev
Type	Following		
Syntax	<a_><!>FSHFD<r>	GT6K	6.0
Units	r = shift distance	GV6K	6.0
Range	r = 0.00000 - 999,999,999 (scalable with SCLD)		
Default	n/a		
Response	n/a		
See Also	FGADV, FOLEN, FOLK, FOLRN, FOLRNF, FOLRD, [ FS ], FSHFC, MC, ONCOND, [ PSHF ], SCLD, TFS, TPSHF		

---

The FSHFD command allows time-based follower moves to be superimposed on continuous Following moves. This results in a *shift* (change in phase, or registration) between the master position and the follower position. Preset shift moves of defined or variable distances, may be commanded only while the follower is in the Following mode (FOLEN1). The FSHFD distance is scaled by the SCLD value of scaling is enabled (SCALE1).

The most recently commanded velocity (V) and acceleration (A) for the follower axis will determine the speed at which the FSHFD move takes place. The velocity and direction of the FSHFD shift is independently superimposed on whatever velocity and direction results from the ratio and motion of the master.

The FSHFC parameters stop (Ø) and kill (3) can be used to halt an FSHFD.

It should be noted that FSHFD is similar in execution to GO. The entire preset distance shift, or ramp-to-shift velocity, must finish before the Gem6K drive proceeds to the next command.

The FSHFD shift is not a change in ratio; rather, it is a velocity added to a ratio. The velocity commanded will be added to the present speed at which the follower is moving, up to the velocity limit of the product. For example, assume a follower is traveling at 1 rps in the positive direction while following a master. If a FSHFD move is commanded in the positive direction at 2 rps, the follower's actual velocity (after acceleration) will be 3 rps.

An FSHFD move may be needed to adjust the follower position on the fly because of a load condition which changes during the continuous Following move. For example, suppose an operator is visually inspecting the follower's motion with respect to the master. If the operator notices that the master and follower are out of synchronization, it may be desirable to have an input programmed (e.g., activated with a push-button switch) that will allow the operator to advance or retard the follower a fixed distance, and then let the follower resume tracking the master. The example below illustrates this.

### FSHFD Example:

Assume all scale factors and set-up parameters have been entered for the master and follower. In this example, the follower is continually following the master at a 1:1 ratio. If the operator notices some misalignment between master and follower, he can press 1 of 2 pushbuttons (connected to onboard trigger inputs #1 and #2, which are also referred to as TRG-1A and TRG-1B) to advance or retard the follower a fixed distance of 200 steps. After the adjustment, the follower resumes tracking the master as before.

#### Example:

```

DEF PSHIFT      ; Begin definition of program called PSHIFT
COMEXS1         ; Continue command execution after stop
COMEXC1         ; Continue command execution during motion
FOLMAS1         ; Master encoder input is the master for axis 1
FOLRN1          ; Set follower-to-master Following ratio numerator to 1
FOLRD1          ; Set follower-to-master Following ratio denominator to 1
                ; (ratio set to 1:1)
FOLEN1          ; Enable Following mode

```



```

D+           ; Set direction to positive
MC1         ; Select continuous positioning mode
GO1         ; Start following master continuously
VARB1=b10   ; Define input pattern #1 and assign to VARB
VARB2=b01   ; Define input pattern #2 and assign to VARB
$TESTIN     ; Define label called TESTIN
IF(IN=VARB1) ; IF statement (if onboard input #1 is activated, do the jump)
  JUMP SHIFTP ; Jump to shift follower in positive-direction when pattern 1 active
  NIF         ; End of IF statement
IF(IN=VARB2) ; IF statement (if onboard input #2 is activated, do the jump)
  JUMP SHIFTN ; Jump to shift follower in negative-direction when pattern 2 active
  NIF         ; End of IF statement
JUMP TESTIN ; Return to main program loop
$$SHIFTP    ; Define label called SHIFTP (subroutine to shift in the
            ; positive direction)
FSHFD200    ; Start preset follower shift move of 200 steps in positive direction
WAIT(FS.10=b0) ; Wait for shift to finish
JUMP TESTIN ; Return to main program loop
$$SHIFTN    ; Define label called SHIFTN (subroutine to shift in the
            ; negative direction)
FSHFD-200   ; Start preset follower shift move of 200 steps in the negative
            ; direction
WAIT(FS.10=b0) ; Wait for shift to finish
JUMP TESTIN ; Return to main program loop
END         ; End definition of program called PSHIFT

```

---

## **FVMACC**      **Virtual Master Count Acceleration**

Type	Following	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>FVMACC<i>	GT6K	6.0
Units	i = count acceleration in counts/sec/sec	GV6K	6.0
Range	0-999,999,999		
Default	0		
Response	FVMACC    *FVMACC+0		
See Also	FOLMAS, FVMFRQ, SINAMP, SINANG, SINGO		

---

Use the FVMACC command to define the rate at which the virtual master internal count frequency may change. This command allows smooth changes in master velocity and direction.

---

## **FVMFRQ**      **Virtual Master Count Frequency**

Type	Following	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!> FVMFRQ<i>	GT6K	6.0
Units	i = count frequency in counts/sec	GV6K	6.0
Range	±1,000,000		
Default	+0		
Response	FVMFRQ    *FVMFRQ+0		
See Also	FOLMAS, FVMACC, SINAMP, SINANG, SINGO		

---

Use the FVMFRQ command to define the virtual master count frequency. The “virtual master” is an internal count source, intended to mimic the counts which might be received on an external encoder port. Just as may be encountered with an external encoder, this count source may speed up, slow down, stop, or count backwards.

The count source has a variable count frequency, defined by the user. The count source is always enabled, counting at the signed rate specified by this command. To start and stop the count source, specify non-zero or zero values, respectively, for the FVMFRQ command.

The rate at which the count frequency may change is specified in counts per second per second with the FVMACC command, allowing smooth changes in master velocity and direction.

## GO

### Initiate Motion

		Product	Rev
Type	Motion	GT6K	6.0
Syntax	<a_><!>GO<b>	GV6K	6.0
Units	n/a		
Range	b = 1 (go); or blank		
Default	1		
Response	GO: No response; instead, motion is initiated		
See Also	A, AA, AD, ADA, COMEXC, D, GOBUF, GOWHEN, K, LH, LS, MA, MC, PSET, S, SCLA, SCLD, SCLV, SSV, TAS, TEST, V		

The 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: A, AA, AD, ADA, D, V, LH, LS, MA, and MC.

The GO1 command and the GO command both start motion.

If motion does not occur after a GO command has been issued, check TAS bits 15-18 to ascertain if the hardware or software end-of-travel limits have been encountered.

#### On-The-Fly (Pre-emptive GO) Motion Profiling

While motion is in progress (regardless of the positioning mode), you can change these motion parameters “on-the-fly” (OTF) to affect a new profile:

- Acceleration (A) — S-curve acceleration is not supported in OTF motion changes
- Deceleration (AD) — S-curve acceleration is not supported in OTF motion changes
- Velocity (V)
- Distance (D)
- Preset or Continuous Positioning Mode Selection (MC)
- Incremental or Absolute Positioning Mode Selection (MA)
- Following Ratio Numerator and Denominator (FOLRN and FOLRD, respectively)

The motion parameters can be changed by sending the respective command (e.g., A, V, D, MC, etc.) followed by the GO command. If the continuous command execution mode is enabled (COMEXC1), you can execute buffered commands; otherwise, you must prefix each command with an immediate command identifier (e.g., !A, !V, !D, !MC, etc., followed by !GO). The new GO command pre-empts the motion profile in progress with a new profile based on the new motion parameter(s).

For more information, refer to the *Custom Profiling* section in the *Programmer's Guide*.

#### Example:

```
SCALE1           ; Enable scaling
SCLA25000        ; Set the accel. scale factor to 25000 steps/unit
SCLV25000        ; Set the velocity scale factor to 25000 steps/unit
SCLD1           ; Set the distance scaling factor to 1 step/unit
DEL progA        ; Delete program called progA
DEF progA        ; Begin definition of program called progA
MA0             ; Incremental positioning mode
MC0             ; Preset positioning mode
A10             ; Set the acceleration to 10 units/sec/sec
V1             ; Set the velocity to 1 units/
D100000         ; Set the distance to 100000 units
GO1             ; Initiate motion
END             ; End definition of progA
```

---

## GOBUF Store a Motion Segment in Compiled Memory

Type	Compiled Motion	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>GOBUF<b>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	b = 0 (don't go) or 1 (go)		
Default	1		
Response	n/a		
See Also	[ AS ], DEF, END, [ ER ], FOLRNF, MC, MEMORY, PCOMP, PEXE, POUTn, PRUN, PUCOMP, PLOOP, PLN, [ SS ], TAS, TER, TSS, VF, (Compiled Motion overview in <i>Custom Profiling in the Programmer's Guide.</i> )		

---

The GOBUF command creates a motion segment as part of a profile and places it in a segment of compiled memory, to be executed after all previous GOBUF motion segments have been executed. When a GOBUF command is executed, the distance from the new D command is added to the profile's current goal position as soon as the GOBUF command is executed, thus extending the overall move distance of the profile under construction.

GOBUF is not a stand-alone command; it can only be executed within compiled programs, using the PCOMP and PRUN commands.

Each GOBUF motion segment may have its own distance to travel, velocity, acceleration and deceleration. The end of a preset segment (MCØ) is determined by the distance or position specified; a compiled MCØ GOBUF motion segment is finished when the "D" goal is reached. The end of a continuous motion (MC1) segment is determined by the ratio or velocity specified; a compiled MC1 GOBUF motion segment is finished when the velocity or ratio goal is reached. If either a preset segment or continuous segment is followed by a compiled GOWHEN command, motion will continue at the last velocity until the GOWHEN condition becomes true, and the next segment begins.

The GOBUF command is not allowed during absolute positioning mode (MA1).

### Starting velocity of a GOBUF segment

Every GOBUF motion segment will start at a velocity equal to the previous segment's end velocity. If the previous GOBUF segment uses the VFØ command, then it will end at zero velocity; otherwise, the end velocity will be equal to the goal velocity (V) of the previous segment.

### Ending velocity of a GOBUF segment

#### *Preset Positioning Mode (MCØ)*

A preset motion segment starts at the previous motion segment's end velocity, attempts to reach the goal velocity (V) with the programmed acceleration and deceleration (A and AD) values, and is considered completed when the distance (D) goal is reached.

In non-Following motion (FOLENØ), the last preset GOBUF segment always ends at zero velocity, but if you wish the velocity between intermediate GOBUF segments to end at zero velocity, use the VFØ command. In Following mode (FOLEN1), the last preset GOBUF segment will end with the last-specified goal velocity, but if you wish the velocity between intermediate GOBUF segments to end at zero velocity, use the FOLRNF command.

Each GOBUF will build a motion segment that, by default, becomes known as the last segment in the profile. The last motion segment in a profile must end at zero velocity. If using pre-compiled loops (PLOOP) and the loop is closed after the last GOBUF segment (PLN occurs after the last GOBUF), then the unit will not consider the last GOBUF as a final motion segment since it can link to either the first segment of the loop or the next segment after the loop. If the conditions are such that the last motion segment is within a loop and does not end at zero velocity, then an error is generated (TSS/SS bit #31 is set) at compile time (PCOMP), and the profile remains un-compiled.

#### *Continuous Positioning Mode (MC1)*

A continuous segment starts at the previous motion segment's end velocity, and is considered complete when it reaches the goal velocity (V) at the programmed accel (A) or decel (AD) values.

You may use a mode continuous (MC1) non-zero velocity segment as the last motion segment in a profile (no error will result). The axis will continue traveling at the goal velocity.

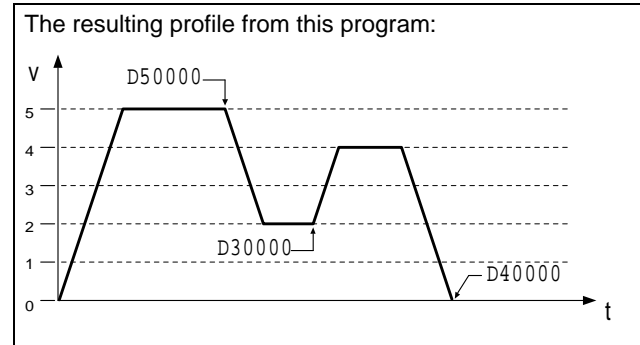
**NOTE:** Each GOBUF motion segment can consume from 2-8 memory segments of compiled memory. If there is no more space left in compiled memory, a compilation error will result.

**Example:**

```

DEF simple ; Begin definition of program
MC0       ; Preset positioning mode
D50000    ; Distance is 50000
A10       ; Acceleration is 10
AD10      ; Deceleration is 10
V5        ; Velocity is 5
GOBUF1    ; 1st motion segment
D30000    ; Distance is 30000
V2        ; Velocity is 2
GOBUF1    ; 2nd motion segment
D40000    ; Distance is 40000
V4        ; Velocity is 4
GOBUF1    ; 3rd motion segment
END       ; End program definition

```



```

PCOMP simple ; Compile simple
PRUN simple ; Run simple

```

---

## GOSUB Call a Subroutine

Type	Program; Subroutine Definition; Program Flow Control	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>GOSUB<t>	GT6K	6.0
Units	t = text (name of program/subroutine)	GV6K	6.0
Range	Text name of 6 characters or less		
Default	n/a		
Response	n/a		
See Also	\$, BREAK, DEF, DEL, END, ERASE, GOTO, JUMP, RUN		

---

The 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:**

```

DEF prog1 ; Begin definition of program prog1 to be used as a subroutine
COMEXC0 ; Disable continuous command processing mode
A10 ; Set acceleration to 10 revs/sec/sec
V1 ; Set velocity to 1 rev/sec
D4000 ; Set distance to 4000 counts
L10 ; Loop 10 times
GO1 ; Initiate motion
GOSUB prog2 ; GOSUB to program prog2 as a subroutine
LN ; End loop
END ; End subroutine definition of program prog1
DEF prog2 ; Begin definition of program prog2
OUT1 ; Turn on output #1
T1 ; Time delay of 1 second
OUT0 ; Turn off output #1
END ; End subroutine definition of program prog2
RUN prog1 ; Execute program prog1
; After program prog1 is initiated, and the first move of 4000 counts is complete, a
; GOSUB will occur causing the execution of program prog2. After program prog2
; finishes, control will be passed back to program prog1, to the command immediately
; following the GOSUB.

```

---

## GOTO

### Goto a Program or Label

Type	Program; Subroutine Definition; Program Flow Control	<b>Product</b>	<b>Rev</b>
Syntax	<!>GOTO<t>	GT6K	6.0
Units	t = text (name of program/label)	GV6K	6.0
Range	Text name of 6 characters or less		
Default	n/a		
Response	n/a		
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.

<b>CAUTION</b>
----------------

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:

```
DEF pick          ; Begin definition of subroutine named pick
D-5000           ; Set distance to -5000
GO1              ; Initiate motion
END              ; End subroutine definition
DEF place        ; Begin definition of subroutine named place
GOTO pick        ; Goto to subroutine named pick
D+5000           ; Set distance to +5000
GO1              ; Initiate motion
END              ; End subroutine definition
place            ; Execute program named place
; After the GOTO command, the D+5000 and GO1 commands 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.
```

---

## GOWHEN

### Conditional Go

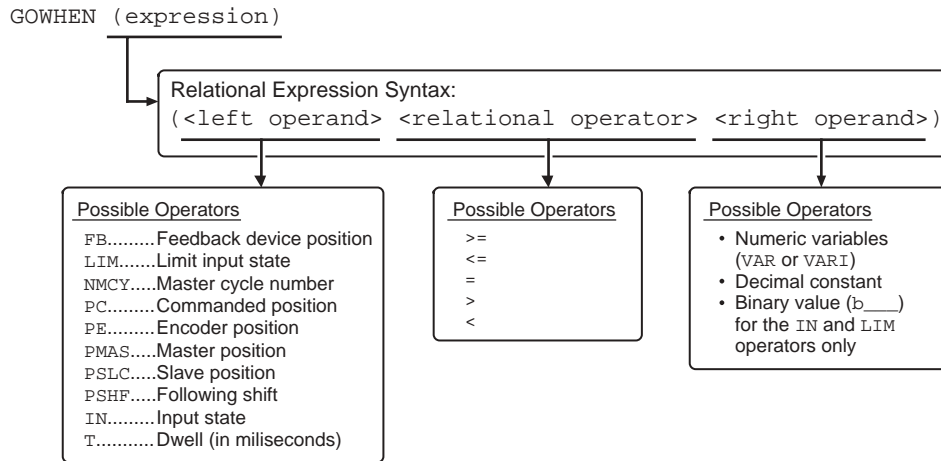
Type	Motion	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>GOWHEN(expression)	GT6K	6.0
Units	n/a	GV6K	6.0
Range	Up to 80 characters (including parentheses)		
Default	n/a		
Response	n/a		
See Also	[ AS ], COMEXC, [ ER ], ERROR, ERRORP, [ FB ], FGADV, FSHFC, FSHFD, GO, [ IN ], [ LIM ], [ NMCY ], [ PC ], [ PE ], [ PMAS ], [ PSHF ], [ PSLV ], T, TAS, TER, TRGFN, WAIT		

---

Use the GOWHEN command to synchronize a motion profile of an axis with a specified position count (commanded, feedback device, motor, master, follower, Following shift), input status, dwell (time delay), or master cycle number on that axis. Command processing does not wait for the GOWHEN conditions (relational expressions) to become true during the GOWHEN command. Rather, the motion from the subsequent start-motion command (GO, FGADV, FSHFC, and FSHFD) will be suspended until the condition becomes true.

Start-motion type commands that **cannot** be synchronized using the GOWHEN command are: HOM, JOG, JOY, and PRUN. A preset GO command that is already in motion can start a new profile using the GOWHEN and GO sequence of commands. Continuous moves (MC1) already in progress can change to a new velocity based upon the GOWHEN and GO sequence. Both preset and continuous moves can be started from rest with the GOWHEN and GO sequence.

### GOWHEN Syntax:



EXAMPLES	
GOWHEN (PE>40000)	; suspend next GO until encoder position > 40000
GOWHEN (IN.4=b1)	; suspend next GO until onboard input #4 is activated (b1)
GOWHEN (PMAS>255)	; suspend next GO until the master has traveled 255 master distance units

SCALING
If scaling is enabled (SCALE1), the right-hand operand is multiplied by SCLD if the left-hand operand is FB, PC, PE, PSLV, or PSHF. The right-hand operand is multiplied by the SCLMAS value if the left-hand operand is PMAS. The SCLD or SCLMAS values used correlate to the axis specified with the variable (e.g., a GOWHEN expression with PE scales the encoder position by the SCLD value specified).

### GOWHEN Status:

**Axis Status — Bit #26:** Bit #26 is set when motion has been commanded by a GO, FGADV, FSHFC, or FSHFD command, but the change in motion is suspended due to a pending GOWHEN condition. This status bit is cleared when the GOWHEN condition is true or when a stop (!S) or kill (!K or ^K) command is executed. A GOWHEN command can be cleared using an axis-specific S or K command (e.g., !S1 or !K1).

- AS. 26 .....Assignment & comparison operator — use in a conditional expression (see AS).
- TASF.....Full text description of each status bit. (see “Gowhen is Pending” line item)
- TAS.....Binary report of each status bit (bits 1-32 from left to right). [See bit #26.](#)

**Axis Status — Bit #29:** Bit #29 is set if the input state or position relationship specified in the GOWHEN command is already true when the subsequent GO, FGADV, FSHFC, or FSHFD command is issued. To clear this error condition, issue another GOWHEN command; or issue a !K command and check the program logic (use the TRACE and STEP features if necessary).

- AS. 29 .....Assignment & comparison operator — use in a conditional expression (see AS).
- TASF.....Full text description of each status bit. (see “Gowhen error” line item)
- TAS.....Binary report of each status bit (bits 1-32 from left to right). [See bit #29.](#)

**Error Status — Bit #14:** Bit #14 is set if the input state or position relationship specified in the GOWHEN command is already true when the GO, FGADV, FSHFC, or FSHFD command is issued. The error status is monitored and reported only if you enable error-checking bit #14 with the ERROR command (e.g., ERROR.14-1). NOTE: When the error occurs, the drive will branch to the error program (assigned with the ERRORP command). To clear this error condition, issue another GOWHEN command; or issue a !K command and check the program logic (use the TRACE and STEP features if necessary).

ER.14..... Assignment & comparison operator — use in a conditional expression (see AS).

TERF..... Full text description of each status bit. (see “Gowhen condition true” line item)

TER..... Binary report of each status bit (bits 1-32 from left to right). See bit #14.

### GOWHEN ... on a Trigger Input: Use a TRGFNC1

If you wish motion to be triggered with a trigger input, use the TRGFNC1 command. The TRGFNC1 command executes in the same manner as the GOWHEN command, except that motion is executed when the specified trigger input (c) is activated. For more information, refer to the TRGFN command description.

### GOWHEN vs. WAIT:

A WAIT will cause program flow to halt (except for execution of immediate commands) until the condition specified is satisfied. Common uses for this function include delaying subsequent I/O activation until the master has achieved a required position or an object has been sensed.

By contrast, a GOWHEN will suspend the motion profile until the specified condition is met. It does **not** affect program flow. If you wish motion to be triggered with a trigger input, use the TRGFNC1 command. The TRGFNC1 command executes in the same manner as the GOWHEN command, except that motion is executed when the specified trigger input (c) is activated (see TRGFN command description for details). In addition, GOWHEN expressions are limited to the operands listed above; WAIT can use additional operands such as FS (Following status) and VMAS (velocity of master).

### Factors Affecting GOWHEN Execution:

If a second GOWHEN command is executed **before** a start-motion command (GO, FGADV, FSHFC, or FSHFD), then the first GOWHEN is over-written by the second GOWHEN command. (GOWHEN commands are not nested.) An error is not generated when a GOWHEN command is over-written by another GOWHEN.

While waiting for a GOWHEN condition to be met **and** a start-motion command **has** been issued, if a second GOWHEN command is encountered, then the first sequence is disabled and another start-motion command is needed to re-arm the second GOWHEN sequence.

A new GOWHEN command must be issued for each start-motion command (GO, FGADV, FSHFC, or FSHFD). That is, once a GOWHEN condition is met and the motion command is executed, subsequent motion commands will not be affected by the same GOWHEN command.

If the GOWHEN and start-motion commands are issued, the motion profile is delayed until the GOWHEN condition is met. If a second start-motion command is encountered, the second start-motion command will override the GOWHEN command and start motion. If this override situation is not desired, it can be avoided by using a WAIT condition between the first start-motion command and the second start-motion command.

It is probable that the GOWHEN command, the GO command, and the GOWHEN condition becoming true may be separated in time, and by other commands. Situations may arise, or commands may be given which make the GOWHEN invalid or inappropriate. In these cases, the GOWHEN condition is cleared, and any motion pending the GOWHEN condition becoming true is canceled. These situations include execution of the JOG, JOY, HOM, PRUN, and DRIVEØ commands, as well motion being stopped due to hard or soft limits, a drive fault, an immediate stop (!S), or an immediate kill (!K or ^K).

**GOWHEN in Compiled Motion:** When used in a compiled program, a GOWHEN will pause the profile in progress (motion continues at constant velocity) until the GOWHEN condition evaluates true. When executing a compiled Following profile, the GOWHEN is ignored on the reverse Following path (i.e., when the master is moving in the opposite direction of that which is specified in the FOLMAS command). A compiled GOWHEN may require up to 4 segments of compiled memory storage.

### Example:

In the example below, the axis must start motion when input 3 becomes active. While input 3 is off, the program must be monitoring inputs and serving other system requirements, so a WAIT statement cannot be used; instead, a GOWHEN and GO sequence will delay the profile of the axis.

```
SCALE1          ; Enable scaling
SCLV25000       ; Set velocity scaling factor
SCLD10000       ; Set distance scaling factor
DEL prog1       ; Delete program called prog1
DEF prog1       ; Begin definition of program called prog1
MCO             ; Set to preset move mode
D20            ; Set distance end-point
COMEXC1        ; Enable continuous command execution mode
V1             ; Set velocity
A100           ; Set acceleration
GOWHEN(IN.3=B1) ; Delay profile. When the expression is true
                ; (input 3 is active), allow the axis to start motion
GO1            ; Command axis to move. Axis will not start until
                ; conditions in the GOWHEN statement are true.
                ; Command processing does not wait, so other system
                ; functions may be performed.
END            ; End definition of program
```

---

## HALT Terminate Program Execution

Type	Program Flow Control	<b>Product</b>	<b>Rev</b>
Syntax	<!>HALT	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	BP, BREAK, C, ELSE, IF, K, NIF, NWHILE, PS, REPEAT, S, T, UNTIL, WAIT, WHILE		

---

The 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:

```
DEF prog1       ; Define a program called prog1
GO1            ; Initiate motion
GOSUB prog2    ; Gosub to subroutine named prog2
GO1            ; Initiate motion again
END            ; End program definition
DEF prog2       ; Define a program called prog2
GO1            ; Initiate motion
IF(IN=b1X0)    ; If onboard input 1 is active (1), and input 3 is inactive (0)
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 the encoder
NIF           ; End If statement
END            ; End program definition
RUN prog1      ; Execute program prog2
;
; Upon completion of motion, 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

### Technical Support

Type	Program Debug Tool	<b>Product</b>	<b>Rev</b>
Syntax	<!>HELP	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	See description below		
See Also	None		

---

The (HELP) command provides the telephone numbers for technical support.

---

## HOM

### Go Home

Type	Homing	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>HOM<b>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	b = 0 (home in positive direction), 1 (home in negative direction), or X (do not home)		
Default	X		
Response	n/a		
See Also	[ AS ], HOMA, HOMAA, HOMAD, HOMADA, HOMBAC, HOMDF, HOMEDG, HOMV, HOMVF, HOMZ, PSET, TAS, TIN, TLIM		

---

The HOM command instructs the Gem6K drive to search for the home position in the direction specified by the command. If an end-of-travel limit is activated while searching for the home limit, the drive 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 the axis status register (refer to the TAS or AS command). *When the homing operation is successfully completed, the absolute position register is set to zero (equivalent to PSETØ).*

<b>NOTE</b>
-------------

Pause and resume functions are not recommended during the homing operation. A Pause command or input will pause the homing motion; however, when the subsequent Resume command or input occurs, motion will resume at the beginning of the homing motion sequence.
--

---

The homing operation has several parameters that determine the homing algorithm:

- Home acceleration (HOMA and HOMAA)
- Home deceleration (HOMAD and HOMADA)
- Home velocity (HOMV)
- Final home velocity (HOMVF)
- Home reference edge (HOMEDG)
- Backup to home (HOMBAC)
- Final home direction (HOMDF)
- Home to encoder Z-channel (HOMZ)

For more information on homing, including sample scenarios, refer to the *Homing* section of the *Programmer's Guide*.

**Example:**

```
SCALE1          ; Enable scaling
SCLA25000       ; Set accel. Scaling to 25000 steps/unit/unit;
SCLV25000       ; Set vel. Scaling to 25000 steps/unit;
SCLD1          ; Set distance scaling factor to 1 step/unit
DEL Homrdy      ; Delete program called Homrdy
DEF Homrdy      ; Begin definition of program called Homrdy
MA0             ; Incremental mode
MC0            ; Preset mode
HOMA10         ; Set home acceleration to 10 units/sec/sec
HOMAD20        ; Set home deceleration to 20 units/sec/sec
HOMBAC1        ; Enable backup to home switch
HOMEDG0        ; Stop on the positive-direction edge of the home switch
HOMDF0         ; Set final home direction to positive
HOMZ0          ; Disable homing to encoder Z-channel
HOMV1          ; Set home velocity to 1 units/sec
HOMVF.1        ; Sets home final velocity to 0.1 units/sec
HOMO           ; Execute go home in positive-direction
END            ; End definition of Homrdy
```

---

## HOMA Home Acceleration

Type	Homing	Product	Rev
Syntax	<a_><!>HOMA<r>	GT6K	6.0
Units	r = units/sec/sec (linear motors: see DMEPIT for linear/rotary conversion)	GV6K	6.0
Range	0.0001 - 9999.9999		
Default	10.0000		
Response	HOMA: *HOMA10.0000		
See Also	DMEPIT, HOM, HOMA, HOMAA, HOMAD, HOMADA, HOMBAC, HOMDF, HOMEDG, HOMV, HOMVF, HOMZ, SCALE, SCLA		

---

The HOMA command specifies the acceleration rate to be used upon executing the next go home (HOM) command.

**UNITS OF MEASURE and SCALING:** refer to page 16.

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

Type	Motion (S-Curve)	Product	Rev
Syntax	<a_><!\>HOMAA<r >	GT6K	6.0
Units	r = units/sec/sec	GV6K	6.0
Range	0, or 0.0001 - 9999.9999		
Default	10.00 (Default is a constant accel ramp, where HOMAA tracks HOMA. S-curve accel is achieved when HOMAA is changed independent of HOMA. To restore a constant accel ramp and tracking, set HOMAA = 0.)		
Response	HOMAA: *HOMAA10.0000		
See Also	A, AD, ADA, HOM, HOMA, HOMAA, HOMAD, HOMADA, , HOMBAC, SCALE, SCLA		

---

The 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*. Refer to page 15 for details on S-curve profiling.

Scaling (SCLA) affects HOMAA the same as it does for HOMA. Refer to page 16 for details on scaling.

### Example:

```
SCALE0          ; Disable scaling
DEL proge       ; Delete program called proge
DEF proge       ; Begin definition of program called proge
MA0             ; Select incremental positioning mode
HOMA10         ; Set homing max. accel to 10 rev/sec/sec
HOMAA5         ; Set homing avg. accel to 5 rev/sec/sec
HOMAD10        ; Set homing max. decel to 10 rev/sec/sec
HOMADA5        ; Set homing avg. decel to 5 rev/sec/sec
HOM1           ; Execute a pure S-curve homing move in the negative-direction
END            ; End definition of program
```

---

## HOMAD Home Deceleration

Type	Homing	Product	Rev
Syntax	<a_><!\>HOMAD<r>	GT6K	6.0
Units	r = units/sec/sec	GV6K	6.0
Range	0, or 0.0001 - 9999.9999		
Default	10.0000 (by default, HOMAD tracks HOMA until HOMAD is changed independent of HOMA; to restore tracking, set HOMAD = 0)		
Response	HOMAD: *HOMAD10.0000		
See Also	HOM, HOMA, HOMAA, HOMAD, HOMADA, HOMBAC, HOMEDG, HOMDF, HOMV, HOMVF, HOMZ, [ LIM ], LIMEN, LIMLVL, SCALE, SCLA		

---

The HOMAD command specifies the deceleration rate to be used upon executing the next go home (HOM) command.

<b>UNITS OF MEASURE and SCALING:</b> refer to page 16.
--

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 HOMAD command has not been entered, the home acceleration (HOMA) command will set the deceleration rate. Once the 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

Type	Motion (S-Curve)	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>HOMADA<r>	GT6K	6.0
Units	r = units/sec/sec	GV6K	6.0
Range	0.0001 - 9999.9999		
Default	10.00 (By default, HOMADA tracks HOMAA until HOMADA is changed independent of HOMAA. To restore tracking, set HOMAD = 0.)		
Response	HOMADA: *HOMADA10.0000		
See Also	A, AD, HOM, HOMA, HOMAA, HOMAD, HOMADA???, SCALE, SCLA		

---

The 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*. Refer to page 15 for details on S-curve profiling.

Scaling (SCLA) affects HOMADA the same as it does for HOMAD. Refer to page 16 for details on scaling.

**Example:** Refer to the Homing Average Acceleration (HOMAA) command example.

---

## HOMBAC Home Backup Enable

Type	Homing	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>HOMBAC<b>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	b = 0 (disable) or 1 (enable)		
Default	0		
Response	HOMBAC: *HOMBAC0		
See Also	HOM, HOMA, HOMAA, HOMAD, HOMADA, HOMDF, HOMEDG, HOMV, HOMVF, HOMZ		

---

The 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

Type	Homing	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>HOMDF<b>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	b = 0 (positive-direction) or 1 (negative-direction)		
Default	0		
Response	HOMDF: *HOMDF0		
See Also	HOM, HOMA, HOMAA, HOMAD, HOMADA, HOMBAC, HOMEDG, HOMV, HOMVF, HOMZ		

---

The HOMDF command specifies the direction 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.

---

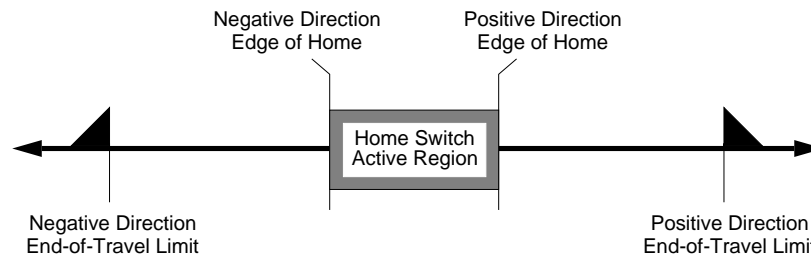
## HOMEDG Home Reference Edge

Type	Homing	Product	Rev
Syntax	<a_><!>HOMEDG<b>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	b = 0 (positive-direction edge) or 1 (negative-direction edge)		
Default	0		
Response	HOMEDG: *HOMEDG0		
See Also	HOM, HOMA, HOMAA, HOMAD, HOMADA, HOMBAC, HOMDF, HOMV, HOMVF, HOMZ, INFNC, INLVL		

---

The HOMEDG command specifies which edge of the home switch the homing operation will consider as its final destination.

As illustrated below, the positive-direction edge of the home switch is defined as the first switch transition seen by the Gem6K when traveling off of the positive-direction end-of-travel limit in the negative direction. The negative-direction edge of the home switch is defined as the first switch transition seen by the Gem6K when traveling off of the negative-direction end-of-travel limit in the positive-direction. This command is operational when backup to home (HOMBAC) is enabled.



**Example:** Refer to the go home (HOM) command example.

---

## HOMV Home Velocity

Type	Homing	Product	Rev
Syntax	<a_><!>HOMV<r>	GT6K	6.0
Units	r = units/sec (scalable with SCLV) (linear motors: see DMEPIT for linear/rotary conversion)	GV6K	6.0
Range	Stepper Axes: 0.0000-60.0000 (after conversion to revs/sec) Servo Axes: 0.0000-200.0000 (after conversion to revs/sec; if ERES > 10,000, maximum = 2,000,000/ERES)		
Default	1.0000		
Response	HOMV: *HOMV1.0000		
See Also	DMEPIT, HOM, HOMA, HOMAA, HOMAD, HOMADA, HOMBAC, HOMDF, HOMEDG, HOMVF, HOMZ, SCALE, SCLV		

---

The 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.

**UNITS OF MEASURE and SCALING:** refer to page 16.

**Example:** Refer to the go home (HOM) command example.

---

## HOMVF Home Final Velocity

Type	Homing	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>HOMVF<r>	GT6K	6.0
Units	r = units/sec (scalable with SCLV) (linear motors: see DMEPIT linear/rotary conversion)	GV6K	6.0
Range	Stepper Axes: 0.0000-60.0000 (after conversion to revs/sec) Servo Axes: 0.0000-200.0000 (after conversion to revs/sec; if ERES > 10,000, maximum = 2,000,000/ERES)		
Default	0.1000		
Response	HOMVF: *HOMVF0.1000		
See Also	DMEPIT, HOM, HOMA, HOMAA, HOMAD, HOMADA, HOMBAC, HOMDF, HOMEDG, HOMV, HOMZ, SCALE, SCLV		

---

The 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.

<b>UNITS OF MEASURE and SCALING:</b> refer to page 16.
--

**Example:** Refer to the go home (HOM) command example.

---

## HOMZ Home to Encoder Z-channel Enable

Type	Homing	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>HOMZ<b>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	b = 0 (disable) or 1 (enable)		
Default	0		
Response	HOMZ: *HOMZ0		
See Also	[ ASX ], HOM, HOMA, HOMAA, HOMAD, HOMADA, HOMBAC, HOMDF, HOMEDG, HOMV, HOMVF, INFNC, INLV, TASX		

---

The HOMZ command enables homing to an encoder z-channel after the initial home input has gone active. NOTE: The home limit input is required to go active prior to homing to the Z channel. The state of the Z-channel is reported with bit 6 of the ASX and TASX register.

**Example:** Refer to the go home (HOM) command example.

---

## IF( ) IF Statement

Type	Program Flow Control or Conditional Branching	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>IF(expression)	GT6K	6.0
Units	n/a	GV6K	6.0
Range	Up to 80 characters (including parentheses)		
Default	n/a		
Response	n/a		
See Also	ELSE, NIF		

---

The IF 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, and the commands after the ELSE until the NIF are ignored. False IF evaluations cause commands between the ELSE and the NIF to be executed, and the commands between the IF and the ELSE are 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 a 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, AS, ASX, D, ER, IN, MOV, OUT, PC, PCE, PCM, PE, PER, PMAS, SEG, SS, TIM, US, V, VEL, VELA, etc.) can be used within the IF expression.

Multiple parentheses may not be used within the IF command.

**Example:**

```
IF(IN=b1X0 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
                                ; Transfer revision level
                                ; End IF statement
    TREV                          ;
    NIF                            ;
IF(A<5000 AND PC>50000)         ; If the acceleration is less than 5000, and
                                ; the commanded position is greater than
                                ; 50000, then do the IF statement. Note: The
                                ; acceleration value used is programmed acceleration,
                                ; not actual.
                                ; Increment variable 1
                                ; End if statement
    VAR1=VAR1+1                    ;
    NIF                            ;
IF(VEL<123 OR VEL>156)         ; If the current velocity is less than 123
                                ; or if it is greater than 156, then do the commands
                                ; following the IF statement
                                ; Put message Something's Wrong<cr> in output buffer
                                ; End if statement
    WRITE"Something's Wrong\13" ;
    NIF                            ;
IF(OUT=b110X1 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
                                ; End IF statement
    VAR1=VAR1+1                    ;
    ELSE                            ;
    VAR1=VAR1-1                    ;
    NIF                            ;
```

## IN Virtual Input Override

		Product	Rev
Type	Inputs	GT6K	6.0
Syntax	<a_><!><B>IN<=xx>	GV6K	6.0
Units	B = I/O brick number xx = 32-bit data operand (see list below)		
Range	B = 1-8		
Default	n/a		
Response	n/a		
See Also	[ IN ], INDEB, INEN, INFNC, INPLC, INSTW, TIN, TIO		

The IN command allows you to substitute almost any 32-bit data parameter as a virtual input brick of 32 inputs. The virtual inputs behave similarly to real inputs in that they are affected by INEN and INLVL, and they affect INFNC, INPLC, INSTW, INDUST, ONIN, and GOWHEN (<B>IN=b<bbbb>) commands. Unlike real inputs, virtual inputs are not affected by the INDEB debounce setting.

The data operands allowed for virtual input assignments are listed below (integer values are converted to binary):

A ..... Acceleration	NMCY ..... Master cycle number	SWAP ..... Task swap assignment
AD ..... Deceleration	OUT ..... Output status	TASK ..... Task number
ANI ..... Analog input voltage	PC ..... Commanded position	TIM ..... Timer value
ANO ..... Analog output voltage	PCC ..... Captured command pos.	TRIG ..... Trigger interrupt status
AS ..... Axis status	PCE ..... Captured encoder pos.	US ..... User-defined status
ASX ..... Extended axis status	PCME ..... Captured master enc. pos.	V ..... Velocity
D ..... Distance	PE ..... Encoder position	VARI ..... Integer variable
DKEY ..... RP240 keypad value	PER ..... Position error	VARB ..... Binary variable
ER ..... Error status	PMAS ..... Position of Master	VEL ..... Commanded velocity
FB ..... Feedback device pos.	PME ..... Master encoder pos.	VELA ..... Actual velocity
FS ..... Following status	PSHF ..... Net position shift	VMAS ..... Velocity of the master
IN ..... Input status	PSLV ..... Follower pos. command	
INO ..... Enable input status	SC ..... Controller status	
LIM ..... Limit input status	SCAN ..... PLC scan time	
MOV ..... Axis moving status	SEG ..... Free segment buffers	
	SS ..... System status	

**NOTE:** A virtual input can only be defined for I/O bricks that are not connected on the serial I/O network (remember that up to 8 I/O bricks are allowed). For example, if your Gem6K unit has two I/O bricks, you can designate I/O bricks 3-8 as virtual I/O bricks.

Virtual inputs provide programming input functionality for data or external events (see operand list above) that are not ordinarily represented by inputs.

For example, suppose a PLC is sending binary data via the VARB1 command to the Gem6K. If the binary state of VARB1 is assigned to input brick 2 (2IN=VARB1), the Gem6K can respond based on programmable input functions set up with the INFNC command.

```

2TIN           ; Brick 2 is not connected; therefore, the Gem6K will
                ; respond with an error message: "*INCORRECT I/O BRICK"
2IN=VARB1      ; Map the binary state of VARB1 to be the
                ; input state of "virtual" input brick 2 (2IN)
VARB1=b1010000 ; Change "virtual" input brick 2IN to a new VARB1 value
2TIN           ; Check the input status. The response will be:
                ; "*2IN1010_0000_0000_0000_0000_0000_0000"

```



---

**[ IN ]****Input Status**

		<b>Product</b>	<b>Rev</b>
Type	Assignment or Comparison		
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	GOWHEN, IN, INFNC, [ LIM ], ONIN, TIN, VARB		

---

Use the IN operator 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=<B>IN where “n” is the binary variable number and “<B>” is the number of the I/O brick where the input resides (not required if addressing inputs on the Gem6K’s DRIVE I/O connector), or IN can be used in an expression such as IF(2IN=b11Ø1), or IF(2IN=h7F). 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=2IN.10 assigns the binary state of input 10 (2<sup>nd</sup> pin on SIM 2) on I/O brick 2 to binary variable 1.

The function of the inputs is established with the INFNC command (although the IN operator looks at any trigger or external digital input, regardless of its assigned function from the INFNC command).

**Example:**

```
VARB1=3IN          ; Input status on I/O brick 3 assigned to binary variable 1
VARB2=2IN.12       ; Input bit 12 on I/O brick 2 assigned to binary variable 2
VARB2              ; Response if bit 12 is set to 1:
                  ; *VARB2=XXXX_XXXX_XXX1_XXXX_XXXX_XXXX_XXXX_XXXX
IF(1IN=b111011X11) ; If the input status contains 1's for inputs 1,2,3,5,6,8,& 9,
                  ; and 0 for input 4 on I/O brick 1, do the commands
                  ; following the IF statement
TREV              ; Transfer revision level
NIF              ; End IF statement
IF(2IN=hEF00)     ; If the input status contains 1's for I/O brick 2's inputs
                  ; 1,2,3,5,6,7,& 8, and 0's for every other input, do the
                  ; commands following the IF statement
TREV              ; Transfer revision level
NIF              ; End IF statement
```

---

## INDEB            Input Debounce Time

Type	Input	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!><B>INDEB<i>	GT6K	6.0
Units	i = time in milliseconds (ms)	GV6K	6.0
Range	i = 2-250		
Default	4		
Response	INDEB:    *INDEB4		
See Also	INFNC, INLVL, LIMFNC, RE, REG, TIN, TLIM, TRGFN, TRGLOT		

---

The INDEB command governs the debounce time for the onboard inputs on the DRIVE I/O connector, or inputs on a specified I/O brick. The debounce time is the period of time that the input must be held in a certain state before the drive recognizes it. This directly affects the rate at which the inputs can change state and be recognized. The default setting is 4 ms.

**Exception for Trigger Inputs:** For inputs that are assigned the “Trigger Interrupt” function (INFNCi-H), the debounce is instead governed by the TRGLOT setting. The TRGLOT setting applies to all inputs defined as “Trigger Interrupt” inputs. The TRGLOT 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. The default setting is 24 ms.

**Limit Inputs.** The limit inputs found on the DRIVE I/O connector are not normally debounced; however, if a limit is assigned a different function with the LIMFNC command (other than LIMFNCi-R, LIMFNCi-S, or LIMFNCi-T), the input is debounced using the INDEB setting for the onboard inputs. If a general-purpose input is assigned a limit input function (INFNCi-R, INFNCi-S, or INFNCi-T), the input will not be debounced.

**Example:**

```
INDEB6            ; Assign all onboard inputs a debounce time of 6 ms (inputs
                  ; assigned as triggers with INFNCi-H will not be debounced)
2INDEB10         ; Assign inputs on I/O brick 2 a debounce time of 10 ms
1INDEB12         ; Assign inputs on I/O brick 1 a debounce time of 12 ms
```

---

## INDUSE            Enable/Disable User Status

Type	Drive Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>INDUSE<b>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	b = 0 (disable) or 1 (enable)		
Default	0		
Response	INDUSE:    *INDUSE0		
See Also	INDUST, ONUS, TUS, [ US ]		

---

The 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:**

```
INDUSE1            ; Enable user status
```

<b>INDUST User Status Definition</b>		<b>Product</b>	<b>Rev</b>
Type	Drive Configuration	GT6K	6.0
Syntax	<a_><!><B>INDUST<i><-<i><c>>	GV6K	6.0
Units	See description below		
Range	1st i = 1 - 16; 2nd i = 1 - 32; c = A through S		
Default	See description below		
Response	INDUST: *INDUST1-1A AXIS 1 STATUS - STATUS OFF (...repeated for all 16 user status bits...) *INDUST16-16A AXIS 1 STATUS - STATUS OFF INDUST1: *INDUST1-1A AXIS 1 STATUS - STATUS OFF		
See Also	[ AS ], [ ASX ], [ IN ], INDUSE, ONUS, [ SS ], TAS, TASX, TIN, TSS, TUS, [ US ]		

The **INDUST** command establishes the user status bit function. Each bit can correspond to an axis status bit, a system status bit, an input, an interrupt bit, or an extended axis status bit. The default for each user status bit is as follows:

Default for the Gem6K product (first two AS status bits):

Bits 1-16 = first 16 bits of axis status (AS)

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 **<B>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 axis status (AS), the system status (SS), the input status (IN), or the extended axis status (ASX).
- The **<c>** defines what status to use:

<b>&lt;c&gt;</b>	<b>Value</b>	<b>Function</b>	<b>&lt;c&gt;</b>	<b>Value</b>	<b>Function</b>
	A	Use axis status (AS)		K	RESERVED
	B - H	RESERVED		L	Use extended axis status (ASX)
	I	Use system status (SS) *		M - S	RESERVED
	J	Use input status (IN) **			

\* If you are using multitasking, the "I" value requires you to prefix the **INDUST** command with the task identifier (e.g., `2%INDUST6-2I` assigns system status bit 2 for task 2 to user status bit 6). If no task prefix is given, the system status for task zero is used by default.

\*\* The "J" value requires you to prefix the **INDUST** command with the I/O brick identifier (e.g., `2INDUST14-4J` assigns the status of I/O point on I/O brick 2 to user status bit 14). If no brick prefix is given, the onboard inputs are referenced by default. Refer to page 8 to fully understand the I/O bit patterns and use of the brick identifier.

**Example**

```
INDUSE1           ; Enable user status
INDUST1-5A       ; User status bit 1 defined as axis status bit 5
INDUST2-3A       ; User status bit 2 defined as axis status bit 3
3INDUST3-5J      ; User status bit 3 defined as input 5 on I/O brick 3
2%INDUST16-2I    ; User status bit 16 defined as system status bit 2 for task 2
```

---

## INEN

### Input Enable

Type	Input or Program Debug Tool	Product	Rev
Syntax	<a_><!><@><B>INEN<d><d>...<d> (one <d> for each input)	GT6K	6.0
Units	n/a	GV6K	6.0
Range	d = 0 (disable, leave off), 1 (disable, leave on), E (enable), or X (don't change)		
Default	E		
Response	INEN: *INENEEEE_E 1INEN: *1INENEEEE_EEEE_EEEE_EEEE_EEEE_EEEE_EEEE_EEEE 1INEN.3 *E		
See Also	ERROR, IN, [ IN ], INFNC, INLVL, INPLC, INSTW, KIOEN, LH, LIMEN, TIN, TIO, TSTAT		

---

The INEN command allows you to simulate the activation of specific general-purpose inputs or external digital inputs (without actually wiring the inputs) by disabling them and setting them to a specific level (ON or OFF).

The default INEN condition is enabled (E), requiring external wiring to exercise the input's respective INFNC function.

**Using Inputs on Expansion I/O Bricks:** If the I/O brick is disconnected or if it loses power, the drive will perform a kill (all tasks) and set error bit #18 (see ERROR). (If you disable the “Kill on I/O Disconnect” mode with KIOENØ, the Gem6K will not perform the kill.) The drive will remember the brick configuration (volatile memory) in effect at the time the disconnection occurred. When you reconnect the I/O brick, the drive checks to see if anything changed (SIM by SIM) from the state when it was disconnected. If an existing SIM slot is changed (different SIM, vacant SIM slot, or jumper setting), the drive will set the SIM to factory default INEN and OUTLVL settings. If a new SIM is installed where there was none before, the new SIM is auto-configured to factory defaults.

**Example:** 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.

**INEN has no effect on ...**

- Inputs when they are configured as “trigger interrupt” inputs with the INFNCi-H command. This includes position capture and registration functions.
  - Inputs configured as “end-of-travel limit” inputs with the INFNCi-R or INFNCi-S commands. Instead, use the LH command.
- 

Input bit assignments for the INEN command vary by external I/O brick configuration. The input bit patterns for general-purpose inputs and external I/O bricks are explained on page 8 of this document.

**Example:**

```
DEF tester          ; Begin definition of program tester
WHILE(IN=b11X10)   ; While inputs 1, 2, and 4 are active, and input 5 is not
                   ; active, execute the statements between the WHILE & NWHILE
GO1                ; Initiate motion
NWHILE             ; End WHILE statement
END               ; End definition of program tester
INEN11X10         ; Disable inputs 1,2,4, & 5, and set inputs 1, 2 & 4 in
                   ; the active state, and input 5 in the inactive state
RUN tester        ; Initiate program tester
!INEN00000        ; Disable onboard inputs 1,2,3,4, & 5, and leave them in the
                   ; inactive state
INENeeeeee        ; Re-enable inputs 1 through 5
```

<b>INFNC</b>		<b>Input Function</b>	
Type	Input	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!><B>INFNC<i>-<<a>c>	GT6K	6.0
Units	i = input #, a = program # for function P; c = function identifier letter	GV6K	6.0
Range	i = 1-32 (I/O brick dependent – see page 8); c = A-T		
Default	All inputs set to A (general-purpose)		
Response	INFNC: (input function and status of general-purpose inputs) 1INFNC: (input function and status of I/O brick 1 inputs) 1INFNC1: *1INFNC1-A NO FUNCTION - STATUS OFF		
See Also	COMEXR, COMEXS, ENCCNT, [ ER ], ERROR, [ IN ], INDEB, INEN, INLVL, INPLC, INSELP, INSTW, INTHW, JOY, JOYAXH, JOYVH, JOYVL, K, KDRIVE, LH, LIMFNC, PSET, [ SS ], TER, TIN, TIO, TRGFN, TRGLOT, [ TRIG ], TSS, TSTAT, TTRIG		

The INFNC command defines the function of each individual input, where *i* is the input bit number, or the program number for the case of input function P, and *c* is the function. A limit of 32 inputs may be assigned INFNC functions; this excludes functions A (“general-purpose”) and H (“trigger interrupt”).

**Input Debounce.** Using the Input Debounce Time (INDEB) command, you can change the input debounce time for all general-purpose inputs. The debounce is the period of time that the input must be held in a certain state before the Gem6K recognizes it. This directly affects the rate at which the inputs can change state and be recognized. Inputs that are assigned the “Trigger Interrupt” function (INFNCi-H), are instead debounced by the TRGLOT value. Inputs defined as limit inputs (INFNCi-R, INFNCi-S, or INFNCi-T), will not be debounced.

**Input bit assignments vary.** The input bit patterns for onboard and external I/O bricks are explained on page 8 of this document.

**Input Scan Rate.** The programmable inputs are scanned once per *system update* (2 milliseconds).

**Multitasking.** If the INFNC command does not include the task identifier (%) prefix, the function affects the task that executes the INFNC command. The functions that may be directed to a task with % are: C, D, E, F, and P (e.g., 2%INFNC3-F assigns onboard input 3 as a user fault input for task 2). Multiple tasks may share the same input, but the input may only be assigned one function.

Identifier	Function Description																				
A	<b>General Purpose Input.</b> Normal input, used with the IN assignment																				
B	<b>BCD Program Select.</b> BCD input assignment to programs, lowest numbered input is least significant bit (LSB). BCD values for inputs are as follows:																				
	<table> <thead> <tr> <th></th> <th>BCD Value</th> </tr> </thead> <tbody> <tr> <td>Least Significant Bit Value</td> <td>1</td> </tr> <tr> <td>.</td> <td>2</td> </tr> <tr> <td>.</td> <td>4</td> </tr> <tr> <td>.</td> <td>8</td> </tr> <tr> <td>.</td> <td>10</td> </tr> <tr> <td>.</td> <td>20</td> </tr> <tr> <td>.</td> <td>40</td> </tr> <tr> <td>.</td> <td>80</td> </tr> <tr> <td>Most Significant Bit Value</td> <td>100</td> </tr> </tbody> </table>		BCD Value	Least Significant Bit Value	1	.	2	.	4	.	8	.	10	.	20	.	40	.	80	Most Significant Bit Value	100
	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.

- | Identifier | Function Description  |
|------------|---|
| C          | <b>Kill.</b> Kills motion and halts all command processing (refer to K and KDRIVE command descriptions for further details on the <i>kill</i> 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          | <b>Stop.</b> Stops motion. If COMEXS is set to zero (COMEXS0), program execution will be terminated. If COMEXS is set to 1 (COMEXS1), command processing will continue. With COMEXS set to 2 (COMEXS2), program execution is terminated, but the INSELP value is retained. Motion deceleration during the stop is controlled by the AD & ADA commands. If error bit #8 is enabled (e.g., ERROR.8-1), activating a Stop input will set the error bit and cause a branch to the ERRORP program. Activating a Stop input does not return a prompt to the terminal. |
| E          | <b>Pause/Continue.</b> If COMEXR is disabled (COMEXR0), 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 (and motion in COMEXR1 mode).  |
| F          | <b>User Fault.</b> Refer to the ERROR command. If error bit #7 is enabled (e.g., ERROR.7-1), activating a User Fault input will set the error bit and cause a branch to the ERRORP program.<br><b>CAUTION:</b> Activating the user fault input sends a !K command to the drive, “killing” motion (refer to the K command description for ramifications).  |
| G          | <b>Reserved</b>   |
| H          | <b>Trigger Interrupt</b> - This function can only be assigned to the general-purpose inputs on the DRIVE I/O connector, as specified below:   |

Input #	Pin #	GT6K & GV6K Function (INFNC default)
1	37	INFNC1-A (General-purpose. If assigned, TRG-A must be on Input #1)
2	38	INFNC2-A (General-purpose. If assigned, TRG-B must be on Input #2)
3	39	INFNC3-A (General-purpose)
4	34	INFNC4-A (General-purpose)
5	35	INFNC5-A (General-purpose. If assigned, MASTER TRIGGER must be on Input #5)

A “Trigger Interrupt” input can be used for these purposes:

- **Position Capture.** You can assign up to two trigger inputs, referred to as “TRIG-A” and “TRIG-B. If assigned, TRIG-A must be on onboard input #1 (pin 37). TRIG-B must be on onboard input #2 (pin 38). These inputs are located on the 25-pin DRIVE I/O connector. To assign TRIG-A and TRIG-B, use the INFNC1-H and INFNC2-H commands, respectively.

When assigned the Trigger Interrupt function, activating the input performs a *hardware capture* of axis position. If the axis is used as a follower in Following, activating the trigger also performs an *interpolated capture* of the associated master axis position.

You can assign an additional “Master Trigger”, referred to as “TRIG-M”. If assigned, TRIG-M must be on onboard input #5 (pin 35). TRIG-M may be used to perform a hardware capture of the “MASTER ENCODER” (the encoder connected to the “Master Encoder” connector), as well as axis position (encoder position on servo axes; commanded or encoder position for steppers, depending on the ENCCNT setting). To assign trigger interrupt input TRIG-M to input #5, use the INFNC5-H command.

When a Trigger Interrupt input is activated, the drive captures the relevant positions and stores them in registers that are available at the next system update (2 ms) through the use of these transfer and assignment/comparison commands:

Captured Information	Transfer	Assignment/Comparison	Offset *	Scale Factor **
Commanded position	TPCC	PCC	PSET	SCLD
Encoder position	TPCE	PCE	PSET or PESET	SCLD
Master encoder position	TPCME	PCME	PMESET	SCLMAS
Master cycle position	TPCMS	PCMS	PSET	SCLMAS

\* Captured values are offset by any existing PSET or PMESET offset.

\*\* If scaling is enabled, the captured position is scaled by SCLD or SCLMAS.

*More about Trigger Interrupt function on next page ...*

H (con't.) *Continued from previous page (Trigger Interrupt function):*

**NOTES ABOUT POSITION CAPTURE:**

- Hardware Capture: The encoder position is captured within  $\pm 1$  encoder count. The commanded position capture accuracy is  $\pm 1$  count.
- Interpolated Capture: There is a time delay of up to 50  $\mu$ s between activating the trigger interrupt input and capturing the position; therefore, the accuracy of the captured position is equal to 50  $\mu$ s multiplied by the velocity of the axis at the time the input was activated.
- Servo vs. Stepper. The nature of the axis position captured with a Trigger Interrupt input may be different, depending on whether the drive is a servo (GV6K) or stepper GT6K. For servo axes, both the commanded and encoder position for the axis are captured. For stepper axes, if the ENCCNT command is set to ENCCNT0 (default condition), only the commanded position is captured. If ENCCNT1 mode is enabled, only the encoder position is captured.

- **Registration.** (see RE description for details)
- **Special trigger functions** defined with the TRGFN command (see TRGFN for details).

**NOTES ABOUT TRIGGER INTERRUPT INPUTS:**

- When an input is assigned the "Trigger Interrupt" function, the debounce is governed by the TRGLOT command setting (default is 24 ms). The TRGLOT setting overrides the existing INDEB setting for only the inputs that are assigned the "Trigger Interrupt" function.
- When an input is assigned the "Trigger Interrupt" function, the input cannot be affected by the input enable (INEN) command.
- Trigger Interrupt Status: Use the TTRIG and TRIG commands to ascertain if a trigger interrupt input has been activated. TTRIG displays the status as a binary report, and TRIG is an assignment/comparison operator for using the status information in a conditional expression (e.g., in an IF statement). The TTRIG/TRIG bits are cleared when the respective captured position is read (see table above).

- I **Alarm Event** - Will cause the Gem6K drive to set an Alarm Event in the Communications Server over the Ethernet interface. You must first enable the Alarm checking bit for this input-driven alarm (INTHW. 23-1). For details on using alarms, refer to the *Programmer's Guide*.
- J **JOG positive-direction** - Will jog the axis in a positive direction. The JOG command must be enabled for this function to work.
- K **JOG negative-direction.** Will jog the axis in a negative direction. The JOG command must be enabled for this function to work.
- 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.
- M **Joystick Release.** Signals the drive to end joystick operation and resume program execution with the next statement in your program. When the input is open (high), the joystick mode is disabled (joystick mode can be enabled only if the input is closed, and only with the JOY command). When the input is closed (low), joystick mode can be enabled with the JOY command. The process of using Joystick mode is:
  1. Assign the "Joystick Release" input function to a programmable input.
  2. At the appropriate place in the program, enable joystick control of motion (with the JOY command). (Joystick mode cannot be enabled unless the "Joystick Release" input is closed.) When the JOY command enables joystick mode, program execution stops (assuming the Continuous Command Execution Mode is disabled with the COMEXCØ command).
  3. Use the joystick to move the axis as required.
  4. When you are finished using the joystick, open the "Joystick Release" input to disable the joystick mode. This allows program execution to resume with the next statement after the initial JOY command that started the joystick mode.

N **Reserved**

Identifier	Function Description
O	<p><b>Joystick Velocity Select.</b> Allows you to select the velocity for joystick motion. The JOYVH and JOYVL commands establish two joystick velocities. Opening the Velocity Select input (input is high) selects the JOYVH configuration. Closing the Velocity Select input (input is low) selects the JOYVL configuration. The JOYVL velocity could be used to quickly move to a location, the JOYVH velocity could be used for low-speed accurate positioning. NOTE: When this input is not connected, joystick motion always uses the JOYVH velocity setting.</p>
aP	<p><b>Program Select.</b> 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. For example, in the 2INFNC1-3P command, 3 is the program number. An input defined as a Program Select Input will not function until the INSELP command has been enabled.</p>
Q	<p><b>Program Security.</b> Issuing the INFNCi-Q command enables the <i>Program Security</i> feature and assigns the <i>Program Access</i> function to the specified programmable input.</p> <p>The program security feature denies you access to the DEF, DEL, ERASE, MEMORY, INFNC and LIMFNC 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. <i>The INFNCi-Q command is not saved in battery-backed RAM, so you may want to put it in the start-up program (STARTP).</i></p> <p>For example, once you issue the 3INFNC12-Q command, the input on the 4<sup>th</sup> pin on SIM2 (I/O point 12) of I/O brick 3 is assigned the program access function and access to the DEF, DEL, ERASE, MEMORY, INFNC, and LIMFNC commands will be denied until you activate the input.</p> <p>To regain access to these commands without the use of the program access input, you must issue the INEN command to disable the program security input, make the required user memory changes, and then issue the INEN command to re-enable the input. For example, if input 3 on brick 2 is assigned as the Program Security input, use 2INEN. 3=1 to disable the input and leave it activated, make the necessary user memory changes, and then use 2INEN. 3=E to re-enable the input.</p>
R	<p><b>End-of-Travel Limit, Positive Direction.</b> This assigns the positive direction end-of-travel limit input function. For example, INFNC1-R assigns the "Positive EOT limit" function to general-purpose input #1. "Positive direction" correlates to motion in the positive-counting direction as reported with TPE and TPC. <b>REMEMBER:</b> Once an input is assigned a limit function, it is no longer debounced (INDEB has no effect), and it must be enabled/disabled with the LH command instead of the INEN command.</p>
S	<p><b>End-of-Travel Limit, Negative Direction.</b> This assigns the negative direction end-of-travel limit input function. For example, INFNC2-S assigns the "Negative EOT limit" function to general-purpose input #2. "Negative direction" correlates to motion in the negative-counting direction as reported with TPE and TPC. <b>REMEMBER:</b> Once an input is assigned a limit function, it is no longer debounced (INDEB has no effect), and it must be enabled/disabled with the LH command instead of the INEN command.</p>
T	<p><b>Home Limit.</b> This assigns the home limit input function. For example, INFNC3-T assigns the "Home limit" function to general-purpose input #3. <b>REMEMBER:</b> Once an input is assigned a limit function, it is no longer debounced. For more information on homing, refer to the HOM command and the Homing section in the <i>Programmer's Guide</i>.</p>

**Example:**

3INFNC1-D ; Input at I/O point #1 on brick 3 is defined to be a stop input



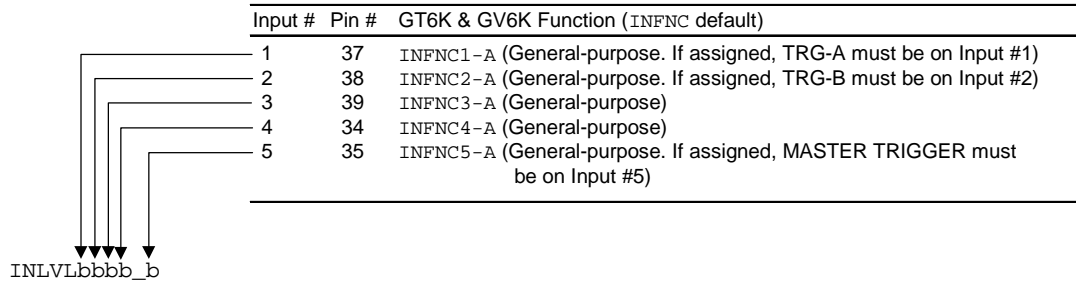
## INLVL

### Input Active Level

Type	Input	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!><@><B>INLVL<b><b>...<b><b>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	b = 0 (active low), 1 (active high, or X (don't change))		
Default	0		
Response	INLVL: *INLVL0000_0 1INLVL: *1INLVL0000_0000_0000_0000_0000_0000_0000_0000 1INLVL.3: *0 (active low)		
See Also	INDEB, INEN, INFNC, INPLC, INSTW, TIN, TIO		

The INLVL command defines the active state of the general-purpose inputs on the DRIVE I/O connector, and on external I/O bricks. Example: INLVL11000 sets the active level for onboard inputs 1 & 2 to active high, all other onboard inputs are set to active low.

INLVL bit assignments for inputs on the DRIVE I/O connector (bits are numbered 1-5 from left to right):



### Active Level Relationships (inputs on the DRIVE I/O connector):

Active Level	Sinking/Sourcing *	Switch	TIN and IN status
INLVL0 (active low)	Sourcing	Closed (connected to ground)	1
INLVL0 (active low)	Sourcing	Open	0
INLVL1 (active high)	Sourcing	Closed (connected to ground)	0
INLVL1 (active high)	Sourcing	Open	1
INLVL0 (active low)	Sinking	Closed (connected to +V)	0
INLVL0 (active low)	Sinking	Open	1
INLVL1 (active high)	Sinking	Closed (connected to +V)	1
INLVL1 (active high)	Sinking	Open	0

\* The inputs are factory configured to source current. If you wish the inputs to sink current, connect the pull-up terminals (pins 27 and 33) on the DRIVE I/O connector to ground (see the *Hardware Installation Guide* for wiring instructions). Pin 27 is the pull up for limit inputs 1-3, and pin 33 is the pull up for general-purpose inputs 1-5.

### Active Level Relationships (external SIM8-IN-EVM32 inputs on an EVM32 I/O brick):

Active Level *	Jumper Selection *	Switch	Voltage at Input	LED	IN/TIN/TIO Report
INLVL0 (active low)	Position 3 (sourcing)	Open	≥ 2/3 of V+	On	0
INLVL0 (active low)	Position 3 (sourcing)	Closed	< 1/3 of V+	Off	1
INLVL1 (active high)	Position 3 (sourcing)	Open	≥ 2/3 of V+	On	1
INLVL1 (active high)	Position 3 (sourcing)	Closed	< 1/3 of V+	Off	0
INLVL0 (active low)	Position 1 (sinking)	Open	< 1/3 of V+	Off	1
INLVL0 (active low)	Position 1 (sinking)	Closed	≥ 2/3 of V+	On	0
INLVL1 (active high)	Position 1 (sinking)	Open	< 1/3 of V+	Off	0
INLVL1 (active high)	Position 1 (sinking)	Closed	≥ 2/3 of V+	On	1

\* Factory default: INLVL0 (active low) and jumper in position 3 (sourcing). Move jumper to position 1 for sinking.

### Example:

```
2INLVL0101 ; Set active level for these inputs on I/O brick 2:
; inputs 1 & 3 are active low, inputs 2 & 4 are active high.
```

---

## [ INO ] Other Input Status

Type	Assignment or Comparison	<b>Product</b>	<b>Rev</b>
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	[ IN ], [ LIM ], TINO, TINOF		

---

Use the INO command 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. To assign only one bit (one specific input) value to a binary variable, instead of all 8, use the bit select (.) operator . For example, VARB1=INO.6 assigns the status of the ENABLE input to binary variable 1.

Format for binary assignment:                    0000\_0000  
  ↑  ↑  
  Bit #1  Bit #8

Bit	Function	Location
1-5	RESERVED	
6	Enable input (1 = OK for motion)	DRIVE I/O connector
7-8	RESERVED	

---

### Example:

```
VARB2=INO.6      ; ENABLE input status assigned to binary variable 2
VARB2            ; Response if bit 6 is set to 1: *VARB2=XXXX_X1XX
IF(INO.6=b1)    ; If ENABLE input status is 1 (OK for motion), 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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!><B>INPLC<i>,<i-i>,<i>,<i>	GT6K	6.0
Units	See below	GV6K	6.0
Range	See below		
Default	1,0-0,0,0		
Response	INPLC1: *INPLC1,0-0,0,0 1INPLC1: *1		
See Also	INEN, INFNC, INLVL, INSTW, OUTPLC, [ TW ]		

---

The INPLC command, in combination with the OUTPLC command, configures 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>	<b>Set #:</b> There are 4 possible INPLC sets (1-4). This field identifies which set to use.
Field 2: <i-i>	<b>Input #s:</b> Data is read into the Gem6K through programmable inputs on expansion I/O bricks. This field identifies the 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. You must use inputs on an expansion I/O brick. Refer to page 8 for help in identifying which input bits are available to place in this field.
Field 3: <i>	<b>Sign Input #:</b> 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>	<b>Data Valid Input #:</b> 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,0-0,0,0 where n is the PLC set (1-4).

**Example:**

```

1INPLC2,1-8,9,10 ; Set INPLC set 2 as BCD digits on inputs 1-8 on I/O brick 1,
                  ; with input 9 as the sign bit, and input 10 as the data valid
1OUTPLC2,1-4,5,50 ; Set OUTPLC set 2 as output strobes on outputs 1-4 on I/O
                  ; brick 1, with output 5 as the output enable bit, and
                  ; strobe time of 50 milliseconds
A(TW6)           ; Read data into acceleration using INPLC set 2 and
                  ; OUTPLC set 2 as the data configuration

```

## INSELP Select Program Enable

Type	Input	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>INSELP<i>, <i>	GT6K	6.0
Units	See below	GV6K	6.0
Range	1st i = 0, 1, or 2; 2nd i = 0 - 5000		
Default	0,0		
Response	INSELP: *INSELP0,0		
See Also	COMEXS, INEN, INFNC, INLVL, INPLC, INSTW, [ SS ], TDIR, TSS		

The INSELP command enables program selection by inputs. In addition, the command establishes the strobe time for the inputs, and whether 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 (0 = Disable, 1 and 2 = Enable). Use INFNCi-B inputs if i = 1; use 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 inputs are scanned once per *system update* (2 milliseconds).

The Kill (!K) command releases this mode, in addition to INSELPØ. The Stop (!S) command or an input defined as a stop input (INFNCi-D) will also release this mode, as long as COMEXS has been disabled.

**Example:**

```
2INFNC1-1P      ; Input #1 on I/O brick 2 defined to select program #1
2INFNC2-2P      ; Input #2 on I/O brick 2 defined to select program #2
2INFNC3-7P      ; Input #3 on I/O brick 2 defined to select program #7
INSELP2,50      ; Enable continuous scan of inputs to select a program to run
```

---

## INSTW Establish Thumbwheel Data Inputs

Type	Input	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!><B>INSTW<i>,<i-i>,<i>	GT6K	6.0
Units	See below	GV6K	6.0
Range	See below		
Default	1,0-0,0		
Response	INSTW1: *INSTW1,0-0,0 1INSTW1: *1INSTW1,0-0,0		
See Also	INEN, INFNC, INLVL, INPLC, OUTTW, [ SS ], TSS, [ TW ]		

---

The INSTW command, in combination with the OUTTW command, configures the inputs and outputs to read data from an active thumbwheel device. 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>	<b>Set #:</b> There are 4 possible INSTW sets (1-4). This field identifies which set to use.
Field 2: <i-i>	<b>Input #s:</b> Data is read into the Gem6K through the programmable inputs. This field identifies the 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. Refer to page 8 for help in identifying which input bits are available to place in this field.
Field 3: <i>	<b>Sign Input #:</b> 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:**

```
3INSTW2,1-4,5   ; Set INSTW set 2 as BCD digits on inputs 1-4 on I/O brick 3,
                 ; with input 5 as the sign bit
2OUTTW2,1-3,4,50 ; Set OUTTW set 2 as output strobes on outputs 1-3 on I/O
                 ; brick 2, with output 4 as the output enable bit, and
                 ; strobe time of 50 milliseconds
A(TW2)          ; Read data into acceleration using INSTW set 2
                 ; and OUTTW set 2 as the data configuration
```

---

## INTHW Check for Alarm Events

Type	Alarm Event	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>INTHW<b><b>...<b><b> (one b for each of 32 interrupts)	GT6K	6.0
Units	n/a	GV6K	6.0
Range	b = 0 (disable), 1 (enable), or X (don't change)		
Default	0		
Response	INTHW: *INTHW0000_0000_0000_0000_0000_0000_0000_0000		
See Also	[ ER ], INFNC, INTSW, LIMFNC, TER, TIMINT		

---

Use the INTHW command to determine which conditions will cause an alarm event in the Gem6K Communications Server ([this requires an Ethernet interface to the Gem6K](#)). The alarm bit in the Gem6K is cleared as soon as the alarm occurs, but the status of the alarm remains available, through the



**Example:**

```

INTHW1      ; Enable alarm event bit #1
A20         ; Set acceleration to 20 units/sec/sec
V2          ; Set velocity to 2 units/sec
D25000     ; Set move distance to 25000 units
GO1         ; Initiate motion
INTSW1      ; Force alarm event bit #1 as soon as the move is finished
; *****
; * Note: After the alarm occurs, it is the client application program's *
; * responsibility to examine the communication server's fast status *
; * register to determine the cause of the alarm. *
; *****

```

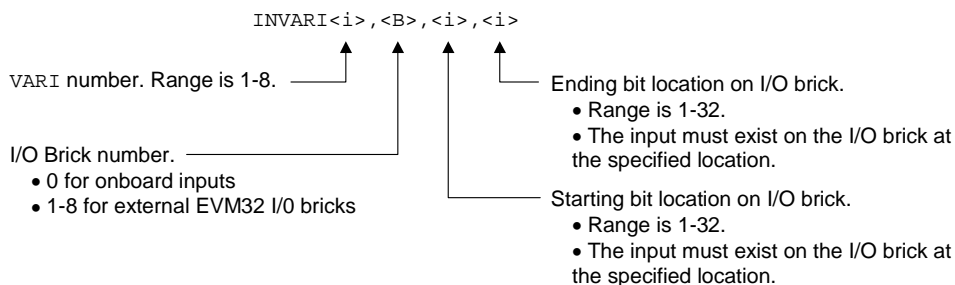
---

**INVARI**      **Map Inputs to Integer Variable**

Type	Variables	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>INVARI<i>,<b>,<i>,<i>	GT6K	6.0
Units	See below	GV6K	6.0
Range	See below		
Default	n/a		
Response	INVARI: *INVARI1,1,1,12		
See Also	INDEB, INEN, INLVL, VARI		

---

The INVARI command allows a selected group of contiguous inputs to be interpreted as a binary number and continuously assigned to the selected integer variable (VARI). The inputs are specified by I/O brick number, starting bit number, and ending bit number. All inputs to be mapped to a VARI must be contiguous and on the same I/O brick. These inputs are read and masked internally, then shifted such that the starting bit is the low-order bit of the resulting binary value. A change in the starting bit input will always result in a change of ±1 in the resulting VARI, even if the starting bit number is not 1.



The Gem6K inputs are read every 2 milliseconds, modified by INLVL and INEN, and debounced with the time specified in INDEB. The inputs specified by INVARI are monitored after they are modified by INLVL and INEN, but before the debounce. Thus, the VARI variable specified by the INVARI command will be updated every 2 milliseconds.

The VARI variables are not updated with inputs unless the INVARI is issued with valid values for starting and ending bits. If the specified input bit does not exist onboard or on the I/O brick, an error message (“\*INVALID DATA”) will result.

Specifying bit 0 for both starting and ending bit disables the INVARI mapping (INVARI<i>,<b>,0,0).

**Example:**

A following application requires an axis to follow a master source that has position information presented as a 12-bit binary number on digital outputs. An external I/O brick with two 8-bit digital input SIMs is used to read the 12 bits of the source. The source is wired to bits 1-12.

```
DEF SETUP
INVARI4,1,1,12      ; VARI4 reflects bits 1-12 of brick 1
FOLMAS48           ; Axis follows VARI4
FOLEN1            ; Enable Following mode.
END
```

**Test:** (Assert bits 2 and 3)

```
>VARI4
*VARI4=+6
>TPMAS
*TPMAS+6
```

**JOG****Jog Mode Enable**

Type	Jog	Product	Rev
Syntax	<a_><!>JOG<b>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	b = 0 (disable) or 1 (enable)		
Default	0		
Response	JOG: *JOG0		
See Also	DJOG, JOGA, JOGAA, JOGAD, JOGADA, JOGVH, JOGVL, INFNC		

The JOG1 command enables jog mode. Once jog mode has been enabled, the jog inputs can be used to produce motion. 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 JOG0 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 the axis. To disable the RP240 Jog Mode, use the !DJOG0 command or press the RP240's **MENU RECALL** button.

**Example:**

```
1INFNC1-L      ; Input #1 on I/O brick 1 defined as jog velocity select input
1INFNC2-J      ; Input #2 on I/O brick 1 defined as jog positive-direction input
1INFNC3-K      ; Input #3 on I/O brick 1 defined as jog negative-direction input
JOGA100        ; Jog acceleration set to 100 units/sec/sec
JOGAD200       ; Jog deceleration set to 200 units/sec/sec
JOGVH10        ; The velocity when the jog velocity select input is high is
                ; 10 units/sec
JOGVL1         ; The velocity when the jog velocity select input is low is
                ; 1 units/sec
JOG1           ; Enable jog mode. When an input occurs on input 2 or input 3,
                ; the motor will move at the appropriate jog velocity until
                ; the input is released
WAIT(1IN.4=b1) ; Wait for input #4 on I/O brick 1 to become active.
                ; Input #4 is being used as a signal to disable jog mode.
JOG0           ; Disable jog mode
```

---

## JOGA

### Jog Acceleration

		Product	Rev
Type	Jog	GT6K	6.0
Syntax	<a_><!>JOGA<r>	GV6K	6.0
Units	r = units/sec/sec		
Range	0.00001 - 39,999,998 (depending on the scaling factor)		
Default	10.0000		
Response	JOGA: *JOGA10.0000		
See Also	DJOG, JOG, JOGAA, JOGAD, JOGADA, JOGVH, JOGVL, INFNC, SCALE, SCLA		

---

The JOGA command specifies the acceleration to be used upon receiving a jog input.

<b>UNITS OF MEASURE and SCALING:</b> refer to page 16.
--

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	Jog; Motion (S-Curve)	GT6K	6.0
Syntax	<a_><!>JOGAA<r>	GV6K	6.0
Units	r = units/sec/sec		
Range	0.00001 - 39,999,998 (depending on the scaling factor)		
Default	10.00 (Default is a constant accel ramp, where JOGAA tracks JOGA. S-curve accel is achieved when JOGAA is changed independent of JOGA. To restore a constant accel ramp and tracking, set JOGAA = 0.)		
Response	JOGAA: *JOGAA10.0000		
See Also	A, ADA, JOG, JOGA, JOGAD, JOGADA, SCALE, SCLA		

---

The 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*. Refer to page 15 for details on S-curve profiling.

Scaling (SCLA) affects JOGAA the same as it does for JOGA. Refer to page 16 for details on scaling.

**Example:**

```
JOGA10          ; Sets the maximum jogging acceleration
JOGAA5          ; Sets the average jogging acceleration
```



<b>JOGAD Jog Deceleration</b>		<b>Product</b>	<b>Rev</b>
Type	Jog	GT6K	6.0
Syntax	<a_><!\>JOGAD<r>	GV6K	6.0
Units	r = units/sec/sec		
Range	0.00001 - 39,999,998 (depending on the scaling factor)		
Default	10.0000 (by default, JOGAD tracks JOGA until JOGAD is changed independent of JOGA; to restore tracking, set JOGAD = 0)		
Response	JOGAD: *JOGAD10.0000		
See Also	DJOG, JOG, JOGA, JOGAA, JOGADA, JOGVH, JOGVL, INFNC, SCALE, SCLA		

The JOGAD command specifies the deceleration to be used when a jog input is released.

**UNITS OF MEASURE and SCALING:** refer to page 16.

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 (JOGADØ), 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.

<b>JOGADA Jogging Average Deceleration</b>		<b>Product</b>	<b>Rev</b>
Type	Jog; Motion (S-Curve)	GT6K	6.0
Syntax	<a_><!\>JOGADA<r>	GV6K	6.0
Units	r = units/sec/sec		
Range	0.00001 - 39,999,998 (depending on the scaling factor)		
Default	10.00 (By default, JOGADA tracks JOGAA until JOGADA is changed independent of JOGAA. To restore tracking, set JOGAD = 0.)		
Response	JOGADA: *JOGADA10.0000		
See Also	A, AD, JOG, JOGA, JOGAA, JOGAD, SCALE, SCLA		

The 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*. Refer to page 15 for details on S-curve profiling.

Scaling (SCLA) affects JOGADA the same as it does for JOGAD. Refer to page 16 for details on scaling.

**Example:**

```
JOGAD10      ; Sets the maximum jog deceleration
JOGADA5     ; Sets the average jog deceleration
```

---

## JOGVH Jog Velocity High

Type	Jog	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>JOGVH<r>	GT6K	6.0
Units	r = units/sec (scalable with SCLV)	GV6K	6.0
Range	Stepper Axes: 0.0000-60.0000 (after conversion to revs/sec) Servo Axes: 0.0000-200.0000 (after conversion to revs/sec; if ERES > 10,000, maximum = 2,000,000/ERES)		
Default	10.0000		
Response	JOGVH: *JOGVH10.0000		
See Also	DJOG, JOG, JOGA, JOGAA, JOGAD, JOGADA, JOGVL, INFNC , SCALE, SCLV		

---

The 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.

**UNITS OF MEASURE and SCALING:** refer to page 16.

**Example:** Refer to the jog mode enable (JOG) command example.

---

## JOGVL Jog Velocity Low

Type	Jog	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>JOGVL<r>	GT6K	6.0
Units	r = units/sec (scalable with SCLV)	GV6K	6.0
Range	Stepper Axes: 0.0000-60.0000 (after conversion to revs/sec) Servo Axes: 0.0000-200.0000 (after conversion to revs/sec; if ERES > 10,000, maximum = 2,000,000/ERES)		
Default	0.5000		
Response	JOGVL: *JOGVL0.50000		
See Also	DJOG, JOG, JOGA, JOGAA, JOGAD, JOGADA, JOGVH, INFNC , SCALE, SCLV		

---

The 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.

**UNITS OF MEASURE and SCALING:** refer to page 16.

**Example:** Refer to the jog mode enable (JOG) command example.

---

## JOY

### Joystick Mode Enable

Type	Joystick	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>JOY<b>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	b = 0 (disable) or 1 (enable)		
Default	0		
Response	JOY: *JOY0		
See Also	ANIRNG, [ AS ], COMEXC, INFNC, JOYA, JOYAA, JOYAD, JOYADA, JOYAXH, JOYCDB, JOYCTR, JOYEDB, JOYVH, JOYVL, JOYZ, TAS, TIN		

---

The Gem6K drive supports joystick operation with digital inputs and analog inputs.

**To Set Up Joystick Operation** (refer also to the example code below):

1. Select the required digital inputs and analog inputs required for joystick operation. Connect the joystick as instructed in the *Hardware Installation Guide*.
2. Assign the appropriate input functions to the digital inputs used for the joystick's operation:
  - Release Input: INFNCi-M
  - Velocity Select Input: INFNCi-O
3. (optional) Use the ANIRNG command to select the voltage range for the analog input you will use. The default range is -10VDC to +10VDC (other options are 0 to +5V, -5 to +5V, and 0 to +10V).
4. Assign analog input, using:
  - JOYAXH: Standard analog input-to-axis assignment.
5. Define the joystick motion parameters:
  - Max. Velocity when Velocity Select input switch is open/high (JOYVH command). If the Velocity Select input is not used, joystick motion always uses the JOYVH velocity.
  - Max. Velocity when Velocity Select input switch is closed/low (JOYVL command).
  - Accel (JOYA command).
  - Accel for s-curve profiling (JOYAA command).
  - Decel (JOYAD command).
  - Decel for s-curve profiling (JOYADA command).
6. Define the usable voltage zone for your joystick:  
(make sure you have first assigned the analog inputs – see step 4 above)
  - End Deadband (JOYEDB): Defines the voltage offset (from the -10V & +10V endpoints) at which max. velocity occurs. Default is 0.1V, maxing voltage at -9.9V and +9.9V.
  - Center Voltage (JOYCTR or JOYZ): Defines the voltage when the joystick is at rest to be the zero-velocity center. Default JOYCTR setting is 0V.
  - Center Deadband (JOYCDB): Defines the zero-velocity range on either side of the Center Voltage. Default is 0.1V, setting the zero-velocity range at -0.1V to +0.1V.
7. To jog the axis:
  - a. In your program, enable Joystick Operation with the JOY command (Joystick Release input must be closed in order to enable joystick mode). When the JOY command enables joystick mode, program execution stops (assuming the Continuous Command Execution Mode is disabled with the COMEXCØ command).
  - b. Move the load with the joystick.
  - c. When you are finished, open the Joystick Release input to disable joystick mode. This allows program execution to resume with the next statement after the initial JOY command that started the joystick mode.

## Programming Example (refer also to the illustration below):

### Application Requirements:

This example represents a typical joystick application in which a high-velocity range is required to move to a region, then a low-velocity range is required for a fine search. After the search is completed it is necessary to record the load position, then move to the next region. A digital input can be used to indicate that the position should be read. The Joystick Release input is used to exit the joystick mode and continue with the motion program.

### Hardware Configuration:

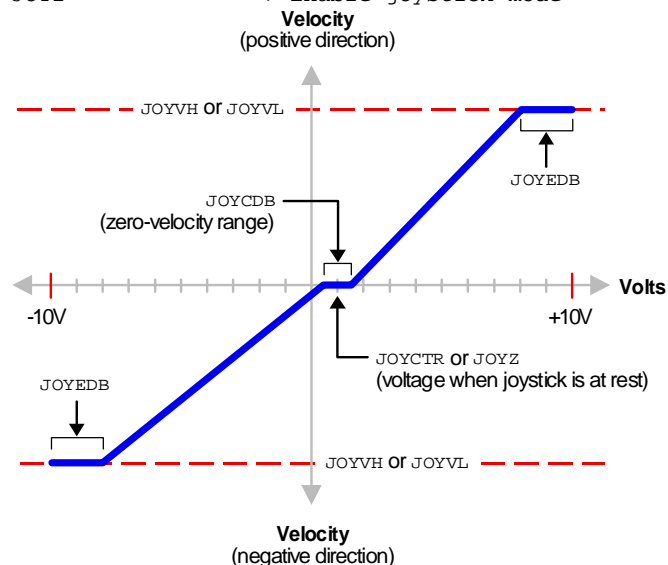
- An analog input SIM is installed in the 3rd slot of I/O brick 1. The eight analog inputs (1-8) are addressed as input numbers 17-24 on the I/O brick. Analog input 17 will control the axis.
- A digital input SIM is installed in the 1st slot of I/O brick 1. The eight digital inputs (1-8) are addressed as input numbers 1-8 on the I/O brick. Digital input 6 will be used for the Joystick Release function, and input 7 will be used for the Joystick Velocity Select input. Input 8 will be used to indicate that the position should be read.

### Setup Code (the drawing below shows the usable voltage configuration):

```

1INFNC6-M           ; Assign Joystick Release function to brick 1, input 6
1INFNC7-O           ; Assign Joystick Velocity Select function to brick 1, input 7
JOYAXH1-17          ; Assign analog input 17 to control the axis
JOYVH10             ; Max. velocity is 10 units/sec when the Velocity Select
                   ; input switch (1IN.7) is open (high)
JOYVL1             ; Max. velocity is 1 unit/sec when the Velocity Select
                   ; input switch (1IN.7) is closed (low)
JOYA100            ; Set joystick accel to 100 units/sec/sec
JOYAD100           ; Set joystick decel to 100 units/sec/sec
;**** COMMANDS TO SET UP USABLE VOLTAGE: ****
1JOYCTR.17=+1.0    ; Set center voltage for analog input 17 to +1.0V. The +1.0V value
                   ; was ascertained by checking the voltage of the input (with
                   ; the 1TANI.17 command) when the joystick was at rest
1JOYCDB.17=0.5     ; Set center deadband to compensate for the fact that
                   ; when the joystick is at rest, the voltage received on
                   ; the analog input may fluctuate +/- 0.5V on either
                   ; side of the +1.0V center.
1JOYEDB.17=2.0     ; Set end deadband to compensate for the fact that the
                   ; joystick can produce only -8.0V to +8.0V.
;*****
JOY1               ; Enable joystick mode

```



---

## JOYA

### Joystick Acceleration

		Product	Rev
Type	Joystick	GT6K	6.0
Syntax	<a_><!>JOYA<r>	GV6K	6.0
Units	r = units/sec/sec		
Range	0.3 - 9999.9999		
Default	10.0000		
Response	JOYA: *JOYA10.0000		
See Also	JOY, JOYAA, JOYAD, JOYADA, JOYAXH, JOYCDB, JOYCTR, JOYEDB, JOYVH, JOYVL, JOYZ, SCALE, SCLA		

---

The JOYA command specifies the acceleration to be used during joystick mode.

**UNITS OF MEASURE and SCALING:** refer to page 16.

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)	GT6K	6.0
Syntax	<a_><!>JOYAA<r>	GV6K	6.0
Units	r = units/sec/sec		
Range	0, or 0.3 - 9999.9999		
Default	10.00 (Default is a constant accel ramp, where JOYAA tracks JOYA. S-curve accel is achieved when JOYAA is changed independent of JOYA. To restore a constant accel ramp and tracking, set JOYAA = 0.)		
Response	JOYAA: *JOYAA10.0000		
See Also	AA, AD, JOY, JOYA, JOYAD, JOYADA, SCALE, SCLA		

---

The 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*. Refer to page 15 for details on S-curve profiling.

Accelerating Scaling (SCLA) affects JOYAA the same as it does for JOYA. Refer to page 16 for details on scaling.

**Example:**

```
JOYA10           ; Set the maximum joystick acceleration
JOYAA5           ; Set the average joystick acceleration
```

---

## JOYAD Joystick Deceleration

Type	Joystick	Product	Rev
Syntax	<a_><!>JOYAD<r>	GT6K	6.0
Units	r = units/sec/sec	GV6K	6.0
Range	0, or 0.3 - 9999.9999		
Default	10.0000 (by default, JOYAD tracks JOYA until JOYAD is changed independent of JOYA; to restore tracking, set JOYAD = 0)		
Response	JOYAD: *JOYAD10.0000		
See Also	JOY, JOYA, JOYAA, JOYADA, JOYAXH, JOYCDB, JOYCTR, JOYEDB, JOYVH, JOYVL, JOYZ, SCALE, SCLA		

---

The JOYAD command specifies the deceleration to be used during the joystick mode.

**UNITS OF MEASURE and SCALING:** refer to page 16.

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

Type	Motion (S-Curve)	Product	Rev
Syntax	<a_><!>JOYADA<r>	GT6K	6.0
Units	r = units/sec/sec	GV6K	6.0
Range	0.3 - 9999.9999		
Default	10.00 (By default, JOYADA tracks JOYAA until JOYADA is changed independent of JOYAA. To restore tracking, set JOYAD = 0.)		
Response	JOYADA: *JOYADA10.0000		
See Also	A, AD, JOY, JOYA, JOYAA, JOYAD, SCALE, SCLA		

---

The 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*. Refer to page 15 for details on S-curve profiling.

Acceleration Scaling (SCLA) affects JOYADA the same as it does for JOYAD. Refer to page 16 for details on scaling.

**Example:**

```
JOYAD10          ; Sets the maximum joystick deceleration
JOYADA5         ; Sets the average joystick deceleration
```

---

## JOYAXH Joystick Analog Channel High

Type	Joystick	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>JOYAXH<B-i>	GT6K	6.0
Units	B = I/O brick number i = Location of the analog input on I/O brick (see page 8)	GV6K	6.0
Range	B = 1-8 i = 1-32		
Default	0-0		
Response	JOYAXH: *JOYAXH1-1		
See Also	ANIRNG, [ IN ], INFNC, JOY, JOYA, JOYAA, JOYAD, JOYADA, JOYCDB, JOYCTR, JOYEDB, JOYVH, JOYVL, JOYZ, TIN, TLIM		

---

The JOYAXH command specifies the analog input that will control the axis while Joystick Mode is enabled.

To understand how specific I/O points are addressed on the I/O bricks, refer to page 8.

**NOTE:** If you install an analog input SIM on an external I/O brick, use the ANIRNG command to select the voltage range for the external 12-bit analog inputs you will use. The default range is -10VDC to +10VDC (other options are 0 to +5V, -5 to +5V, and 0 to +10V). The onboard analog input on the DRIVE I/O connector is not affected by the ANIRNG command.

**Example:** Refer to the joystick mode enable (JOY) command example.

---

## JOYCDB Joystick Center Deadband

Type	Joystick	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!><B>JOYCDB<.i><=r>	GT6K	6.0
Units	i = I/O location for the analog input on brick B (see page 8) r = volts	GV6K	6.0
Range	i = 1-32 r = 0.00 - 10.00 (depending on ANIRNG setting)		
Default	0.1		
Response	1JOYCDB.1 *1JOYCDB.1=0.1		
See Also	ANIRNG, INFNC, JOY, JOYA, JOYAA, JOYAD, JOYADA, JOYAXH, JOYCTR, JOYEDB, JOYVH, JOYVL, JOYZ		

---

The JOYCDB command defines, for the specified analog input, the zero-velocity range on either side of the Center Voltage established with the JOYCTR command or the JOYZ command. The default setting is 0.1V, which sets the zero-velocity range at -0.1V to +0.1V (assuming the default JOYCTR default of 0.0V is used). **NOTE:** Executing the JOYCDB command before the JOYAXH command will cause an error (“INPUT(S) NOT DEFINED AS JOYSTICK INPUT”).

**Example:** Refer to the joystick mode enable (JOY) command example.

---

## JOYCTR Joystick Center

Type	Joystick	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!><B>JOYCTR<.i><=r>	GT6K	6.0
Units	i = I/O location for the analog input on brick B (see page 8) r = volts	GV6K	6.0
Range	i = 1-32 r = -10.00 - +10.00 (depending on ANIRNG setting)		
Default	0.00		
Response	1JOYCTR.1 *1JOYCTR.1=0.00		
See Also	ANIRNG, INFNC, JOY, JOYA, JOYAA, JOYAD, JOYADA, JOYAXH, JOYCDB, JOYEDB, JOYVH, JOYVL, JOYZ		

---

The JOYCTR command defines, for the specified analog input(s), the voltage to be considered as the zero-velocity center (usually associated with leaving the joystick in the resting position). Default is 0V. The zero-velocity range about the center is determined by the JOYCDB command. As an alternative to the JOYCTR command, you could use the JOYZ command. **NOTE:** Executing the JOYCTR command before the JOYAXH command will cause an error (“INPUT(S) NOT DEFINED AS JOYSTICK INPUT”).

**Example:** Refer to the joystick mode enable (JOY) command example.

---

---

## JOYEDB Joystick End Deadband

Type	Joystick	Product	Rev
Syntax	<a_><!><B>JOYEDB<.i><=r>	GT6K	6.0
Units	i = I/O location for the analog input on brick B (see page 8) r = volts	GV6K	6.0
Range	i = 1-32 r = 0.00 - 10.00 (depending on ANIRNG setting)		
Default	0.1		
Response	1JOYEDB.1 *1JOYCTR.1=0.1		
See Also	ANIRNG, INFNC, JOY, JOYA, JOYAA, JOYAD, JOYADA, JOYAXH, JOYCDB, JOYCTR, JOYVH, JOYVL, JOYZ		

---

The JOYEDB command defines, for the specified analog input, the voltage offset (from the -10V & +10V endpoints) at which maximum velocity occurs. This command is useful if your joystick does not reach either limit of the voltage range (-10.00V to +10.00V). The default setting is 0.1V, creating a maximum voltage range of -9.9V to +9.9V. **NOTE:** Executing the JOYEDB command before the JOYAXH command will cause an error (“INPUT(S) NOT DEFINED AS JOYSTICK INPUT”).

**Example:** Refer to the joystick mode enable (JOY) command example.

---

## JOYVH Joystick Velocity — Velocity Select Input High

Type	Joystick	Product	Rev
Syntax	<a_><!>JOYVH<r>	GT6K	6.0
Units	r = units/sec	GV6K	6.0
Range	Stepper Axes: 0.004-60.0000 (after conversion to revs/sec) Servo Axes: 0.004-200.0000 (after conversion to revs/sec; if ERES > 10,000, maximum = 2,000,000/ERES)		
Default	0.5000		
Response	JOYVH: *JOYVH0.5000		
See Also	[ IN ], INFNC, JOY, JOYA, JOYAA, JOYAD, JOYADA, JOYCDB, JOYCTR, JOYEDB, JOYVL, JOYZ, SCALE, SCLV, TIN, TLIM		

---

The JOYVH command specifies the maximum velocity that can be obtained at full deflection during joystick mode, with the Joystick Velocity Select input open (high). The Joystick Velocity Select input function is defined with the INFNCi-0 command. If the Velocity Select input is not used, joystick motion always uses the JOYVH velocity.

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.

<b>UNITS OF MEASURE and SCALING:</b> refer to page 16.
--

**Example:** Refer to the joystick mode enable (JOY) command example.



---

## JOYVL Joystick Velocity — Velocity Select Input Low

Type	Joystick	Product	Rev
Syntax	<a_><!>JOYVL<r>	GT6K	6.0
Units	r = units/sec	GV6K	6.0
Range	Stepper Axes: 0.004-60.0000 (after conversion to revs/sec) Servo Axes: 0.004-200.0000 (after conversion to revs/sec; if ERES > 10,000, maximum = 2,000,000/ERES)		
Default	0.2000		
Response	JOYVL: *JOYVL0.2000		
See Also	[ IN ], INFNC, JOY, JOYA, JOYAA, JOYAD, JOYADA, JOYAXH, JOYCDB, JOYCTR, JOYEDB, JOYVH, JOYZ, SCALE, SCLV, TIN, TLIM		

---

The JOYVL command specifies the maximum velocity that can be obtained at full deflection during joystick mode, with the Joystick Velocity Select input closed (low). The Joystick Velocity Select input function is defined with the INFNCi-O command. If the Velocity Select input is not used, joystick motion always uses the JOYVH velocity.

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.

**UNITS OF MEASURE and SCALING: refer to page 16.**

**Example:** Refer to the joystick mode enable (JOY) command example.

---

## JOYZ Joystick Zero

Type	Joystick	Product	Rev
Syntax	<a_><!><B>JOYZ<.i><=b> (multiple inputs per brick may be configured at one time)	GT6K	6.0
Units	B = I/O brick number i = Location of the analog input on I/O brick (see page 8) b = enable bit	GV6K	6.0
Range	B = 1-8 i = 1-32 b = 0 (don't zero) or 1 (zero), or X (don't change)		
Default	n/a		
Response	n/a		
See Also	ANIRNG, JOY, JOYA, JOYAA, JOYAD, JOYADA, JOYAXH, JOYCDB, JOYCTR, JOYEDB, JOYVH, JOYVL		

---

The JOYZ command defines the voltage when the joystick is at rest to be the zero-velocity center. Simply leave the joystick in its resting position and issue a JOYZ command to define the current voltage of the respective analog inputs as the zero-velocity center. The zero-velocity range about the center is determined by the JOYCDB command.

The JOYZ command is an alternative to using the JOYCTR command.

**Example:**

```
1INFNC7-M      ; Assign Joystick Release function to brick 1, input 7
1INFNC8-O      ; Assign Joystick Velocity Select function to brick 1, input 8
JOYAXH1-17     ; Assign analog input 17 to control the axis
JOYVH10        ; Max. velocity is 10 units/sec when the Velocity Select
                ; input (1IN.8) is high (sinking current)
JOYVL1         ; Max. velocity is 1 unit/sec when the Velocity Select
                ; input (1IN.8) is low (not sinking current)
JOYA100        ; Set joystick accel to 100 units/sec/sec
JOYAD100       ; Set joystick decel to 100 units/sec/sec
;**** COMMANDS TO SET UP USABLE VOLTAGE: ****
1JOYZ.17=1     ; This command is executed while the joystick is at rest. It
                ; sets the current voltage on analog input 17 as the
                ; zero-velocity center.
```

```

1JOYCDB.17=0.5      ; Set center deadband to compensate for the fact that
                    ; when the joystick is at rest, the voltage received on
                    ; the analog input may fluctuate +/- 0.5V on either
                    ; side of the zero-velocity center established with JOYZ.
1JOYEDB.17=2.0      ; Set end deadband to compensate for the fact that the
                    ; joystick can produce only -8.0V to +8.0V.
;*****
JOY1                 ; Enable joystick mode

```

---

## JUMP Jump to a Program or Label (and do not return)

Type	Program or Subroutine Definition or Program Flow Control	Product	Rev
Syntax	<a_><!>JUMP<t>	GT6K	6.0
Units	t = text (name of program/label)	GV6K	6.0
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

```

; *****
; * In this example, the program bottom is executed and calls the top *
; * subroutine. The top subroutine then initiates motion (GO1) and *
; * jumps to the program called middle to turn on output #2. Then, *
; * because the JUMP command cleared the top subroutine, program *
; * execution is terminated instead of returning to the bottom program. *
; *****
DEL top           ; Delete program top
DEF top           ; Begin definition program top
GO1               ; Initiate motion
JUMP middle       ; Jump to program middle
END               ; End subroutine definition
DEL middle        ; Delete program middle
DEF middle        ; Begin definition of program middle
OUTx1             ; Turn on output #2
END               ; End program definition
DEL bottom        ; Delete program bottom
DEF bottom        ; Begin definition of program bottom
GOSUB top         ; Gosub to program top
GO1               ; Initiate motion
END               ; End subroutine definition
RUN bottom        ; Execute program bottom

```

<b>K Kill Motion</b>		<b>Product</b>	<b>Rev</b>
Type	Motion	GT6K	6.0
Syntax	<a_><!>K<b>	GV6K	6.0
Units	n/a		
Range	b = 0 (don't kill) or 1 (kill)		
Default	n/a		
Response	!K No response, instead motion is killed		
See Also	FOLK, GO, <CTRL>K, KDRIVE, LHAD, LHADA, S, SCANP, TAS		

The Kill Motion (K) command instructs the drive to stop motion. Two types of Kill are available:

- **Kill motion only:** K1 (buffered) or !K1 (immediate).
- **Kill motion and terminate program:** !K (immediate), K (buffered), <CTRL>K  
Additional methods include: activate a kill input (INFNCi-C), activate a user fault input (INFNCi-F), or open the Enable input connection.

When a kill is initiated, motion is stopped at the rate set with the LHADA and LHAD commands.

If you want the drive to be disabled upon executing a Kill command, enable the *Disable Drive on Kill* mode with the KDRIVE1 command. **CAUTION:** In the KDRIVE1 mode, a Kill command or Kill input immediately shuts down the drive, allowing the load to *free wheel* to a stop.

If the axis is involved in a PLC Scan (initiated with SCANP), a K command will clear the scan.

**Example:**

```
A2           ; Set acceleration to 2 units/sec/sec
AD2          ; Set deceleration to 2 units/sec/sec
V1           ; Set velocity to 1 units/
D10          ; Set distance to 10 units
GO1          ; Initiate motion -- motion begins.
             ; After a short period the Kill command is sent.
!K           ; Kill motion (stop at the LHADA/LHAD decel)
```

<b>&lt;CTRL&gt;K Kill Motion</b>		<b>Product</b>	<b>Rev</b>
Type	Motion	GT6K	6.0
Syntax	<CTRL>K	GV6K	6.0
Units	n/a		
Range	n/a		
Default	n/a		
Response	<CTRL>K: No response, instead motion is killed		
See Also	GO, K, KDRIVE, LHAD, LHADA, S		

The Kill Motion (<ctrl>K) command instructs the Gem6K to stop motion and terminate program execution. In essence, the <ctrl>K command is an immediate kill (!K) command.

Motion is stopped at the rate set with the LHADA and LHAD commands. If the *Disable Drive on Kill* mode is enabled with the KDRIVE1 command, a <ctrl>K command immediately shuts down the drive, allowing the load to *free wheel* to a stop.

**Example:**

```
A2           ; Set acceleration to 2 units/sec/sec
AD2          ; Set deceleration to 2 units/sec/sec
V1           ; Set velocity to 1 units/
D10          ; Set distance to 10 units
GO1          ; Initiate motion -- motion begins.
             ; After a short period the Kill command is sent.
<CTRL>K     ; Kill motion (stop at the LHADA/LHAD deceleration)
```

---

## KDRIVE      Disable Drive on Kill

Type	Drive Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>KDRIVE<b>	GT6K	6.0
Units	b = enable bit	GV6K	6.0
Range	0 (disable) or 1 (enable)		
Default	0		
Response	KDRIVE: *KDRIVE0		
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.

**CAUTION:** This cuts all control to the motor and allows the load to freewheel to a stop.

To re-enable the drive, issue the DRIVE1 command.

If you leave the KDRIVE command in its default state (∅, disabled), the kill function behaves in its normal manner, leaving the drive enabled.

### Example:

```
KDRIVE1                   ; Disable the drive when a kill occurs  
K                         ; Kill is performed and drive is disabled
```

---

## KIOEN      Kill on EVM32 Disconnect

Type	Drive Configuration; Inputs; Outputs	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>KIOEN<b>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	b = 0 (disable) or 1 (enable)		
Default	0		
Response	KIOEN: *KIOEN0		
See Also	ERROR, TER, TIO		

---

The KIOEN command allows you to control the functionality of the Gem6K when an *EVM32 I/O failure* has been detected (cause could be cable disconnection, or loss of power on the EVM32 I/O brick). The options are as follows:

- KIOEN0 (factory default): The Gem6K will not perform a kill. **(SEE WARNING NOTE BELOW)**
- KIOEN1: The Gem6K will perform a kill (all motion and program execution on all tasks) if an EVM32 I/O failure is detected.

Regardless of the state of the KIOEN command, you can enable error bit 18 (ERROR.18-1) and use an error program to respond to an EVM32 I/O failure. Note that when operating in the KIOEN0 mode (factory default), the branch to the error program is a GOSUB, but in KIOEN1 mode the branch is a GOTO.

---

### WARNING

---

If you use the factory default KIOEN0 mode (no kill if EVM32 I/O failure), bear in mind that when an EVM32 I/O failure occurs, the state of any external conditions becomes unknown to the Gem6K drive. For example, if an input on brick 2 is defined as a hardware end-of-travel limit, and if brick 2 loses power or is disconnected from the Gem6K, the Gem6K will never see a state change on the limit.

The KIOEN0 mode is designed for use during system setup (for example, if you will be connecting and disconnecting I/O bricks in the process of wiring and programming your system).

Therefore, to help prevent damage to equipment and serious injury to personnel, we recommend enabling this feature (KIOEN1) during normal operation of your motion control system.

---

---

<b>L</b>	<b>Loop</b>	<b>Product</b>	<b>Rev</b>
Type	Loops; Program Flow Control	GT6K	6.0
Syntax	<a_><!>L<i>	GV6K	6.0
Units	i = number of times to loop		
Range	0-999,999,999 (0 = infinite loop)		
Default	0		
Response	L: No response; instead, this has the same function as L0		
See Also	C, COMEXS, LN, LX, PLN, PLOOP, PS, S		

---

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 specified by L<i>. If <i> =  $\emptyset$ , 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, but only in the COMEXS1 mode. 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:**

```

DEL flash      ; Delete program flash
DEF flash      ; Begin definition of program flash
L5             ; Repeat the commands between L and LN five times
A20           ; Set acceleration to 20 revs/sec
D8300         ; Set distance to 8300 counts
WAIT(AS.1=b0) ; Wait until no motion is commanded
OUTxxx1       ; Turn on output #4
T.75         ; Dwell for 0.75 seconds
OUTxxx0       ; Turn off output #4
T.75         ; Dwell for 0.75 seconds
LN            ; End loop
END           ; End definition

```

---

<b>LDAMP</b>	<b>Load Damping</b>	<b>Product</b>	<b>Rev</b>
Type	System	GT6K	n/a
Syntax	<a_><!>LDAMP<r>	GV6K	6.0
Units	Rotary motor: r = Nm/rad/sec Linear motor: r = N/meter/sec		
Range	Rotary motor: 0.0000 to 1.0000 : $\pm 0.0001$ Linear motor: DMEPIT (electrical pitch) dependent		(applicable only to servo axes)
Default	0.0000		
Response	LDAMP: *LDAMP0.2000		
See Also	DMTD, SGPRAT, SGVRAT, TGAIN, TSGSET		

---

The LDAMP command specifies the damping provided by the mechanical load only, not including the motor itself (which is specified by DMTD).

**Working with servo gains.**

- Servo tuning process: refer to the *Hardware Installation Guide*.
- Check the values of all active gains (LDAMP is one of many servo gains): use TGAIN.
- Creating and invoking gain sets: see SGSET, SGENB, TGAIN, TSGSET.

## LH Hardware End-of-Travel Limit — Enable Checking

Type	Limit (End-of-Travel); Drive Configuration	Product	Rev
Syntax	<a_><!>LH<i>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	i = 0 (disable both), 1 (disable positive-direction), 2 (disable negative-direction), or 3 (enable both)		
Default	3		
Response	LH: *LH3		
See Also	[ AS ], [ ER ], DRIVE, ERROR, INDEB, INFNC, INLVL, LHAD, LHADA, [ LIM ], LIMEN, LIMFNC, LIMLVL, LS, LSAD, LSADA, LSNEG, LSPOS, TAS, TER, TLIM, TSTAT		

Use the LH command to enable or disable either or both end-of-travel limit inputs.

Disable negative-direction limit; Disable positive-direction limit: LH0  
Enable negative-direction limit; Disable positive-direction limit: LH1  
Disable negative-direction limit; Enable positive-direction limit: LH2  
Enable negative-direction limit; Enable positive-direction limit: LH3

The “end-of-travel limit” functions are assigned with the LIMFNC command to the onboard limit inputs on the DRIVE I/O connector; or with the INFNC command to onboard or external inputs. The factory default LIMFNC assignments on the DRIVE I/O connector are as follows (but any of the three limit inputs, five general purpose inputs or inputs on external I/O bricks could be re-assigned as end-of-travel inputs):

Limit #	Pin #	GT6K & GV6K Function (LIMFNC default)
1	28	LIMFNC1-R (Positive end-of-travel limit)
2	29	LIMFNC2-S (Negative end-of-travel limit)
3	31	LIMFNC3-T (Home limit)

The LH command is required to enable checking the state of the end-of-travel limits (i.e., LIMFNCi-R, or LIMFNCi-S); for example, LH3 is required to detect the occurrence of the hardware limit activation, as reported with axis status bits 15 and 16 (see TASF, TAS, AS). The default LH condition is enabled (LH3).

With limits disabled, motion will not be restricted. When a specific limit is enabled (positive- or negative-direction), and the limit wiring for the enabled limit is a physical open circuit, motion will be restricted (assuming the default active level of LIMLVLØ). The LIMLVL command controls the active level of the limit inputs.

If an “end-of-travel limit” input is redefined with a different function (i.e., not LIMFNCi-R or LIMFNCi-S), it is no longer controlled by the LH command. Instead, use the LIMEN command.

### NOTE

If a hardware limit is encountered while limits are enabled, motion must occur in the opposite direction; then you can make a move in the original direction. If limits are disabled (LH0), you are free to make a move in either direction. When a hardware limit is encountered, the drive sets TAS bit 15 or 16, as well as TER bit 2. To clear these status bits, you can execute a GO in the opposite direction or disable the limits with the LH0 command.

### Example:

```
LH3           ; Enable limits
LHAD100      ; Set hard limit decel to 100 units/sec/sec
A10          ; Set acceleration to 10 units/sec/sec
V1           ; Set velocity to 1 unit/sec
D100000      ; Set distance to 100000 units
GO1          ; Initiate motion
```

---

## LHAD

### Hard Limit Deceleration

Type	Limit (End-of-Travel)	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>LHAD<r>	GT6K	6.0
Units	r = units/sec/sec (linear motors: see DMEPIT for linear/rotary conversion)	GV6K	6.0
Range	0.0001 - 9999.9999		
Default	100.0000		
Response	LHAD: *LHAD100.0000		
See Also	AD, DMEPIT, DRES, INFNC, INLVL, K, [ LIM ], LIMEN, LIMFNC, LIMLVL, LH, LHADA, LS, LSAD, LSADA, LSNEG, LSPOS, SCALE, SCLA, TLIM		

---

The LHAD command determines the value at which to decelerate after a hardware end-of-travel limit has been hit.

The “end-of-travel limit” functions are assigned with the LIMFNC command to the onboard limit inputs on the DRIVE I/O connector; or with the INFNC command to onboard or external inputs. The factory default LIMFNC assignments on the DRIVE I/O connector are as follows (but any of the three limit inputs, five general purpose inputs or inputs on external I/O bricks could be re-assigned as end-of-travel inputs):

Limit #	Pin #	GT6K & GV6K Function (LIMFNC default)
1	28	LIMFNC1-R (Positive end-of-travel limit)
2	29	LIMFNC2-S (Negative end-of-travel limit)
3	31	LIMFNC3-T (Home limit)

**UNITS OF MEASURE and SCALING:** refer to page 16.

When a drive fault, a Kill command (K, !K, or ^K), or a Kill input (INFNCi-C) occurs, motion is stopped at the rate set with the LHAD and LHADA commands. However, if the *Disable Drive on Kill* mode is enabled (KDRIVE1), the drive is immediately shut down upon a Kill command or Kill 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

Type	Motion (S-Curve)	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>LHADA<r>	GT6K	6.0
Units	r = units/sec/sec (linear motors: see DMEPIT for linear/rotary conversion)	GV6K	6.0
Range	0.0001 - 9999.9999		
Default	100.000 (Default is a constant decel ramp, where LHADA tracks LHAD. S-curve decel is achieved when LHADA is changed independent of LHAD. To restore a constant decel ramp and tracking, set AA = 0.)		
Response	LHADA: *LHADA100.0000		
See Also	AD, ADA, INFNC, K, LHAD, LIMFNC, LIMLVL, SCALE, SCLA		

---

The LHADA command allows you to specify the average deceleration for an S-curve deceleration profile when a hardware end-of-travel limit is hit.

The “end-of-travel limit” functions are assigned with the LIMFNC command to the onboard limit inputs on the DRIVE I/O connector; or with the INFNC command to onboard or external inputs. The factory default LIMFNC assignments on the DRIVE I/O connector are as follows (but any of the three limit inputs, five general purpose inputs or inputs on external I/O bricks could be re-assigned as end-of-travel inputs):

Limit #	Pin #	GT6K & GV6K Function (LIMFNC default)
1	28	LIMFNC1-R (Positive end-of-travel limit)
2	29	LIMFNC2-S (Negative end-of-travel limit)
3	31	LIMFNC3-T (Home limit)

Acceleration scaling (SCLA) affects LHADA the same as it does for LHAD. Refer to page 16 for details on scaling. S-curve profiling provides smoother motion control by reducing the rate of change in deceleration; this decel rate of change is known as *jerk*. Refer to page 15 for details on S-curve profiling.

**Example:**

```
LHAD10 ; Set the maximum deceleration to 10 units/sec/sec
LHADA5 ; Set the average deceleration to 5 units/sec/sec
```

**[ LIM ]**

**Limit Status**

Type	Assignment or Comparison	Product	Rev
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	[ IN ], INDEB, INFNC, LH, LIMEN, LIMFNC, LIMLVL, TLIM		

The LIM operator 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, 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.

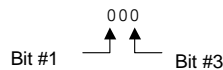
LIM does not depict the status of onboard inputs or external digital inputs assigned an end-of-travel or home limit function (INFNCi-R, INFNCi-S, or INFNCi-T). For such inputs, you must use the IN operator.

**Syntax:** VARBn=LIM where n is the binary variable number,  
or LIM can be used in an expression such as IF(LIM=b1XX), or IF(LIM=h7)

The LIM value is the debounced version of the limits status (debounced with the 0INDEB value). Axis status (AS) bits 15 and 16 reports the non-debounced version of the end-of-travel limits.

There are 3 limit inputs: positive-direction, negative-direction, and home 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.3 assigns limit input 3 (home limit) to binary variable 1.

Format for binary assignment:



LIM bit	Function
1	Positive-direction Limit
2	Negative-direction Limit
3	Home Limit

**Example:**

```
IF(LIM=b11X) ; If both end of travel limit inputs are active, then
              ; do the statements between the IF and NIF
TLIM         ; Transfer limit status
NIF          ; End IF statement
```



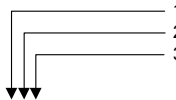
## LIMEN Limit Input Enable

Type	Inputs; Program Debug Tool	Product	Rev
Syntax	<a_><!>LIMEN<d><d><d>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	d = 0 (disable, leave off), 1 (disable, leave on), E (enable), or X (don't change)		
Default	E		
Response	LIMEN: *LIMENEEE LIMEN.3 *E		
See Also	INEN, LH, [ LIM ], LIMFNC, LIMLVL, TIO, TLIM		

The LIMEN command allows you to simulate the activation of specific limit inputs (without actually wiring the inputs to the drive) by disabling them and setting them to a specific level (ON or OFF). This is useful for testing and debugging your program (see program example below). LIMEN may only be used for onboard limit inputs (found on the "DRIVE I/O" connector), not for onboard or external digital inputs. The default state is enabled (E), requiring external wiring to exercise their respective LIMFNC functions.

LIMEN response for limits on the DRIVE I/O connector (bits are numbered 1-3 from left to right):

Limit #	Pin #	GT6K & GV6K Function (LIMFNC default)
1	28	LIMFNC1-R (Positive end-of-travel limit)
2	29	LIMFNC2-S (Negative end-of-travel limit)
3	31	LIMFNC3-T (Home limit)



LIMENbbb

The LH command is required to enable checking the state of the end-of-travel limits (i.e., LIMFNCi-R, or LIMFNCi-S); for example, LH3 is required to detect the occurrence of the hardware limit activation, as reported with axis status bits 15 and 16 (see TASF, TAS, AS). The default LH condition is enabled (LH3).

The input bit patterns for onboard and external I/O bricks are explained on page 8 of this document.

### Example:

```
DEL tsting          ; Delete program called "tsting"
DEF tsting          ; Begin definition of program called "tsting"
LIMFNC3-E          ; Define hardware end-of-travel limit #3 (normally defined
                  ; as the home end-of-travel limit) as a "pause/resume" input.
COMEXR1           ; Activating the pause/resume input will pause command and
                  ; motion execution, de-activating the pause/resume input
                  ; will resume command and motion execution.
MC1               ; Set the axis to continuous motion profiling mode
A15               ; Set acceleration to 15 units/sec/sec
AD5               ; Set deceleration to 5 units/sec/sec
V4                ; Set velocity to 4 revs/sec
GO1               ; Initiate continuous motion
END               ; End definition of program called "tsting"
; While this program is running and motion is in progress, you can send
; immediate LIMEN commands to simulate the function of the "pause/resume"
; input as follows:
; 1. Start the program by sending the RUN TSTING command to the drive.
;    The axis will start moving, using a continuous motion profile.
; 2. Send the !LIMEN.3=1 command to the drive. This disables the
;    "pause/resume" input but simulates its activation. Motion and
;    program execution will pause.
; 3. Send the !LIMEN.3=0 command to the drive. This disables the
;    "pause/resume" input but simulates its de-activation. Motion and
;    program execution will resume.
; 4. Send the !LIMEN.3=E command to the drive. This re-enables the
;    "pause/resume" input for normal operation with an external switch
;    or sensor.
; 5. To stop this experiment, send the !K command to the drive.
;    This "kills" program execution and motion.
```

## LIMFNC Input Function for Limit Inputs

Type	Inputs; Limits (end of travel); Homing	Product	Rev
Syntax	<a_><!>LIMFNC<i>-<a>c</i>	GT6K	6.0
Units	i = input # on the "DRIVE I/O" connector (see page 8); a = program # for function P; c = function identifier letter	GV6K	6.0
Range	i = 1-3; c = A-T		
Default	(pre-assigned limits and home function -- see below)		
Response	LIMFNC: *LIMFNC1-A NO FUNCTION - STATUS OFF (repeated for all onboard limit inputs)		
See Also	COMEXR, COMEXS, [ ER ], ERROR, INDEB, INFNC, INPLC, INSELP, INSTW, INTHW, JOY, JOYAXH, JOYVH, JOYVL, K, KDRIVE, LH, [ LIM ], LIMEN, LIMFNC, LIMLVL, PSET, [ SS ], TER, TIN, TRGFN, TRGLOT, [ TRIG ], TSS, TSTAT, TTRIG		

The LIMFNC command defines the function of each individual limit input found on the "DRIVE I/O" connector. The factory default configuration is that each dedicated hardware end-of-travel and home limit is assigned to its respective LIMFNC function. That is, positive limit is assigned to LIMFNC1-R, negative limit is assigned to LIMFNC2-S, home limit is assigned to LIMFNC3-T.

**Input debounce.** By default, the limit inputs are assigned their respective end-of-travel or home LIMFNC functions (R, S or T). These limit functions are not debounced, but the hardware status (see TLIM and LIM) is debounced with the Input Debounce Time (INDEB) command setting for I/O brick zero (default is 4 ms). The INDEB debounce is the period of time that the input must be held in a certain state before the drive recognizes it. This directly affects the rate at which the inputs can change state and be recognized. When a limit input is assigned a function other than its respective end-of-travel or home LIMFNC function (R, S or T), it is debounced the INDEB setting for I/O brick zero (default is 4 ms). If a limit is once again returned to its respective LIMFNC function, the debounce is removed.

**Input scan rate:** The limit inputs are scanned once per *system update* (2 milliseconds).

**Enabling & disabling inputs.** Limit inputs assigned an end-of-travel input function (functions R or S described below) are enabled/disabled with the LH command — the default is enabled. Limit input functions may be overridden with the LIMEN command — the default is enabled (no override).

**Multitasking.** If the LIMFNC command does not include the task identifier (%) prefix, the function affects the task that executes the LIMFNC command. The functions that may be directed to a task with % are: C, D, E, F, and P (e.g., 2%LIMFNC3-F assigns limit input 3 as a user fault input for task 2). Multiple tasks may share the same input, but the input may only be assigned one function.

### Identifier Function Description

- A **No special function** (general-purpose input). Status can be used with the LIM assignment/comparison operator.
- 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
Most Significant Bit Value	4

**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.

---

**Identifier    Function Description**

---

- C    **Kill.** Kills motion 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 (`LIMFNCi-E`). When enabled with the `ERROR` command, bit #6 of the `TER` and `ER` commands will report the kill status.
- D    **Stop.** Stops motion. If `COMEXS` is set to zero (`COMEXS0`), program execution will be terminated. If `COMEXS` is set to 1 (`COMEXS1`), command processing will continue. With `COMEXS` set to 2 (`COMEXS2`), program execution is terminated, but the `INSELP` value is retained. Motion deceleration during the stop is controlled by the `AD` & `ADA` commands. If error bit #8 is enabled (e.g., `ERROR.8-1`), activating a Stop input will set the error bit and cause a branch to the `ERRORP` program.
- E    **Pause/Continue.** If `COMEXR` is disabled (`COMEXR0`), 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 (and motion in `COMEXR1` mode).
- F    **User Fault.** Refer to the `ERROR` command. If error bit #7 is enabled (e.g., `ERROR.7-1`), activating a User Fault input will set the error bit and cause a branch to the `ERRORP` program. **CAUTION:** Activating the user fault input sends an `!K` command to the drive, "killing" motion (refer to the `K` command description for ramifications).
- G,H    **Reserved**
- I    **Alarm Event** - Will cause the drive to set an Alarm Event in the Communications Server over the Ethernet interface. You must first enable the Alarm checking bit for this input-driven alarm (`INTHW.23-1`). For details on using alarms, refer to the *Programmer's Guide*.
- J    **JOG positive-direction** - Will jog the axis in a positive-direction. The `JOG` command must be enabled for this function to work.
- K    **JOG negative-direction.** Will jog the axis in a negative-direction. The `JOG` command must be enabled for this function to work.
- 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.
- M    **Joystick Release.** Signals the drive to end joystick operation and resume program execution with the next statement in your program. When the input is open (high), the joystick mode is disabled (joystick mode can be enabled only if the input is closed, and only with the `JOY` command). When the input is closed (low), joystick mode can be enabled with the `JOY` command. The process of using Joystick mode is:
1. Assign the "Joystick Release" input function to a programmable input.
  2. At the appropriate place in the program, enable joystick control of motion (with the `JOY` command). (Joystick mode cannot be enabled unless the "Joystick Release" input is closed.) When the `JOY` command enables joystick mode, program execution stops (assuming the Continuous Command Execution Mode is disabled with the `COMEXC0` command).
  3. Use the joystick to move the axis as required.
  4. When you are finished using the joystick, open the "Joystick Release" input to disable the joystick mode. This allows program execution to resume with the next statement after the initial `JOY` command that started the joystick mode.
- N    **Reserved**

---

Identifier	Function Description
------------	----------------------

---

- **Joystick Velocity Select.** Allows you to select the velocity for joystick motion. The JOYVH and JOYVL commands establish two joystick velocities. Opening the Velocity Select input (input is high) selects the JOYVH configuration. Closing the Velocity Select input (input is low) selects the JOYVL configuration. The JOYVL velocity could be used to quickly move to a location, the JOYVH velocity could be used for low-speed accurate positioning. NOTE: When this input is not connected, joystick motion always uses the JOYVH velocity setting.
- aP **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. For example, in the LIMFNC1-3P command, 3 is the program number. An input defined as a Program Select Input will not function until the INSELP command has been enabled.

- Q **Program Security.** Issuing the LIMFNCi-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, LIMFNC, 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 LIMFNCi-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 LIMFNC1-Q command, limit input 3 is assigned the program access function and access to the DEF, DEL, ERASE, MEMORY, LIMFNC, and INFNC commands will be denied until you activate the input.

To regain access to these commands without the use of the program access input, you must issue the LIMEN command to disable the program security input, make the required user memory changes, and then issue the LIMEN command to re-enable the input. For example, if limit input 3 is assigned as the Program Security input, use LIMEN. 3=1 to disable the input and leave it activated, make the necessary user memory changes, and then use LIMEN. 3=E to re-enable the input.

**Limit Function Assignments:** The end-of-travel and home limit functions listed below (R, S & T) can only be assigned to their respective inputs (which is the factory default configuration). For example, limit input functions R, S & T must be assigned to limit inputs 1-3, respectively. If you attempt any other permutation of LIMFNC assignment of functions R, S & T, the drive will respond with the error message "INVALID LIMFNC COMBINATION".

- R **End-of-Travel Limit, Positive Direction.** This is the factory default function for limit input 1 found on the "DRIVE I/O" connector. If an onboard input or a digital input on an external I/O brick is assigned this function (e.g. 2INFNC1-R), then change the limit input's function to something else (e.g., change LIMFNC1-R to LIMFNC1-A). This end-of-travel limit input function is not debounced.
- S **End-of-Travel Limit, Negative Direction.** This is the factory default function for limit input 2 found on the "DRIVE I/O" connector. If an onboard input or a digital input on an external I/O brick is assigned this function (e.g. 2INFNC2-S), then change the limit input's function to something else (e.g., change LIMFNC2-S to LIMFNC2-A). This end-of-travel limit input function is not debounced.
- T **Home Limit.** This is the factory default function for limit input 3 found on the "DRIVE I/O" connector. If an onboard input or a digital input on an external I/O brick is assigned this function (e.g. 2INFNC3-T), then change the limit input's function to something else (e.g., change LIMFNC3-T to LIMFNC3-A). This home limit input function is not debounced.

---

**Example:**

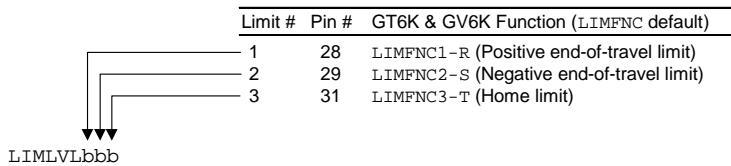
```
LIMFNC1-D          ; Redefine the positive EOT input (limit input #1)
                   ; to be a stop input
```

## LIMLVL Hardware Limit Input Active Level

Type	Limit (End-of-Travel and Homing)	Product	Rev
Syntax	<a_><!>LIMLVL<b><b><b>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	b = 0 (active low: requires n.c. EOT switch & n.o. home switch), 1 (active high: requires n.o. EOT switch & n.c. home switch), or X (don't care)		
Default	0		
Response	LIMLVL: *LIMLVL000		
See Also	[ AS ], LH, LIMEN, [ LIM ], LIMFNC, HOM, TAS, TLIM		

Use the LIMLVL command to define the active state the end-of-travel and home limits (found on the “DRIVE I/O” connector). The default state is active low.

Command syntax for limits on the DRIVE I/O connector (bits are numbered 1-3 from left to right):



Active Level Setting	Required Switch Type *	State	LIM/TLIM Report
Active low (LIMLVL0) <i>This is the default setting.</i>	End-of-travel limit: N.C. Home limit: N.O.	<b>Grounded</b> — sinking current (device driving the input is on)	1 (no limit hit)
		<b>Not Grounded</b> — not sinking current (device driving the input is off)	0 (FAULT: limit hit)
Active high (LIMLVL1)	End-of-travel limit: N.O. Home limit: N.C.	<b>Grounded</b> — sinking current (device driving the input is on)	0 (FAULT: limit hit)
		<b>Not Grounded</b> — not sinking current (device driving the input is off)	1 (no limit hit)

\* Compumotor recommends that all end-of-travel limit switches be normally-closed, because with normally-closed limit switches the drive detects a limit fault condition (and stops or inhibits motion) when the switch contact is open or if the wiring to the switch is broken.

Axis Status (AS, TAS, and TASF) bits 15 and 16 indicate when a hardware end-of-travel limit has been activated (i.e., invoking the “inhibit motion” function).

Wiring instructions and specifications for the limit inputs are provided in the *Hardware Installation Guide*.

## LJRAT System Load-to-Rotor/Forcer Inertia/Mass Ratio

Type	System	Product	Rev
Syntax	<a_><!>LJRAT<r>	GT6K	6.0
Units	Rotary motor: r = ratio of load inertia to motor rotor inertia Linear motor: r = ratio of load mass to forcer mass	GV6K	6.0
Range	0.0 - 100.0		
Default	0.0		
Response	LJRAT: *LJRAT4		
See Also	DACTDP, DELVIS, TGAIN, TSGSET		

The LJRAT command sets the system’s load-to-rotor inertia ratio (rotary motors) or load-to-forcer mass ratio (linear motors). The ratio is expressed in the following equation:

### Rotary Motors:

$$\text{LJRAT} = \text{load inertia} / \text{motor rotor inertia}$$

(Total system inertia = load inertia + motor rotor inertia)

### Linear Motors:

$$\text{LJRAT} = \text{load mass} / \text{forcer mass}$$

(Total system mass = load mass + forcer mass)

**NOTE:** LJRAT must be set in order for the Electronic Viscosity (DELVIS) and Active Damping (DACTDP) features to function properly.

**Working with servo gains.**

- Servo tuning process: refer to the *Hardware Installation Guide*.
- Check the values of all active gains (LJRAT is one of many servo gains): use TGAIN.
- Creating and invoking gain sets: see SGSET, SGENB, TGAIN, TSGSET.

<b>LN</b>		<b>End of Loop</b>	
Type	Loops or Program Flow Control	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>LN	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	No response; used in conjunction with the L command		
See Also	L, LX		

The 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:**

```
L5           ; Repeat the commands between L and LN five times
GO1         ; Start motion
LN          ; End loop
```

<b>LOCK</b>		<b>Lock Resource to Task</b>	
Type	Multi-tasking	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>LOCK<i,i>	GT6K	6.0
Units	1st i = resource number	GV6K	6.0
	2nd i = 1 (lock the resource) or 0 (unlock the resource)		
Range	1st i = 1 (COM1 port), 2 (COM2 port), or 3 (task swapping)		
	2nd i = 1 (lock the resource) or 0 (unlock the resource)		
Default	0 (= not locked)		
Response	LOCK (see example below)		
See Also	[ , ], DRPCHK, E, PORT, TSKTRN		

Use the LOCK command to make a resource available only to the specified task. The LOCK-able resources are:

- COM1 — the “RS-232/485” communication port or the “ETHERNET” communication port
- COM2 — the “RS-232” communication port
- Task Swapping — When task swapping is locked to a specific task, statements in all other tasks will not be executed until task swapping is again unlocked.

To check the LOCK status of all available resources, enter the LOCK command without field value. Below is an example response:

```
*LOCK1,0 COM PORT 1 - UNLOCKED
*LOCK2,0 COM PORT 2 - UNLOCKED
*LOCK3,0 TASK SWAPPING - UNLOCKED
```

See the *Programmer’s Guide* for more information about using the LOCK command with multi-tasking.

## NOTES

- If one task attempts to lock a resource in a different task (e.g., if Task1 attempts to execute the 2%LOCK1,1 command), the drive will respond with an error message ("ALTERNATE TASK NOT ALLOWED").
- If task "A" attempts to lock a resource that is already locked to task "B", command processing in task "A" will pause on the LOCK command, and SS bit 26 or 27 will be set ("SUSPENDED ON COMn"). When task "B" unlocks the resource, task "A" will be able to lock the resource and continue processing.
- If task "A" attempts to lock task swapping while any other task has locked a resource, command processing in task "A" will pause on the LOCK command, and SS bit 24 will be set ("SUSPENDED ON SWAP"). When all other tasks release locks on all other resources (COM1, COM2), task "A" will be able to lock task swapping and continue processing.
- A resource may be locked by a task only while that task is executing a program. If program execution is terminated for any reason (e.g., stop, kill, limit, fault, or just reaching the END of a program), all resources locked by that task will become unlocked.

**Example:**

```

LOCK1,1           ; Ensure exclusive COM1 access for the task executing
                  ; this program
WRITE"travel is" ; First part of output string
WRVAR1           ; Numeric value of travel
WRITE"inches."   ; Finish complete string
LOCK1,0           ; Allow other tasks access to COM1
    
```

## LS

### Soft Limit Enable

Type	Limit (End-of-Travel)	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>LS<i>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	i = 0 (disable both), 1 (disable positive-direction), 2 (disable negative-direction) or 3 (enable both)		
Default	0		
Response	LS: *LS0		
See Also	[ AS ], [ ER ], LH, LSAD, LSADA, LSNEG, LSPOS, PSET, TAS, TER, TSTAT		

The 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 (LSPOS and LSNEG), 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 negative- and positive-direction soft limits	i = 0
Enable negative-direction, disable positive-direction soft limit	i = 1
Enable positive-direction, disable negative-direction soft limit	i = 2
Enable negative- and positive-direction soft limits	i = 3

**NOTE:** The Gem6K drive maintains an absolute count, even though you may be programming in the incremental mode (MAØ). The soft limits will also function in preset mode (MCØ) or continuous mode (MC1). The soft limit position references the commanded position (TPC), not the position as measured by a feedback device, such as an encoder (TPE).

## 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**

```

LSPOS500000      ; Set soft limit positive-direction absolute position to be
                  ; 500000 units (Soft limits are always absolute)
LSNEG-500000     ; Set soft limit negative-direction absolute position to
                  ; be -500000 units (Soft limits are always absolute)
LS3              ; Enable both soft limits
LSAD100          ; Set soft limit decel to 100 units/sec/sec
PSET0            ; Set absolute position to 0
A10              ; Set accel to 10 units/sec/sec
V1              ; Set velocity to 1 unit/sec
D100000         ; Set distance to 100000 units
GO1              ; Initiate motion

```

---

## LSAD                      Soft Limit Deceleration

Type	Limit (End-of-Travel)	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>LSAD<r>	GT6K	6.0
Units	r = units/sec/sec (linear motors: see DMEPIT for linear/rotary conversion)	GV6K	6.0
Range	0.0001 – 9999.9999		
Default	100.0000		
Response	LSAD:     *LSAD100.0000		
See Also	DMEPIT, DRES, LHAD, LS, LSADA, LSNEG, LSPOS, SCALE, SCLA		

---

The LSAD command determines the value at which to decelerate after a programmed soft limit (LSPOS or LSNEG) has been hit.

**UNITS OF MEASURE and SCALING:** refer to page 16.

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); Limit (End-of-Travel)	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>LSADA<r>	GT6K	6.0
Units	r = units/sec/sec (linear motors: see DMEPIT for linear/rotary conversion)	GV6K	6.0
Range	0.0001 – 9999.9999		
Default	100.0000 (Default is a constant decel ramp, where LSADA tracks LSAD. S-curve decel is achieved when LSADA is changed independent of LSAD. To restore a constant decel ramp and tracking, set AA = 0.)		
Response	LSADA:     *LSADA100.0000		
See Also	AD, ADA, LS, DMEPIT, LHADA, LS, LSAD, SCALE, SCLA		

---

The 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*. Refer to page 15 for details on S-curve profiling.

Acceleration scaling (SCLA) affects LSADA the same as it does for LSAD. Refer to page 16 for details on scaling.

**Example:**

```

LSAD10           ; Set the maximum deceleration to 10 units/sec/sec
LSADA5           ; Set the average deceleration to 5 units/sec/sec

```



---

## LSNEG      Soft Limit Negative Travel Range

Type	Limit (End-of-Travel)	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>LSNEG<r>	GT6K	6.0
Units	r = units of distance	GV6K	6.0
Range	-999,999,999 to +999,999,999 (after conversion to steps)		
Default	+0		
Response	LSNEG:    *LSNEG+0		
See Also	LS, LSAD, LSADA, LSPOS, PSET, SCALE, SCLD		

---

The LSNEG command specifies the distance in absolute units where motion will be restricted when traveling in a negative-travel 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 LSPOS value greater than the LSNEG value.**

The LSNEG value remains set until you change it with a subsequent LSNEG command.

All soft limit values entered are in absolute distance counts. If scaling is enabled (SCALE1), LSNEG is internally multiplied by the distance scale factor (SCLD). The soft limit position references the commanded position (TPC), not the position as measured by a feedback device, such as an encoder (TPE).

**Example:** Refer to the soft limit enable (LS) command example.

---

## LSPOS      Soft Limit Positive Travel Range

Type	Limit (End-of-Travel)	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>LSPOS<r>	GT6K	6.0
Units	r = units of distance	GV6K	6.0
Range	-999,999,999 to +999,999,999 (after conversion to steps)		
Default	+0		
Response	LSPOS:    *LSPOS+0		
See Also	LS, LSAD, LSADA, LSNEG, PSET, SCALE, SCLD		

---

The LSPOS command specifies the distance in absolute units where motion will be restricted when traveling in a positive-travel 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 LSPOS value greater than the LSNEG value.**

The LSPOS value remains set until you change it with a subsequent LSPOS command.

All soft limit values entered are in absolute distance counts. If scaling is enabled (SCALE1), LSPOS is internally multiplied by the distance scale factor (SCLD). The soft limit position references the commanded position (TPC), not the position as measured by a feedback device, such as an encoder (TPE).

**Example:** Refer to the soft limit enable (LS) command example.

---

## LX      Terminate Loop

Type	Loops; Program Flow Control	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>LX	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	L, LN, PLN, PLOOP		

---

The 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:**

```

; *****
; This program will make the move specified by the GO1 command
; indefinitely until input 2 goes high, at which point, an LX will
; be issued, terminating the loop.
; *****
L0          ; Repeat the commands between L and LN infinitely, or until
           ; a Terminate Loop (LX) command is received
GO1         ; Start motion
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

```

---

<b>MA</b>		<b>Absolute/Incremental Mode Enable</b>	
Type	Motion	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>MA<b>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	b = 0 (incremental mode) or 1 (absolute mode)		
Default	0		
Response	MA: *MA0		
See Also	COMEXC, D, GO, PSET		

---

The MA command specifies whether the subsequent motion will be made with respect to current position (incremental mode) or with respect to an absolute zero position (absolute mode).

In incremental mode (MA0), 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.

**ON-THE-FLY CHANGES:** You can change positioning modes *on the fly* (while motion is in progress) in two ways. One way is to send an immediate command (!MA) followed by an immediate go command (!GO). The other way is to enable the continuous command execution mode (COMEXC1) and execute a buffered command (MA) followed by a buffered go command (GO).

**Example:**

```

PSET0      ; Set the present position to be absolute position zero
MA1        ; Enable absolute positioning mode
A2         ; Set acceleration to 2 revs/sec/sec
AD7        ; Set deceleration to 7 revs/sec/sec
V1         ; Set velocity to 1 rev/sec
D4000      ; Set the setpoint distance to absolute position 4000
GO1        ; Start motion: (move 4000 counts in the positive direction)
D2000      ; Set the setpoint distance to absolute position 2000
GO1        ; Start motion: (move 2000 counts in the negative direction)
D8000      ; Set the setpoint distance to absolute position 8000
GO1        ; Start motion: (move 6000 counts in the positive direction)
MA0        ; Enable incremental positioning mode
D4000      ; Set incremental distance to 4000 counts, positive direction
GO1        ; Start motion: (move 4000 counts in the positive direction
           ; and finish at absolute position 12000)
GO1        ; Start motion again: (move 4000 counts in the positive direction
           ; and finish at absolute position 16000)

```

---

## MC

### Preset/Continuous Mode Enable

		Product	Rev
Type	Motion	GT6K	6.0
Syntax	<a_><!>MC<b>	GV6K	6.0
Units	n/a		
Range	b = 0 (preset mode) or 1 (continuous mode)		
Default	0		
Response	MC: *MC0		
See Also	A, AD, COMEXC, COMEXS, D, FOLMD, [ FS ], FSHFC, FSHFD, GO, GOBUF, K, MA, PSET, S, TFS, V		

---

The MC command causes subsequent moves to go a specified distance (MC0), or a specified velocity (MC1).

In the Preset Mode (MC0), 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.

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.

**ON-THE-FLY CHANGES:** You can change positioning modes *on the fly* (while motion is in progress) in two ways. One way is to send an immediate command (!MC) followed by an immediate go command (!GO). The other way is to enable the continuous command execution mode (COMEXC1) and execute a buffered command (MC) followed by a buffered go command (GO).

#### Example:

```
DEL demo           ; Delete program demo
DEF demo           ; Begin definition of program demo
COMEXC0            ; Disable continuous command processing mode
MA0                ; Enable incremental positioning mode
MC0                ; Enable preset positioning mode
A20                ; Set acceleration to 20 revs/sec/sec
AD7                ; Set deceleration to 7 revs/sec/sec
V2                ; Set velocity to 2 revs/sec
D10000            ; Set distance to 10,000 counts
GO1                ; Initiate motion (move 10,000 counts in positive direction)
COMEXC1            ; Enable continuous command processing mode
MC1                ; Enable continuous positioning mode
A12                ; Set acceleration to 12 revs/sec/sec
AD8                ; Set deceleration to 8 revs/sec/sec
V5                ; Set velocity to 5 revs/sec
GO1                ; Initiate motion (travel at a velocity of 5 revs/sec)
T5                ; Wait 5 seconds
V9                ; Set velocity to 9 revs/sec (when the next GO is executed)
GO1                ; Change to the new velocity of 9 revs/sec
T2                ; Wait 2 seconds
V0                ; Set velocity to zero
GO1                ; Go to zero velocity
WAIT(AS.1=b0)     ; Wait until no motion is commanded
COMEXC0            ; Disable continuous command processing mode
END                ; End definition of program demo
```

## MEMORY Partition User Memory

Type	Drive Configuration	Product	Rev
Syntax	<a_><!>MEMORY<i>,<i>	GT6K	6.0
Units	i = bytes of memory (use even number only) 1st <i> = partition for "Programs" 2nd <i> = partition for "Compiled Profiles"	GV6K	6.0
Range	(see table below)		
Default	(see table below)		
Response	MEMORY: *MEMORY150000,150000		
See Also	[ DATP ], DEF, GOBUF, PCOMP, PLCP, [ SEG ], [ SS ], TDIR, TMEM, TSEG, TSS		

The Gem6K's memory has two partitions: one for storing *programs* and one for storing *compiled profiles & PLC programs*. The allocation of memory to these two areas is controlled with the MEMORY command.

"Programs" vs. "Compiled Profiles & Programs"
<u>Programs</u> are defined with the DEF and END commands, as demonstrated in the "Program Development Scenario" in the <i>Programmer's Guide</i> .
<u>Compiled Profiles &amp; PLC Programs</u> are defined like programs (using the DEF and END commands), but are compiled with the PCOMP command and executed with the PRUN command (but PLCP programs are usually executed with SCANP). Compiled profiles/programs could be a <i>profile</i> (a series of GOBUF commands), or a <i>PLC program</i> (for PLC Scan Mode).  Programs intended to be compiled are stored in program memory. After they are compiled with the PCOMP command, they remain in program memory and the <i>segments</i> (see diagram below) from the compiled program are stored in compiled memory. The TDIR report indicates which programs are compiled as compiled profiles ("COMPILED AS A PATH") and which programs are compiled as PLC programs ("COMPILED AS A PLC PROGRAM").  For more information on compiled profiles (Compiled Motion Profiling), and PLC Scan Mode, refer to the <i>Programmer's Guide</i> .

### MEMORY Syntax:

MEMORY150000,150000

Memory allocation for Programs (bytes). Storage requirements depend on the number of ASCII characters in the program.

Memory allocation for Compiled Profiles & Programs (bytes). Storage requirements depend on the number of segments (1 segment consumes 76 bytes). A segment could be one of these commands:

Compiled Motion:	PLC (PLCP) Program:
GOBUF *	IF **
PLOOP	ELSE
GOWHEN	NIF
TRGFN	L
POUTA	LN
	OUT
	ANO
	EXE
	PEXE
	VARI **
	VARB **

\* GOBUF commands may require up to 4 segments.

\*\* IF statement may require at least 2 segments, each AND or OR compound requires an additional segment. VARI and VARB each require 2 segments.

## Allocation Defaults and Limits:

The following table identifies memory allocation defaults and limits for the Gem6K drive. When specifying the memory allocation, use only even numbers. The minimum storage capacity for one partition area (program or compiled) is 1,000 bytes.

Feature	Gem6K
Total memory (bytes)	300,000
Default allocation (program, compiled)	150000, 150000
Maximum allocation for programs	299000, 1000
Maximum allocation for compiled profiles/programs	1000, 299000
Max. # of programs	400
Max. # of labels	600
Max. # of compiled profiles	300
Max. # of compiled profile segments	2069
Max. # of numeric variables	225
Max. # of integer variables	225
Max. # of string variables	25
Max. # of binary variables	125

When teaching variable data to a data program (DATP), be aware that the memory required for each data statement of four data points (43 bytes) is taken from the memory allocation for program storage.

### CAUTION

Issuing a memory allocation command (e.g., `MEMORY200000,100000`) will erase all existing programs and compiled segments. However, issuing the `MEMORY` command by itself (e.g., type `MEMORY <cr>` by itself to request the status of how the memory is allocated) will not affect existing programs or compiled segments.

## Checking Memory Status:

To find out what programs reside in your drive's memory, and how much of the available memory is allocated for programs and compiled profile segments, issue the `TDIR` command (see example response below). Entering the `TMEM` command or the `MEMORY` command (without parameters) will also report the available memory for programs and compiled profile segments.

Sample response to `TDIR` command:

```
*1 - SETUP USES 345 BYTES
*2 - PIKPRT USES 333 BYTES
*14322 OF 150000 BYTES (98%) PROGRAM MEMORY REMAINING
*1973 OF 1973 SEGMENTS (100%) COMPILED MEMORY REMAINING
```

Two system status bits (reported with the `TSS`, `TSSF` and `SS` commands) are available to check when compiled profile segment storage is 75% full or 100% full. System status bit #29 is set when segment storage reaches 75% of capacity; bit #30 indicates when segment storage is 100% full.

## Example:

```
MEMORY200000,100000 ; Set aside 200,000 bytes for program storage,
                    ; 100,000 bytes for compiled profile segments
```

---

## MEPOL Master Encoder Polarity

Type	Encoder; Following; Drive Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>MEPOL<b>	GT6K	6.0
Units	b = polarity bit	GV6K	6.0
Range	b = 0 (normal polarity), 1 (reverse polarity), or X (don't care)		
Default	0		
Response	MEPOL: *MEPOL0		
See Also	[ PCME ], [ PCMS ], [ PMAS ], [ PME ], PASET, TPCME, TPME, TPCMS, TPCMAS		

---

Use the MEPOL command to reverse the counting direction (polarity) of the Master Encoder input (the encoder connector labeled “Master Encoder”). This allows you to reverse the counting direction without having to change the actual wiring to the encoder input.

Immediately after issuing the MEPOL command, the master encoder will start counting in the opposite direction (including all master encoder position registers).

The MEPOL command is automatically saved in non-volatile RAM.

---

## MESND Master Encoder Step and Direction Mode

Type	Encoder; Counter; Following	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>MESND<b>	GT6K	6.0
Units	b = enable bit	GV6K	6.0
Range	b = 0 (quadrature signal), 1 (step & direction), or X (don't care)		
Default	0		
Response	MESND: *MESND0		
See Also	[ PME ], TPME		

---

Use the MESND command to specify the functionality of the Master Encoder input.

MESND0.....(default setting) accept a quadrature signal from the master encoder.

MESND1.....Accept step and direction signals. The count is registered on a positive edge of a transition for a signal measured on encoder channel A+ and A- connections. The direction of the count is specified by the signal on encoder channel B+ and B- connections. Therefore, you should connect your step and direction input device as follows: Connect Step+ to A+, Step- to A-, Direction+ to B+, and Direction- to B-.

---

## [ MOV ] Axis Moving Status

Type	Assignment or Comparison	<b>Product</b>	<b>Rev</b>
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	[ AS ], GO, TAS		

---

The 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, TASF 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=b1), or IF(MOV=h1)

If the axis is in motion, the bit will be a one (1). If the axis is not in motion, the bit will be a zero (0).

**Example:**

```
COMEXC1      ; Enable continuous command processing mode
COMEXS1      ; Save command buffer on stop
MC1          ; Enable continuous mode
A2           ; Set acceleration to 2 units/sec/sec
AD2          ; Set deceleration to 2 units/sec/sec
V1           ; Set velocity to 1 units/sec
GO1          ; Initiate motion
T5           ; Wait 5 seconds
S1           ; Stop motion
WAIT(MOV=b0) ; Wait for motion to come to a halt
COMEXC0      ; Disable continuous command processing mode
```

---

**NIF**                      **End IF Statement**

		<b>Product</b>	<b>Rev</b>
Type	Program Flow Control or Conditional Branching		
Syntax	<a_><!>NIF	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	No response when used in conjunction with the IF command		
See Also	ELSE, IF		

---

The NIF 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, and 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, and 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 a NIF command has already been encountered.

**Example:**

```
IF(IN=b1X0)  ; If input 1 is ON and input 3 is OFF (IF statement evaluates
              ; true), run program #13. If the IF statement evaluates false,
              ; turn on output #3.
  PROG13     ; Run program #13
ELSE         ; If the IF condition evaluates false, execute the subsequent
              ; commands until the NIF.
  OUTxx1     ; Turn on output #3
NIF          ; End IF statement
```

---

## [ NMCY ] Master Cycle Number

Type	Following; Assignment or Comparison	<b>Product</b>	<b>Rev</b>
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	FMCLen, FMCNEW, FMCP, [ FS ], [ PMAS ], TFS, TNMCY, TRGFN		

---

The NMCY command is used to assign the current master cycle number to a numeric variable, or to make a comparison against another value. The master must be assigned first (FOLMAS command) before this command will be useful. For a complete discussion of master cycles, refer to the Following chapter in the *Programmer's Guide*.

The value represents the current cycle number, not the position of the master (or the follower). The master cycle number is set to zero when master cycle counting is restarted, and is incremented each time a master cycle finishes (i.e., rollover occurs). It will often correspond to the number of complete parts in a production run. This value may be used for subsequent decision making, or simply recording the cycle number corresponding to some other event.

**Syntax:** VARn=NMCY where “n” is the variable number, or NMCY can be used in an expression such as IF(NMCY>=5).

**Example:**

```
IF(NMCY>500)      ; If the master has moved through 500 cycles ...
WRITE"500 cycles have occurred" ; Send string to serial port
NIF              ; End of IF statement
VAR12=NMCY       ; Set VAR12 to equal the number of cycles that have occurred on
                 ; the master
```

---

## [ NOT ] Not

Type	Operator (Logical)	<b>Product</b>	<b>Rev</b>
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	[ AND ], IF, NWHILE, [ OR ], REPEAT, UNTIL, WAIT, WHILE		

---

The NOT operator is used in conjunction with the program flow control commands (IF, REPEAT..UNTIL, WHILE. .NWHILE, WAIT). The NOT operator complements 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 is 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:**

```
VAR1=1           ; Set variable 1 equal to 1
WHILE(NOT VAR1=10) ; 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
```



---

## NTADDR Ethernet IP Address

Type	Communication Interface	<b>Product</b>	<b>Rev</b>
Syntax	NTADDR<i,i,i,i>	GT6K	6.0
Units	i,i,i,i = IP address (commas are used in place of periods)	GV6K	6.0
Range	i = 0-255		
Default	192,168,10,30 (network address is 192.168.10.30)		
Response	NTADDR: *192,168,10,30		
See Also	TNTMAC		

---

Use the NTADDR command to change the Gem6K drive's IP address (e.g., to correct an IP address conflict). **NOTE:** The Gem6K drive needs to be reset (cycle power or issue RESET command) in order for the new address to take effect.

The NTADDR setting is automatically saved in battery backed RAM.

---

## NTFEN Ethernet Communication Enable

Type	Communication Interface	<b>Product</b>	<b>Rev</b>
Syntax	NTFEN<b>	GT6K	6.0
Units	b = Ethernet enable bit	GV6K	6.0
Range	0 (disabled), 1 (enabled for Windows 95/98, no File Sharing, closed network), 2 (enabled - recommended for most Ethernet configurations)		
Default	0 (Ethernet communication is disabled)		
Response	NTFEN: *0		
See Also	NTADDR, NTMASK, TNT, TNTMAC		

---

Use the NTFEN command to enable or disable Ethernet communication. The factory default configuration is that Ethernet communication is disabled (NTFEN0). To enable Ethernet communication, you must connect to the Gem6K's "RS-232" serial port (COM2) and send the NTFEN command which is appropriate for your Ethernet configuration (see options below). Then you can communicate over the Ethernet connection.

### Options:

- NTFEN0: Use NTFEN0 to disable Ethernet communication.
- NTFEN1: NTFEN1 is primarily intended for use with Windows 95/98 with no File Sharing and a closed network.
- NTFEN2: NTFEN2 is recommended as the standard Ethernet communication mode, even if you are using a closed network and no file sharing. NTFEN2 is especially useful if you are using Windows NT, or Windows 95/98 with File Sharing and/or an open network.  
**NOTE:** When using NTFEN2, you must also follow the ARP -S Static Mapping procedure (see *Programmer's Guide*).

**NOTE:** You can communicate to either the "ETHERNET" port or the "RS-232/485" port at one time.

The NTFEN setting is automatically saved in battery backed RAM. Thus, if you cycle power, Ethernet communication will still be enabled.

If the Ethernet connection fails, the Gem6K will set error status bit #22 (see ER, TER, and TERF) if error-checking bit #22 is enabled with the ERROR command. Also, if this error occurs, the Gem6K will branch to the ERRORP program.

---

## NTMASK Ethernet Network Mask

Type	Communication Interface	<b>Product</b>	<b>Rev</b>
Syntax	NTMASK<i,i,i,i>	GT6K	6.0
Units	i,i,i,i = mask	GV6K	6.0
Range	i = 0-255		
Default	255,255,255,0		
Response	NTMASK: *255,255,255,0		
See Also	NTADDR, TNTMAC		

---

Use the NTMASK command to configure the Gem6K drive's network mask. **NOTE:** The Gem6K drive needs to be reset (cycle power or issue RESET command) in order for the new network mask to take effect.

The NTMASK setting is automatically saved in battery backed RAM.

---

## NTSFS Ethernet Send Fast Status Packet

Type	Communication Interface	<b>Product</b>	<b>Rev</b>
Syntax	NTSFS	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	<u>NTFEN</u>		

---

The NTSFS command allows a Gem6K program to send a *Fast Status packet*, as needed, from the Gem6K drive to a PC application (COM6SRVR). In a typical application, a PC may be controlling a machine and using the COM6SRVR's *Variable Packets* to send variables and "job" instructions to the Gem6K. When the job is complete, the Gem6K could execute the NTSFS command to send a Fast Status packet back to the PC, thus indicating that the job was completed. Using the VarI and VarB elements of the Fast Status packet allows the Gem6K to communicate that the job was completed satisfactorily or that a specific error caused the job to fail.

For details on using Fast Status and COM6SRVR, refer to the *Communication Server Programmer's Reference*.

---

## NWHILE End WHILE Statement

Type	Program Flow Control or Conditional Branching	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>NWHILE	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	No response when used in conjunction with the WHILE command		
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:**

```

WHILE(IN=b1X0)      ; While input 1 = 1, input 3 = 0, execute commands between WHILE & NWHILE
T5                 ; Wait 5 seconds
TPE                ; Transfer position of encoder
NWHILE             ; End WHILE statement

```

---

## ONCOND    On Condition Enable

Type	On Condition (Program Interrupt)	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!><%>ONCOND<b><b><b><b>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	b = 0 (disable), 1 (enable) or X (don't change)		
Default	0		
Response	ONCOND: *ONCOND0000		
See Also	FSHFD, ONIN, ONP, ONUS, ONVARA, ONVARB, [ SS ], TSS		

---

The 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

```

**When ON conditions WILL NOT interrupt immediately:** These are situations in which an ON condition does not immediately interrupt the program in progress. However, the fact that the ON condition evaluated true is retained, and when the condition listed below is no longer preventing the interrupt, the interrupt will occur.

- While a WAIT statement is in progress
- While a time delay (T) is in progress
- While a program is being defined (DEF)
- While a pause (PS) is in progress
- While a data read (DREAD, DREADF, or READ) is in progress
- While motion is in progress due to GO, GOWHEN, HOM, JOY, JOG, or PRUN and the continuous command execution mode is disabled (COMEXCØ).

**Multi-Tasking:** Each task has its own ONP Program and its own set of On conditions.

**Example:**

```

DEF bigmov        ; Define program bigmov
D20              ; Sets move distance to 20 units
GO1              ; Initiate motion
END              ; End program definition
ONP bigmov       ; Set ON program to bigmov
2ONINxxx1       ; When input #4 on I/O brick 2 is activated,
                ; GOSUB to the ONP program
ONCOND1000      ; 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)	<b>Product</b>	<b>Rev</b>
Syntax	<a_!><%><@><B>ONIN<b><b><b>...<b><b>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	b = 0 (disable), 1 (enable) or X (don't care)		
Default	X		
Response	ONIN: *ONINXXXX_X 1ONIN: *1ONINXXXX_XXXX_XXXX_XXXX_XXXX_XXXX_XXXX_XXXX		
See Also	INFNC, ONCOND, ONP, TIN		

---

The 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.

The number of external inputs available varies by the configuration of I/O bricks used. Refer to page 8 for details.

The ONIN command must be enabled using the ONCOND command before any branching will occur. Once a branch to the ONP program occurs, the 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.

**Multi-Tasking:** Each task has its own ONP Program and its own set of On conditions. Only 1 ONIN condition is allowed per task. Therefore, only one I/O brick can be referenced in an ONIN condition for a specific task.

#### Example:

```
DEF bigmov          ; Define program bigmov
D20                 ; Set move distance to 20 units
GO1                 ; Initiate motion
END                 ; End program definition
ONP bigmov          ; Set ON program to bigmov
2ONINxxx11xx1      ; When inputs 4, 5, and 8 on I/O brick 2 are activated,
                   ; GOSUB to the ONP program
ONCOND1000         ; Enable ONIN condition
;
; Now that the ONP program named bigmov is defined, if inputs 4, 5, and 8 on
; I/O brick 2 become active during normal program operation, the program will
; GOSUB to the ONP program (bigmov).
```

---

## ONP

### On Condition Program Assignment

Type	On Condition (Program Interrupt)	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!><%>ONP<t>	GT6K	6.0
Units	t = text (name of On Condition program)	GV6K	6.0
Range	text name of 6 characters or less		
Default	n/a		
Response	ONP: *ONP bigmov		
See Also	DEF, END, ONCOND, ONIN, ONUS, ONVARA, ONVARB		

---

The 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.

**Multi-Tasking:** Each task has its own ONP Program and its own set of On conditions.

**Example:**

```

DEF bigmov          ; Define program bigmov
D20                 ; Sets move distance to 20 units
GO1                 ; Initiate motion
END                 ; End program definition
ONP bigmov          ; Set ON program to bigmov
2ONIN.4-1           ; When input #4 on I/O brick 2 is activated,
                   ; GOSUB to the ONP program
ONCOND1000         ; 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**

Type	On Condition (Program Interrupt)	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!><%>ONUS<b><b><b>...<b><b> (16 bits)	GT6K	6.0
Units	n/a	GV6K	6.0
Range	b = 0 (disable), 1 (enable) or X (don't care)		
Default	0		
Response	ONUS:    *ONUS0000_0000_0000_0000		
See Also	INDUSE, INDUST, ONCOND, ONP		

---

The 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.

**Multi-Tasking:** Each task has its own ONP Program and its own set of On conditions.

**Example:**

```

INDUSE1             ; Enable user status
INDUST1-5A          ; User status bit 1 defined as axis status bit 5
INDUST2-3A          ; User status bit 2 defined as axis status bit 3
3INDUST3-5J         ; User status bit 3 defined as input 5 on I/O brick 3
INDUST4-14A         ; User status bit 4 defined as axis status bit 14
2%INDUST16-2I       ; User status bit 16 defined as system status bit 2 for task 2
DEF bigmov          ; Define program bigmov
D20                 ; Sets move distance to 20 units
GO1                 ; Initiate motion
END                 ; End program definition
ONP bigmov          ; Set ON program to bigmov
ONUSxxx1           ; On user status bit #4 (axis status bit 14) GOSUB to
                   ; the ONP program
ONCOND0100         ; Enable ONUS condition

```

---

## ONVARA On Variable 1 Condition Gosub

Type	On Condition (Program Interrupt)	Product	Rev
Syntax	<a_!><%>ONVARA<i,i,i>	GT6K	6.0
Units	See below	GV6K	6.0
Range	±999,999,999.99999999		
Default	+0.0,+0.0,+0.0		
Response	ONVARA: *ONVARA+0.0,+0.0,+0.0		
See Also	ONCOND, ONP, ONVARB, VAR, VARI		

---

The 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. If the third field is non-zero, integer variables (VARI) are used for the comparison.

The ONVARA command must be enabled using the ONCOND command before any branching will occur. Once a branch to the ONP program occurs, the 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.

**Multi-Tasking:** Each task has its own ONP Program and its own set of On conditions.

### Example:

```
DEF bigmov          ; Define program bigmov
D20                 ; Sets move distance to 20 units
GO1                 ; Initiate motion
END                 ; End program definition
ONP bigmov          ; Set ON program to bigmov
ONVARA0,12          ; On VAR1 <= 0, or VAR1 >= 12 GOSUB to ONP program
ONCOND0010         ; Enable ONVARA condition
```

---

## ONVARB On Variable 2 Condition Gosub

Type	On Condition (Program Interrupt)	Product	Rev
Syntax	<a_><!><%>ONVARB<i,i,i>	GT6K	6.0
Units	See below	GV6K	6.0
Range	±999,999,999.99999999		
Default	+0.0,+0.0,+0.0		
Response	ONVARB: *ONVARB+0.0,+0.0,+0.0		
See Also	ONCOND, ONP, ONVARA, VAR, VARI		

---

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. If the third field is non-zero, integer variables (VARI) are used for the comparison.

The ONVARB command must be enabled using the ONCOND command before any branching will occur. Once a branch to the ONP program occurs, the 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.

**Multi-Tasking:** Each task has its own ONP Program and its own set of On conditions.

### Example:

```
DEF bigmov          ; Define program bigmov
D20                 ; Sets move distance to 20 units
GO1                 ; Initiate motion
END                 ; End program definition
ONP bigmov          ; Set ON program to bigmov
ONVARB0,12          ; On VAR2 <= 0, or VAR2 >= 12 GOSUB to ONP program
ONCOND0001         ; Enable ONVARB condition
```

## [ OR ]

Or

Type	Operator (Logical)	Product	Rev
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	[ AND ], IF, [ NOT ], NWHILE, REPEAT, UNTIL, WAIT, WHILE		

Use the OR command as a logical operator in a program flow control command (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 is illustrated by the following examples.

If VAR1=1 and VAR2=1 then, even though variable 2 is not greater than 3, this is a true statement: IF(VAR1>0 OR VAR2>3). This 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:

```
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
```

## ORES

Resolution of Step & Direction Output (GT6K) or Encoder Output (GV6K)

Type	Drive Configuration; Outputs	Product	Rev
Syntax	<a_><!>ORES<i> (does not take effect until RESET, DRESET or cycle power)	GT6K	6.0
Units	Rotary motors: i = counts/rev Linear motors: i = counts/electrical pitch	GV6K	6.0
Range	GT6K: 200 - 128000; 0 to disable GV6K: 200 - 1073741823; 0 to disable		
Default	GT6K: 0 GV6K: 4000		
Response	ORES *ORES4000		
See Also	DMEPIT, DMTR, DRES, ERES, TASX, TER, TPC, TPE		

**NOTE:** This command does not take effect until you cycle power to the drive, or issue a RESET or DRESET command.

**AUTO-SETUP for GV6K Drives:** This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6). If you did not use the configuration utility or are not using a Parker Motor, this command is set to zero and you will have to manually set this parameter. Refer to DMTR for a list of auto-configured commands.

**NOTE:** ORES0 disables this output function.

**GT6K:** The ORES command sets the resolution of the auxiliary step & direction output (pins 14-17 on the DRIVE I/O connector).

**GV6K:** The ORES command sets the resolution of the encoder output (pins 14-19 on the DRIVE I/O connector).

### Maximum Frequency

The Maximum output frequency (post quadrature) of the hardware generating the ORES functionality is 2.5 MHz. If the maximum output frequency is exceeded, the drive will fault

(causes a DRIVE0 and sets TASX bit #29). The following equation dictates the maximum allowable ORES value for a given maximum motor velocity:

$$\text{ORES} \leq \frac{2500000}{\text{Maximum Velocity}}$$

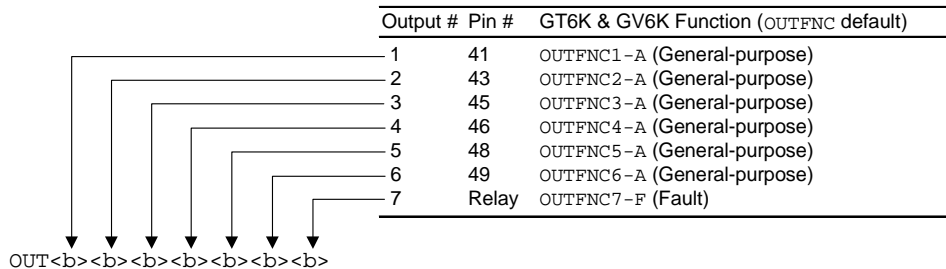
For example, if a rotary application requires a maximum velocity of 100 revs/sec, the maximum ORES value would be ORES25000.

**Maximum Frequency in Bypass Mode (GV6K only):** When ORES = ERES, the encoder bypass mode is activated, which allows the native encoder counts to pass directly through the drive. The maximum encoder frequency in encoder bypass is 2.0 MHz pre-quadrature (8 MHz post-quadrature). For resolver motors, the native resolution of the R/D converter (4096 counts/rev post quadrature) is passed through directly.

Refer to the *Hardware Installation Guide* for specifications and installation instructions for the encoder output.

OUT	Output State	Product	Rev
Type	Outputs	GT6K	6.0
Syntax	<a_><!><@><B>OUT<b><b><b>...<b><b><b>	GV6K	6.0
Units	n/a		
Range	b = 0 (off), 1 (on) or X (don't change)		
Default	0		
Response	n/a		
See Also	OUTALL, OUTEN, OUTFNC, OUTLVL, OUTP, TIO, TOUT		

You can use the OUT command to turn on/off one or more of the digital outputs on the DRIVE I/O connector, as well as the relay output (labeled “RELAY COM” and “RELAY N.O.”) on the 4-pin removable connector.



You may also use the OUT command to control outputs on external I/O bricks.

**NOTE:** You may use OUT to control only the outputs defined as “general-purpose” outputs with the OUTFNCi-A command. If you attempt to change the state of an output that is not defined as general-purpose output, the drive will respond with an error message (“OUTPUT BIT USED AS OUTFNC”) and the OUT command will not be executed (but command processing will continue)

The number of external outputs varies by the configuration of I/O bricks used. Refer to page 8 for details.

If you wish to affect only one output, instead of all outputs, use the bit select (.) operator followed by the number of the specific output. For example, OUT . 5-1 turns on output #5.



## Onboard Outputs (on the “DRIVE I/O” connector) — Relationships:

Relationships:	OUTLVL Setting	OUT State *	Current	TOUT status
	OUTLVL0 (default)	OUT1	Sinking current	1
	OUTLVL0	OUT0	No current flow	0
	OUTLVL1	OUT1	No current flow	1
	OUTLVL1	OUT0	Sinking current	0

\* The output is “active” when it is commanded by the OUT, OUTP, or POUT command (for example, OUT<sub>xx</sub>1 activates output #3).

## Expansion I/O Outputs (on an EVM32) — Interrelationships:

Output Type	OUTLVL Setting	OUT State *	Current	OUT/TOUT status	LED
NPN (sinking)	OUTLVL0 (default)	OUT1	Sinking current	1	ON
NPN	OUTLVL0 (default)	OUT0	No current flow	0	OFF
NPN	OUTLVL1	OUT1	No current flow	1	OFF
NPN	OUTLVL1	OUT0	Sinking current	0	ON
PNP (sourcing)	OUTLVL0	OUT1	No current flow	1	OFF
PNP	OUTLVL0	OUT0	Sourcing current	0	ON
PNP	OUTLVL1 (default)	OUT1	Sourcing current	1	ON
PNP	OUTLVL1 (default)	OUT0	No current flow	0	OFF
Reed Relay	OUTLVL0 (default)	OUT1	Sinking (closed)	1	ON
Reed Relay	OUTLVL0 (default)	OUT0	No flow (open)	0	OFF
Reed Relay	OUTLVL1	OUT1	No flow (open)	1	OFF
Reed Relay	OUTLVL1	OUT0	Sinking (closed)	0	ON

\* The output is “active” when it is commanded by the OUT, OUTP, or POUT command (for example, OUT<sub>xx</sub>1 activates output #3).

### Example:

```

OUTFNC1-A      ; Define output #1 as a general-purpose output
OUTFNC2-A      ; Define output #2 as a general-purpose output
OUT11          ; Turn on outputs 1 & 2
OUT.2-0        ; Turn off output 2
1OUT.9-1       ; Turn on output 9 (the 1st I/O point on SIM2) on I/O brick 1
    
```

## [ OUT ]

### Output Status

Type	Assignment or Comparison	Product	Rev
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	b = 0 (off), 1 (on) or X (don't change)		
Default	0		
Response	n/a		
See Also	OUTALL, OUTEN, OUTFNC, OUTLVL, TIO, TOUT, VARB		

Use the OUT operator 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, 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:** VARB<sub>n</sub>=<B>OUT where “n” is the binary variable number and “<B>” is the number of the I/O brick where the output resides (not required if addressing the onboard outputs), or OUT can be used in an expression such as IF (2OUT=b1101), or IF (1OUT=h7F)

The number of external outputs varies by I/O bricks used. See page 8 for details.

The function of the outputs is established with the OUTFNC command (although the OUT operator 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=2OUT.12 assigns output 12 (the 4<sup>th</sup> I/O point on SIM2) on I/O brick 2 to binary variable 1.

### Onboard Outputs (on the “DRIVE I/O” connector) — Relationships:

Relationships:	OUTLVL Setting	OUT State *	Current	TOUT status
	OUTLVL0 (default)	OUT1	Sinking current	1
	OUTLVL0	OUT0	No current flow	0
	OUTLVL1	OUT1	No current flow	1
	OUTLVL1	OUT0	Sinking current	0

\* The output is “active” when it is commanded by the OUT, OUTP, or POUT command (for example, OUTxx1 activates output #3).

### Expansion I/O Outputs (on an EVM32) — Interrelationships:

Output Type	OUTLVL Setting	OUT State *	Current	OUT/TOUT status	LED
NPN (sinking)	OUTLVL0 (default)	OUT1	Sinking current	1	ON
NPN	OUTLVL0 (default)	OUT0	No current flow	0	OFF
NPN	OUTLVL1	OUT1	No current flow	1	OFF
NPN	OUTLVL1	OUT0	Sinking current	0	ON
PNP (sourcing)	OUTLVL0	OUT1	No current flow	1	OFF
PNP	OUTLVL0	OUT0	Sourcing current	0	ON
PNP	OUTLVL1 (default)	OUT1	Sourcing current	1	ON
PNP	OUTLVL1 (default)	OUT0	No current flow	0	OFF
Reed Relay	OUTLVL0 (default)	OUT1	Sinking (closed)	1	ON
Reed Relay	OUTLVL0 (default)	OUT0	No flow (open)	0	OFF
Reed Relay	OUTLVL1	OUT1	No flow (open)	1	OFF
Reed Relay	OUTLVL1	OUT0	Sinking (closed)	0	ON

\* The output is “active” when it is commanded by the OUT, OUTP, or POUT command (for example, OUTxx1 activates output #3).

#### Example:

```
VARB1=OUT           ; Output status assigned to binary variable 1
VARB2=OUT.4        ; On-board output bit 4 assigned to binary variable 2
VARB2              ; Response if bit 4 is set to 1:
                  ; *VARB2=XXX1_XXX
IF(OUT=b110X1)    ; If the output status contains 1's for outputs 1, 2, & 5,
                  ; and a 0 for output 4, do the IF statement
TREV              ; Transfer revision level
NIF              ; End IF statement
```

---

## OUTALL Output State for Multiple Outputs

Type	Output	Product	Rev
Syntax	<a_><!><@><B>OUTALL<i>,<i>,<b>	GT6K	6.0
Units	1st i = beginning number of output range 2nd i = ending number of output range b = enable/disable bit	GV6K	6.0
Range	1st i = 1 to n (n is max. number of outputs available) 2nd i = First i to n b = 0 (off) or 1 (on)		
Default	0		
Response	n/a		
See Also	OUT, OUTEN, OUTFNC, OUTLVL, TIO, TOUT		

---

The OUTALL command turns a range of output bits on and off. You may use this command to control any contiguous range of the onboard outputs, as well as any outputs on external I/O bricks, as long as all outputs in the range are defined as OUTFNCi-A (general-purpose) outputs. If you attempt to change the state of an output that is not defined as OUTFNCi-A, the drive will respond with an error message (“OUTPUT BIT USED AS OUTFNC”) and the OUTALL command will not be executed (but command processing will continue).

The number of external outputs varies by configuration of I/O bricks used. Refer to page 8 for details.

**Example:**

```
OUTALL1,4,1      ; Turn on on-board outputs 1-4
2OUTALL3,8,1    ; On I/O brick 2, turn on outputs at I/O locations 3-8
                ; (I/O pins 3-8 on SIM1)
```

---

## OUTBD Brake Output Delay

Type	Output	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>OUTBD<i>	GT6K	6.0
Units	milliseconds	GV6K	6.0
Range	0 - 1000		
Default	0		
Response	OUTBD: *OUTBD250		
See Also	DRIVE, OUTFNC		

---

The OUTBD command specifies the amount of time that the fault output (OUTFNCn-F) will remain asserted after current is applied to the motor windings with the DRIVE1 command. This allows torque to build up in the motor while the fault output is still high. This is important in vertical applications, where the motor must be able to support the load before the brake is released.

---

## OUTEN Output Enable

Type	Output or Program Debug Tool	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!><@><B>OUTEN<d><d><d><d>... (one <d> for each input)	GT6K	6.0
Units	n/a	GV6K	6.0
Range	d = 0 (Disable output function and turn output off) d = 1 (Disable output function and turn output on) d = E (Enable output function) d = X (don't change)		
Default	E		
Response	OUTEN: *OUTENEEEE_EEE (onboard outputs) 1OUTEN: *1OUTENEEEE_EEEE_EEEE_EEEE_EEEE_EEEE_EEEE 1OUTEN.3 *E		
See Also	OUT, OUTFNC, OUTLVL, TIO, TOUT, TSTAT		

---

Use the OUTEN command 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.

The OUTEN command has no effect on onboard outputs when they are configured as output-on-position outputs with the OUTFNCi-H command.

The number of external outputs varies by configuration of I/O bricks used. Refer to page 8 for details.

**Example:**

```
; This allows the user to test if the fault output is working,
; without the inconvenience of trying to force a fault.
1OUTFNC1-B      ; Define output #1 on I/O brick 1 as axis moving/not moving
1OUTFNC2-A      ; Define output #2 on I/O brick 1 as programmable
1OUTFNC3-A      ; Define output #3 on I/O brick 1 as programmable
1OUTFNC4-A      ; Define output #4 on I/O brick 1 as programmable
1OUTFNC5-F      ; Define output #5 on I/O brick 1 as fault output
1OUTENxxxx1    ; Disable programmed function of output #5 on I/O brick 1
                ; and turn it on
```

## OUTFNC Output Function

Type	Output	Product	Rev
Syntax	<a_><!><B>OUTFNC<i>-<c>	GT6K	6.0
Units	i = output #, c = function identifier (letter)	GV6K	6.0
Range	i = 1-32 (I/O brick dependent – see page 8) c = A-H		
Default	(see table below)		
Response	OUTFNC: (function and status of onboard outputs) 1OUTFNC: (function and status of outputs on I/O brick 1) 1OUTFNC1: *1OUTFNC1-A PROGRAMMABLE OUTPUT - STATUS OFF		
See Also	OUT, OUTEN, OUTLVL, OUTP, OUTPLC, OUTTW, POUT, SMPER, TIO, TSTAT		

**Default Function Assignments:** outputs on the DRIVE I/O connector, as well as the relay output (labeled “RELAY COM” and “RELAY N.O.”) on the 4-pin removable connector.

Output #	Pin #	GT6K & GV6K Function (OUTFNC default)
1	41	OUTFNC1-A (General-purpose)
2	43	OUTFNC2-A (General-purpose)
3	45	OUTFNC3-A (General-purpose)
4	46	OUTFNC4-A (General-purpose)
5	48	OUTFNC5-A (General-purpose)
6	49	OUTFNC6-A (General-purpose)
7	Relay	OUTFNC7-F (Fault)

Use the OUTFNC command to define the function for each output. A limit of 32 outputs may be assigned OUTFNC functions; this excludes A (“general-purpose”) function.

**Output bit assignments vary by product.** The number of external outputs varies by configuration of I/O bricks used. Refer to page 8 for details.

**Output Scan Rate:** The programmable outputs are scanned once per *system update* (2 milliseconds).

**Multitasking.** If the OUTFNC command does not include the task identifier (%) prefix, the function affects the task that executes the OUTFNC command. Only function “C” may be directed to a specific task with the % prefix (e.g., 2%OUTFNC3-C assigns onboard output 3 as a program-in-progress output for task 2). Multiple tasks may share the same output, but the output may only be assigned one function.

Identifier	Function Description
A	<b>General-Purpose Output:</b> A <i>General Purpose</i> output is a standard output. Turn on or off with the OUT, POUTA, 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	<b>Moving/Not Moving:</b> Output activates when the axis is moving. As soon as the move is completed, the output will change to the opposite state.  Servo Axes: 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. In this manner, the output functions as an <i>In Position</i> output.
C	<b>Program in Progress:</b> Output activates when a program is being executed. After the program is finished, the output’s state is reversed.
D	<b>End-of-Travel Limit Encountered:</b> 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 LH0 command, but this is recommended only if the motor is not coupled to the load.)

- E **Stall Indicator** (GT6K only): Output activates when a stall is detected. To detect an encoder stall, you must first connect an encoder and enable stall detection with the `ESTALL1` command. To detect a drive stall (no encoder required), you must enable drive stall detection with the `DSTALL1` command. For details refer to the *Programmer's Guide*.
- F **Fault Indicator**: Output activates when one or more of these fault conditions exist:
  - A *User 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 was shut down (`DRIVE0`) or the Enable input was opened (reported with `TAS` bit #13), but only if the “fault on drive disable” function is enabled with `FLTDSB1`.
  - Drive fault(s) occurred (`TAS` bit 14 is set).
  - Other drive and motor fault conditions – refer to the status bits denoted with an asterisk (\*) in the `TAS` and `TASX` descriptions.
- G **Position Error Exceeds Max. Limit** (GV6K 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 drive is shut down and `TER` (error status) bit #12 and `TAS` (axis status) bit #23 are set. If the `SMPER` command is set to zero (`SMPER0`), the position error will not be monitored; thus, the *Position Error* output function will not be usable.
- H **Output On Position**: Output activates when the axis is at a specified position (servo axes can use encoder position only; stepper axes can use commanded position or encoder position, depending on the `ENCCNT` setting for that axis). Output On Position function parameters are configured with the `OUTPA` command. Applicable only to onboard output #1 (pin #41 on the “DRIVE I/O” connector).

**Example:**

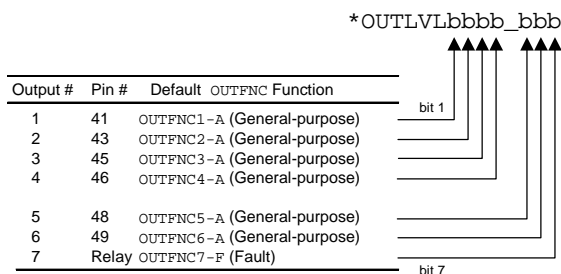
```
1OUTFNC1-3B      ; Define output #1 on I/O brick 1 as moving/not moving output
1OUTFNC2-D      ; Define output #2 on I/O brick 1 to go active when any of
                ; the limits are hit
```

## OUTLVL Output Active Level

Type	Outputs	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!><@><B>OUTLVL<b><b><b>...<b><b><b>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	b = 0 (active low), 1 (active high) or X (don't change)		
Default	Onboard outputs: 0 Expansion I/O outputs: 0 for NPN outputs, 1 for PNP outputs		
Response	OUTLVL: *OUTLVL0000_000 (onboard outputs) 1OUTLVL: *1OUTLVL0000_0000_0000_0000_0000_0000_0000_0000 1OUTLVL.3 *0		
See Also	KIOEN, OUT, OUTEN, OUTFNC, OUTP, OUTPLC, OUTTW, POUT, TOUT		

The `OUTLVL` command defines the active state of each programmable output. The factory default state is active low for the digital outputs on the DRIVE I/O connector, and normally open for the relay output (labeled “RELAY COM” and “RELAY N.O.”) on the 4-pin removable connector.

`OUTLVL` bit assignments (bits are numbered 1-7 from left to right):



**NOTE:** The OUTLVL setting is NOT saved in battery-backed RAM; therefore, on power up or reset, the OUTLVL setting will default to the factory default setting (thus, the OUTLVL command is a good candidate for inclusion in your STARTP program).

The number of external outputs varies by the configuration of I/O bricks used. Refer to page 8 for details.

## Onboard Outputs

Relationships:	OUTLVL Setting	OUT State *	Current	TOUT status
	OUTLVL0 (default)	OUT1	Sinking current	1
	OUTLVL0	OUT0	No current flow	0
	OUTLVL1	OUT1	No current flow	1
	OUTLVL1	OUT0	Sinking current	0

\* The output is "active" when it is commanded by the OUT command (for example, OUTxx1 activates output #3).

## Outputs on Expansion I/O Bricks:

- **Sinking vs. Sourcing Outputs.** On power up, the Gem6K drive auto-detects the type of output SIM installed on each external I/O brick, and automatically changes the OUTLVL setting accordingly. If sinking (NPN) outputs or Reed Relay outputs are detected, OUTLVL is set to active low (OUTLVL0); if sourcing (PNP) outputs are detected, OUTLVL is set to active high (OUTLVL1).
- **Disconnect I/O Brick.** If the I/O brick is disconnected (or if it loses power), the drive will perform a kill (all tasks) and set error bit #18. (If you disable the "Kill on I/O Disconnect" mode with KIOENØ, the Gem6K will not perform the kill.) The drive will remember the brick configuration (volatile memory) in effect at the time the disconnection occurred. When you reconnect the I/O brick, the drive checks to see if anything changed (SIM by SIM) from the state when it was disconnected. If an existing SIM slot is changed (different SIM, vacant SIM slot, or jumper setting), the drive will set the SIM to factory default INEN and OUTLVL settings. If a new SIM is installed where there was none before, the new SIM is auto-configured to factory defaults.

Relationships:	Output Type	OUTLVL Setting	OUT State *	Current	OUT/TOUT status	LED
	NPN (sinking)	OUTLVL0 (default)	OUT1	Sinking current	1	ON
	NPN	OUTLVL0 (default)	OUT0	No current flow	0	OFF
	NPN	OUTLVL1	OUT1	No current flow	1	OFF
	NPN	OUTLVL1	OUT0	Sinking current	0	ON
	PNP (sourcing)	OUTLVL0	OUT1	No current flow	1	OFF
	PNP	OUTLVL0	OUT0	Sourcing current	0	ON
	PNP	OUTLVL1 (default)	OUT1	Sourcing current	1	ON
	PNP	OUTLVL1 (default)	OUT0	No current flow	0	OFF
	Reed Relay	OUTLVL0 (default)	OUT1	Sinking (closed)	1	ON
	Reed Relay	OUTLVL0 (default)	OUT0	No flow (open)	0	OFF
	Reed Relay	OUTLVL1	OUT1	No flow (open)	1	OFF
	Reed Relay	OUTLVL1	OUT0	Sinking (closed)	0	ON

\* The output is "active" when it is commanded by the OUT, OUTP, or POUT command (for example, OUTxx1 activates output #3).

### Example:

```
OUTLVL1x0 ; Configure onboard output 1 to be active high, output 2 unchanged,
; and output 3 as active low
```

## OUTPA Output on Position

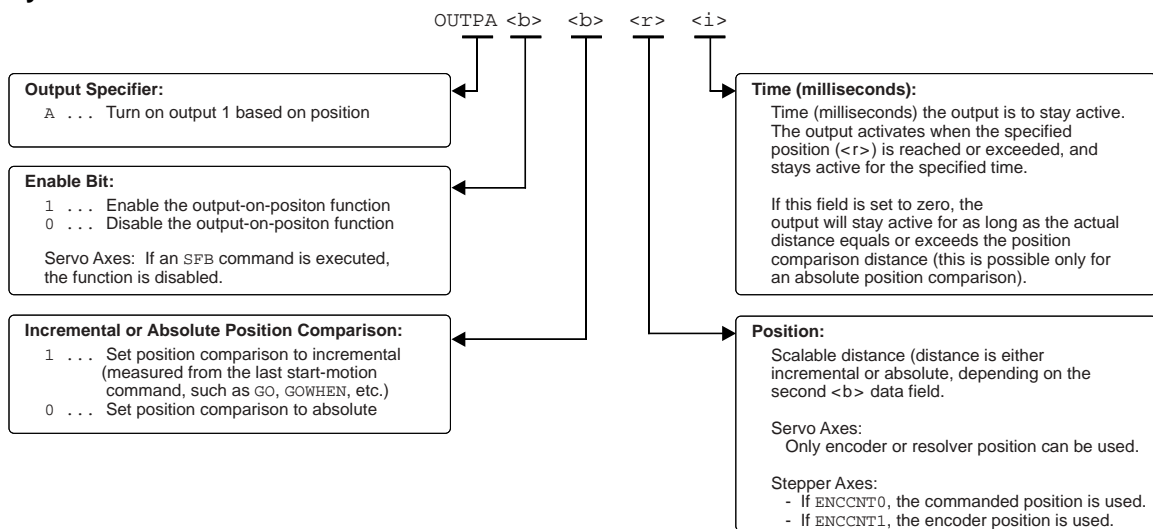
		Product	Rev
Type	Output	GT6K	6.0
Syntax	<!>OUTPA<b>, <b>, <r>, <i>	GV6K	6.0
Units	1 <sup>st</sup> & 2 <sup>nd</sup> b = enable/modal bits; r = scalable distance i = time (ms)		
Range	1 <sup>st</sup> b = 1 (enable output on position) or 0 (disable) 2 <sup>nd</sup> b = 1 (incremental position) or 0 (absolute position) r = -2,147,483,648 to +2,147,483,647 i = 0-65535		
Default	0,0,0,0		
Response	OUTPA: *OUTPA0,0,+0,0		
See Also	ENCCNT, [ OUT ], OUT, OUTFNC, PSET, SFB		

Use the OUTPA command to configure onboard output #1 (pin #41 on the DRIVE I/O connector) to activate based on the specified position of the axis. The position referenced is dependent upon whether the axis is a servo or stepper:

- **GV6K Servo:** The referenced position is the feedback position.
- **GT6K Stepper:** The referenced position depends on the ENCCNT setting at the time the OUTPA command is executed. If ENCCNT0 (factory default), the commanded position is used, if ENCCNT1, the encoder position is used.

To use the OUTPA command, you must first use the OUTFNC1-H command to configure onboard output #1 to function as an *output on position* output. Refer to the programming example below.

### Syntax:



### NOTE

The output activates only during motion; therefore, 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.

### Example (servo axes):

```
SFB1           ; Select encoder feedback
OUTFNC1-H     ; Set onboard output #1 as an "output on position" output
OUTPA1,0,+50000,50 ; Turn on onboard output #1 for 50 ms when the encoder
                  ; position is > or = absolute position +50,000
```

---

## OUTPLC Establish PLC Strobe Outputs

Type	Output	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>OUTPLC<i>,<i-i>,<i>,<i>	GT6K	6.0
Units	See below	GV6K	6.0
Range	See below		
Default	1,0-0,0,0		
Response	OUTPLC1: *0-0,0,0 (onboard outputs referenced)		
See Also	INPLC, OUT, OUTEN, OUTFNC, OUTLVL, OUTTW, [ TW ]		

---

The OUTPLC command with its corresponding INPLC command configures 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>	<b>Set #:</b> There are 4 possible OUTPLC sets (1-4). This field identifies which set to use.
Field 2: <i-i>	<b>Strobe Output #s:</b> 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. Refer to page 8 for help in identifying which output bits are available to place in this field.
Field 3: <i>	<b>TW Command Pending:</b> 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>	<b>Strobe Time:</b> 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,0-0,0,0 where n is the PLC set (1-4).

### Example:

```
1INPLC2,1-8,9,10 ; Set INPLC set 2 as BCD digits on inputs 1-8 on I/O brick 1,
                  ; with input 9 as the sign bit, and input 10 as the data valid bit
OUTPLC2,1-4,5,50 ; Set OUTPLC set 2 as output strobes on onboard outputs 1-4,
                  ; with output 5 as the command pending bit, and strobe time
                  ; of 50 milliseconds
A(TW6)           ; Read data into acceleration using INPLC set 2
                  ; and OUTPLC set 2 as the data configuration
```

---

## OUTTW Establish Thumbwheel Strobe Outputs

Type	Output	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!><B>OUTTW<i>,<i-i>,<i>,<i>	GT6K	6.0
Units	See below	GV6K	6.0
Range	See below		
Default	1,0-0,0,0		
Response	OUTTW1: *0-0,0,0 (onboard outputs referenced)		
See Also	INSTW, OUT, OUTEN, OUTFNC, OUTLVL, OUTPLC, [ TW ]		

---

The OUTTW command with its corresponding INSTW command configure the applicable inputs and outputs to read data from an active thumbwheel device. 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>	<b>Set #:</b> There are 4 possible OUTTW sets (1-4). This field identifies which set to use.
Field 2: <i-i>	<b>Strobe Output #s:</b> 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. Refer to page 8 for help in identifying which output bits are available to place in this field.
Field 3: <i>	<b>Thumbwheel Enable Output:</b> 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 thumbwheel module to respond, thus allowing multiple thumbwheels to be wired to the inputs and outputs. A zero in this field will not activate any output.
Field 4: <i>	<b>Strobe Time:</b> 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:**

```
INSTW2,1-4,5      ; Set INSTW set 2 as BCD digits on onboard inputs 1-4, with
                  ; input 5 as the sign bit
OUTTW2,1-3,4,50   ; Set OUTTW set 2 as output strobes on onboard outputs 1-3,
                  ; with onboard output 4 as the output enable bit, and
                  ; strobe time of 50 milliseconds
A(TW2)            ; Read data into acceleration using INSTW set 2 and
                  ; OUTTW set 2 as the data configuration
```

[ PC ]		Position Commanded	
Type	Assignment or Comparison	Product	Rev
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	ERES, [ FB ], GOWHEN, [ PCC ], [ PE ], [ PER ], PSET, SCALE, SCLD, SMPER, TAS, TFB, TPC, TPCC, TPE, TPER		

Use the PC operator to assign the current *commanded position* (scalable by SCLD) to a variable, or to make a comparison against another value. If you issue a PSET command, the commanded position value will be offset by the PSET command value.

**Servo Axes:** The PC value is measured in encoder or resolver counts. The commanded position (PC) and the actual position (FB) are used in the control algorithm to calculate the position error (PC - FB = PER) and thereby determine the corrective control signal.

**Stepper Axes:** The PC value is measured in commanded counts (“motor counts”).

**UNITS OF MEASURE and SCALING:** refer to page 16.

**Syntax:** VARn=PC where “n” is the variable number; or PC can be used in an expression such as IF(PC>50).

**Example:**

```
VAR1=PC          ; Commanded position is assigned to variable 1
IF(PC<50)        ; If the commanded position is <50, do the IF statement
VAR2=PC+500     ; Commanded position plus 500 is assigned to variable 2
NIF              ; End IF statement
```

## [ PCC ]      Captured Commanded Position

Type	Assignment or Comparison	Product	Rev
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	ENCCNT, INFNC, [ PC ], [ PCMS ], PSET, SCALE, SCLD, SFB, [ TRIG ], TRGLOT, TPC, TPCC, TTRIG		

Use the PCC operator to assign the captured commanded position to a variable, or to make a comparison against another value.

**Syntax:** VARn=PCCc where “n” is the variable number, and “c” designates trigger A or B, or M for the MASTER TRIG input; or PCC can be used in an expression such as IF(PCCB>23450).

General-purpose inputs on the DRIVE I/O connector:

Input #	Pin #	GT6K & GV6K Function (INFNC default)
1	37	INFNC1-A (General-purpose. If assigned, TRG-A must be on Input #1)
2	38	INFNC2-A (General-purpose. If assigned, TRG-B must be on Input #2)
3	39	INFNC3-A (General-purpose)
4	34	INFNC4-A (General-purpose)
5	35	INFNC5-A (General-purpose. If assigned, MASTER TRIGGER must be on Input #5)

**About Position Capture:** The commanded position can be captured only by an input that is defined as a “trigger interrupt” input with the INFNCi-H command (see INFNC for details). When a “trigger interrupt” input is activated, the commanded position is captured and the position is available through the use of the PCC operator and the TPCC display command.

**Note for Stepper Axes:** By default, stepper axes capture only the commanded position. However, if the axis has Encoder Capture Mode enabled with the ENCCNT command, only the encoder position is captured.

**Position Capture Status, Longevity of Captured Position:** Use the TTRIG and TRIG commands to ascertain if a trigger interrupt input has been activated. TTRIG displays the status as a binary report, and TRIG is an assignment/comparison operator for using the status information in a conditional expression (e.g., in an IF statement). Once the captured commanded position value is assigned/compared with the PCC operator, the TTRIG/TRIG status bit for that trigger input is cleared; but the position information remains available until it is overwritten by a subsequent position capture from the same trigger input.

**Position Capture Accuracy:** The commanded position capture accuracy is  $\pm 1$  count.

**Scaling and Position Offset:** 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. If you issue a PSET (establish absolute position reference) command, any previously captured commanded positions will be offset by the PSET command value.

**Example:**

```
INFNC1-H      ; Assign input 1 as TRIG-A input
VAR1=PCCA    ; Assign captured commanded position to variable 1
              ; (position was captured when trigger input TRIG-A became active)
IF(PCCA<40)  ; If the captured commanded position is less than 40, do the
              ; IF statement
VAR2=PCCA+10 ; Add 10 to the captured commanded position (captured when
              ; trigger input TRIG-A became active) and assign the sum to
              ; variable #2
NIF          ; End IF statement
```

## [ PCE ] Position of Captured Encoder

Type	Assignment or Comparison	Product	Rev
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	ENCCNT, INFNC, [ PCMS ], [ PE ], PSET, SCALE, SCLD, SFB, TPCE, [ TRIG ], TRGLOT, TTRIG		

Use the PCE operator to assign the captured encoder position to a variable, or to make a comparison against another value.

**Syntax:** VARn=PCEc where “n” is the variable number, and “c” designates trigger A or B, or M for the **MASTER TRIG** input; or PCE can be used in an expression such as IF (PCEB>23450).

Inputs on the DRIVE I/O connector:

Input #	Pin #	GT6K & GV6K Function (INFNC default)
1	37	INFNC1-A (General-purpose. If assigned, TRG-A must be on Input #1)
2	38	INFNC2-A (General-purpose. If assigned, TRG-B must be on Input #2)
3	39	INFNC3-A (General-purpose)
4	34	INFNC4-A (General-purpose)
5	35	INFNC5-A (General-purpose. If assigned, MASTER TRIGGER must be on Input #5)

**About Position Capture:** The encoder position can be captured only by an input that is defined as a “trigger interrupt” input with the INFNCi-H command (see INFNC command). When a “trigger interrupt” input is activated, the encoder position is captured and the position is available through the use of the PCE operator and the TPCE display command. **Stepper Axes:** By default, stepper axes capture only the commanded position. To capture the encoder position, the axis must be in the Encoder Capture Mode (see ENCCNT command).

**Position Capture Status, Longevity of Captured Position:** Use the TTRIG and TRIG commands to ascertain if a trigger interrupt input has been activated. TTRIG displays the status as a binary report, and TRIG is an assignment/comparison operator for using the status information in a conditional expression (e.g., in an IF statement). Once the captured encoder position value is assigned/compared with the PCE operator, the TTRIG/TRIG status bit for that trigger input is cleared; but the position information remains available until it is overwritten by a subsequent position capture from the same trigger input.

**Position Capture Accuracy:** The encoder position capture accuracy is  $\pm 1$  encoder count.

**Scaling and Position Offset:** If scaling is enabled (SCALE1), the encoder position is scaled by the distance scaling factor (SCLD). If scaling is not enabled (SCALE0), the value assigned will be actual encoder counts. (NOTE: When using a stepper axis operating in ENCCNT1 mode, the PCE value is always encoder counts and is never scaled by SCLD.) If you issue a PSET (establish absolute position reference) command, any previously captured encoder positions will be offset by the PSET command value.

**Example:**

```

INFNC1-H           ; Assign trigger input TRIG-A as trigger interrupt input
VAR1=PCEA         ; Assign captured encoder position to variable 1
                  ; (position was captured when trigger input TRIG-A became active)
IF(PCEA<4000)     ; If the captured encoder position is less than 4000,
                  ; do the IF statement
VAR2=PCEA+10      ; Add 10 to the captured encoder position
                  ; (captured when trigger input TRIG-A became active) and
                  ; assign the sum to variable #2
NIF               ; End IF statement

```

## [ PCME ] Position of Captured Master Encoder

Type	Assignment or Comparison	Product	Rev
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	INFNC, MEPOL, MESND, [ PME ], [ PCMS ], PMECLR, PMESET, TPCME, TPME, TPCMS		

Use the PCME operator to assign the captured master encoder position to a variable, or to make a comparison against another value. The master encoder is connected to the connector labeled “Master Encoder.”

**Syntax:** VARn=PCME where “n” is the variable number; or PCME can be used in an expression such as IF(PCME>2345Ø).

Inputs on the DRIVE I/O connector:

Input #	Pin #	GT6K & GV6K Function (INFNC default)
1	37	INFNC1-A (General-purpose. If assigned, TRG-A must be on Input #1)
2	38	INFNC2-A (General-purpose. If assigned, TRG-B must be on Input #2)
3	39	INFNC3-A (General-purpose)
4	34	INFNC4-A (General-purpose)
5	35	INFNC5-A (General-purpose. If assigned, MASTER TRIGGER must be on Input #5)

**About Position Capture:** The master encoder position can be captured only by the Master Trigger input (input #5), and only when that input is defined as a “trigger interrupt” input with the INFNC5-H command (see INFNC command). When the “trigger interrupt” input is activated (active edge), the master encoder position is captured and the position is available through the use of the PCME operator and the TPCME display command.

**Position Capture Status, Longevity of Captured Position:** Use the TTRIG and TRIG commands to ascertain if a trigger interrupt input has been activated. TTRIG displays the status as a binary report, and TRIG is an assignment/comparison operator for using the status information in a conditional expression (e.g., in an IF statement). Once the captured master encoder position value is assigned/compared with the PCME operator, TTRIG/TRIG status bit #5 is cleared; but the position information remains available until it is overwritten by a subsequent position capture from the master trigger input.

**Position Capture Accuracy:** The master encoder position capture accuracy is  $\pm 1$  encoder count.

**Scaling and Position Offset:** The PCME value is always in master encoder counts; it is never scaled. If you issue a PMESET (establish absolute position reference) command, any previously captured master encoder positions will be offset by the PMESET command value.

### Example:

```
INFNC5-H      ; Assign input #5 as master trigger input (TRIG-M) for the
              ; master encoder
VAR1=PCME     ; Assign captured master encoder position to variable 1
              ; (position was captured when master trigger became active)
IF(PCME<4000) ; If the captured master encoder position
              ; (captured when master trigger input became active) is
              ; less than 4000, do the IF statement
VAR2=PCME+10  ; Add 10 to the captured master encoder position
              ; (captured when master trigger input became active) and
              ; assign the sum to variable #2
NIF          ; End IF statement
```

## [ PCMS ]      Captured Master Cycle Position

Type	Assignment or Comparison	<b>Product</b>	<b>Rev</b>
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	ENCCNT, FOLMAS, INFNC, [ PCC ], [ PCE ], [ PE ], PSET, SCALE, SCLMAS, SFB, TPCMS, [ TRIG ], TRGLOT, TTRIG		

Use the PCMS operator to assign the captured master cycle position for a follower axis to a variable, or to make a comparison against another value.

PCMS (like PMAS) is unique among position assignment variables, because its value rolls over to zero each time the entire master cycle length (FMCLLEN) has been traveled. Thus, the captured PCMS value is essentially a snap-shot of the position relative to the master cycle at the time of the capture.

**The master must be assigned first (FOLMAS command) before this operator will be useful.**

**Syntax:** VARn=PCMSc where “n” is the variable number, and “c” designates trigger A or B for the axis, or M for the MASTER TRIG input (see table below); or PCMS can be used in an expression such as IF(PCMSB>2311).

General-purpose inputs on the DRIVE I/O connector:

Input #	Pin #	GT6K & GV6K Function (INFNC default)
1	37	INFNC1-A (General-purpose. If assigned, TRG-A must be on Input #1)
2	38	INFNC2-A (General-purpose. If assigned, TRG-B must be on Input #2)
3	39	INFNC3-A (General-purpose)
4	34	INFNC4-A (General-purpose)
5	35	INFNC5-A (General-purpose. If assigned, MASTER TRIGGER must be on Input #5)

**About Position Capture:** The master cycle position can be captured only by an input that is defined as “trigger interrupt” input with the INFNCi-H command (see INFNC command). When a “trigger interrupt” input is activated, the master cycle position of the axis is captured and the position is available through the use of the PCMS operator and the TPCMS display command.

**Position Capture Status, Longevity of Captured Position:** Use the TTRIG and TRIG commands to ascertain if a trigger interrupt input has been activated. TTRIG displays the status as a binary report, and TRIG is an assignment/comparison operator for using the status information in a conditional expression (e.g., in an IF statement). Once the captured master cycle position value is assigned/compared with the PCMS operator, the TTRIG/TRIG status bit for that trigger input is cleared; but the position information remains available until it is overwritten by a subsequent position capture from the same trigger input.

**Position Capture Accuracy:** The master cycle position is interpolated; the capture accuracy is 50 µs multiplied by the velocity of the axis at the time the trigger input was activated.

**Scaling and Position Offset:** If scaling is enabled (SCALE1), the master source position is scaled by the distance scaling factor (SCLMAS). If scaling is not enabled (SCALE0), the value assigned will be actual counts from the commanded or encoder master source as selected with the FOLMAS command. If you issue a PSET (establish absolute position reference) command, any previously captured master cycle positions will be offset by the PSET command value.

---

## PCOMP Compile a Profile or Program

Type	Compiled Motion; PLC Program	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>PCOMP<t>	GT6K	6.0
Units	t = text (name of program/path)	GV6K	6.0
Range	Text name of 6 characters or less		
Default	n/a		
Response	n/a		
See Also	DEF, DRES, END, GOBUF, GOWHEN, MEMORY, PEXE, PLOOP, PLCP, PLN, POUTn, PRUN, SCLD, PUCOMP, SCANP, [ SEG ], [ SS ], TDIR, TMEM, TRGFN, TSEG, TSS		

---

Use the PCOMP command to compile GOBUF profiles compile PLCP programs for PLC Scan Mode. (For additional detail on compiled motion, refer to the Custom Profiling chapter in the *Programmer's Guide*.)

### “Programs” vs. “Compiled Profiles & Programs”:

- Programs are defined with the DEF and END commands, as demonstrated in the Program Development Scenario in the *Programmer's Guide*.
- Compiled Profiles are defined like programs (using the DEF and END commands), but are compiled with the PCOMP command and executed with the PRUN command. A compiled profile could be a series of GOBUF commands.
- Compiled PLC programs are defined with DEF PLCPi and END, compiled with PCOMP, and are normally executed in the PLC Scan Mode with the SCANP.

### Compiling and Storing Compiled Programs:

The Gem6K's memory has two partitions: one for storing programs (“program” memory) and one for storing profiles & program segments compiled with the PCOMP command (“compiled” memory). The allocation of memory to these two areas is controlled with the MEMORY command.

Programs intended to be compiled are stored in program memory. After they are compiled with the PCOMP command, they remain in program memory and the segments (see segment command list below) from the compiled profile are stored in compiled memory.

- Compiled Motion segments: GOBUF, PLOOP, GOWHEN, TRGFN, POUTA
- PLC Program segments: IF, ELSE, NIF, L, LN, OUT, EXE, PEXE, VARI, VARB

The TDIR command uses “COMPILED AS A PATH” to denote the programs compiled as a compiled profile, and “COMPILED AS A PLC PROGRAM” to denote the programs compiled as a PLC programs. TDIR also reports the amount of program storage available, as does the TSEG command. System status bit #29 indicates that compiled memory is 75% full, and system status bit #30 indicates that compiled memory is completely full. (Use TSSF, TSS and SS to work with system status bits.)

If a compile (PCOMP) fails, system status bit #31 (see TSSF, TSS and SS) will be set. This status bit is cleared on power-up, reset, or after a successful compile. Possible causes for a failed compile are:

- Errors in profile design (e.g., change direction while at non-zero velocity; distance and velocity equate to < 1 count/system update; preset move profile ends in non-zero velocity).
- Profile will cause a Following error (see TFSF, TFS and [FS] commands).
- Out of memory (see system status bit #30).
- Axis already in motion at the time of a PCOMP command.
- Loop programming errors (e.g., no matching PLOOP or PLN; more than four embedded PLOOP/END loops).
- PLCP program contains invalid commands or command parameters.

### Conditions That Require a Re-Compile (Compiled Motion only):

- If it is desired to change a compiled path's velocity, acceleration, or deceleration, the values must be changed and then the path must be re-compiled.
- If the scaling factors are changed, the program must be downloaded again.
- Compiled Motion ONLY: After compiling (PCOMP) and running (PRUN) a compiled profile, the profile segments will be deleted from compiled memory if you cycle power or issue a RESET command.

## COMPILED MOTION

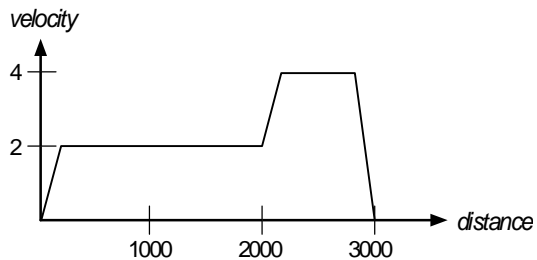
When using compiled loops (PLOOP and PLN), the last segment within the loop must end at zero velocity or there must be a final GOBUF segment placed outside the loop. Otherwise an error will result when the profile is compiled. The error is "ERROR: MOTION ENDS IN NON-ZERO VELOCITY".

**PLC PROGRAM EXAMPLE:** see PLCP command description.

**COMPILED MOTION EXAMPLE** (see profile below)

```

DEF prog2          ; Begin definition of program named prog2
A10                ; Set A, V, and D values
V2
D2000
GOBUF1             ; First segment of motion
V4                ; New A, V, and D values
AD50
D1000
GOBUF1             ; Second segment
END                ; End definition of prog2
PCOMP prog2        ; Compile prog2
PRUN prog2         ; Execute prog2
    
```



### [ PE ]

#### Position of Encoder

Type	Assignment or Comparison	Product	Rev
Syntax	See below	GT6K	6.0
Units	Encoder counts, or scaled by SCLD	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	ENCCNT, [ FB ], GOWHEN, INFNC, [ PC ], [ PCE ], [ PER ], PESET, PSET, SCALE, SCLD, SFB, TFB, TPE		

The PE operator is used to assign one of the encoder register values to a variable, or to make a comparison against another value.

**Stepper axes:** If the ENCCNT1 mode is enabled PE reports the encoder position, but in ENCCNT0 mode (the factory default setting) the PE report represents the commanded position. **NOTE:** When operating in ENCCNT1 mode, the PE value is always encoder counts and is never scaled by SCLD.

**UNITS OF MEASURE and SCALING:** refer to page 16 or to the SCLD command.

If you issue a PSET command, the encoder position value will be offset by the PSET command value. If you are using a stepper axis in the ENCCNT1 mode, use the PESET command instead.

**Syntax:** VARn=PE where "n" is the variable number, or PE can be used in an expression such as IF(PE>23450).

**Example:**

```

VAR1=PE           ; Encoder position is assigned to variable 1
IF(PE<4000)       ; If the encoder position is less than 4000,
                  ; do the IF statement
VAR2=PE+4000      ; Encoder position plus 4000 is assigned
                  ; to variable 2
    
```

NIF ; End IF statement

---

## [ PER ] Position Error

Type	Assignment or Comparison	Product	Rev
Syntax	See below	GT6K	n/a
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		(applicable only to servo axes)
Response	n/a		
See Also	DRES, ERES, SCLD, SFB, SMPER, TAS, TPER, TPE, TPC		

---

The PER operator is used to assign the current position error 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.

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=PER where “n” is the variable number, or PER can be used in an expression such as IF(0PER>50).

**Example:**

```
VAR1=PER ; Position error is assigned to variable 1
IF(PER>2000) ; If the position error is >2000 encoder counts,
; do the IF statement (enable output #4)
OUTXXX1 ; Enable onboard output #4
NIF ; End IF statement
```

---

## PESET Encoder Absolute Position Reference - Stepper Axes

Type	Motion	Product	Rev
Syntax	<a_><!>PESET<r>	GT6K	6.0
Units	r = units (absolute position of encoder)	GV6K	n/a
Range	±999,999,999.99999		
Default	n/a		(applicable to stepper axes only)
Response	n/a		
See Also	ENCCNT, INFNC, [ PCE ], [ PE ], PASET, PSET, SCALE, SCLD, TPCE, TPE		

---

Use the PESET command to offset the current encoder absolute position to establish an *absolute position reference* for the encoder reports (TPE, PE, TPCE, PCE). All PESET values entered are in encoder steps, scalable by the SCLD value if scaling is enabled.

**NOTE:** If you issue a PESET command, any previously captured encoder positions (INFNCi-H or LIMFNCi-H function) will be offset by the PESET value.

**Example:**

```
ENCCNT1 ; Place axis in the encoder count referencing mode
PESET0 ; Set absolute position to zero
TPE ; Display the new position. The new encoder position
; report should be: *TPE0
PESET1000 ; Set absolute position to 1000 units
TPE ; Display the new position. The new encoder position
; report should be: *TPE1000
```



---

## PEXE Execute a Compiled Program

Type	PLC Mode Compiled Program Execution	<b>Product</b>	<b>Rev</b>
Syntax	i%PEXEt	GT6K	6.0
Units	i = Task Number t = Program Name (6 characters or less)	GV6K	6.0
Range	i = 1-10		
Default	n/a		
Response	n/a		
See Also	EXE, GOBUF, PCOMP, PLCP, SCANP		

---

Use the PEXE command to start a compiled PLCP program, or compiled GOBUF profile from within a compiled PLCP program. The PEXE command specifies the name of the compiled program, and the task in which it will be launched. The program named in the PEXE command need not be defined or compiled at the time the PLCP program is compiled; however, the program must be defined and compiled before the SCANP or PRUN is issued. If no task number is assigned with a % prefix, then the task in which the PLCP program is compiled (PCOMP) will be the task that runs the compiled program. Note, however, that the PEXE program cannot be executed in the Task Supervisor (task 0).

The PLCP program will ignore the PEXE command if a currently running program is detected within the specified task; therefore, the PEXE command can essentially only be used to initiate a new task with the program it is launching. Like the INSELP command, the program launched by the PEXE command will not interrupt a currently running program, nor will it interrupt a WAIT or T command. Also, if launching a GOBUF profile, the PEXE will not interrupt motion already in progress.

**CAUTION:** Using the SCANP command to run a PLCP program in Scan mode will cause the PLCP program to execute as often as every system update period (2 ms). A PEXE command used within a PLCP program running in Scan mode could therefore attempt to launch a program in the specified task as often as every 2 ms. This may not allow enough time for the program launched in the specified task by the PEXE command to complete before the same PEXE command is issued again. As stated, the PLCP program will ignore the PEXE command if a currently running program is detected or motion is in progress, so timing must be considered when launching programs with the PEXE command.

To execute a non-compiled program from within a compiled PLCP program, use the EXE command.

### Example:

```
DEF PLCP1          ; Define PLC program PLCP1
1%PEXE PLCP2      ; Launch compiled program PLCP2 in task 1
END
DEF PLCP2          ; Define PLC program PLCP2
OUT(VARB1)        ; Modify outputs
END
PCOMP PLCP1       ; Compile PLCP1
PCOMP PLCP2       ; Compile PLCP2
SCANP PLCP1       ; Scan with program PLCP1
VARB1=h0000       ; Set VARB1
TOUT               ; Check outputs (response is *TOUT0000_000)
VARB1=b1010       ; Reassign VARB1
TOUT               ; Check outputs again (response is *TOUT1010_000)
```

---

## [ PI ]

### PI ( $\pi$ )

Type	Operator (Trigonometric)	<b>Product</b>	<b>Rev</b>
Syntax	See examples below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	[ = ], [ + ], [ - ], [ * ], [ / ], [ & ], [   ], [ ^ ], [ ~ ], [ ATAN ], [ COS ], IF, [ SIN ], [ SQRT ], [ TAN ], VAR		

---

The (PI) command is assigned the value 3.14159265. There are  $2\pi$  radians in  $360^\circ$ . This command is useful for doing trigonometric functions in radian units (RADIAN command).

#### Example:

```
VAR1=PI          ; 3.14159265 is assigned to variable 1
VAR2=2 * PI      ; 2 pi is assigned to variable 2
```

---

## PLCP

### Compiled PLC Program

Type	PLC Scan Program	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>PLCPi	GT6K	6.0
Units	i = number of PLC program	GV6K	6.0
Range	1-99		
Default	n/a		
Response	n/a		
See Also	ANO, [ ANO ], BREAK, DEF, ELSE, EXE, HALT, IF, L, LN, MEMORY, NIF, OUT, PCOMP, PEEXE, PRUN, PUCOMP, [ SC ], [ SCAN ], SCANP, TIMST, TIMSTP, TSC, TSCAN, VARI, VARB		

---

PLCP is not a command; it is used to identify a PLCP program to be defined (e.g., DEF PLCP2), compiled (e.g., PCOMP PLCP2), and executed (e.g., SCANP PLCP2 or PRUN PLCP2). Up to 99 PLCP programs may be defined, identified as PLCP1, PLCP2, PLCP3, and so on. The purpose of PLCP programs is to facilitate fast I/O scanning. The process of creating and executing a PLCP program is:

1. Define the PLCP program (DEF PLCPi statement, followed by commands from the list below, followed by END). Only these commands are allowed in a PLCP program:
  - IF, ELSE, and NIF (conditional branching) — see note below for limitations
  - L and LN (loops)
  - HALT:
    - If the PLCP program is executed with SCANP, HALT will terminate the current PLCP program and kill the SCANP mode.
    - If the PLCP program is executed with PRUN, HALT will stop the PLCP program (and the program that executed the PRUN statement) running in that task.
  - BREAK:
    - If the PLCP program is executed with SCANP, BREAK will end only the current scan loop. At the next 2-millisecond update, the scan will restart at the first line of the PLCP program.
    - If the PLCP program is executed with PRUN, BREAK will stop the PLCP program and execution will continue in the program from which the PRUN was executed (resuming at the command immediately following the PRUN command).
  - TIMST and TIMSTP (start and stop the timer)
  - OUT (turn on a digital output)
  - ANO (set an analog output voltage – requires an extended I/O brick with on an analog output SIM)
  - EXE (execute a program in a specific task — e.g., 2%EXE MOVE)
  - PEEXE (execute a compiled program in a specific task — e.g., 3%PEEXE PLCP4)
  - VARI (integer variables).
  - VARB (binary variables). Bitwise operations are limited to Boolean And (&), Boolean Inclusive Or (|), and Boolean Exclusive Or (^).

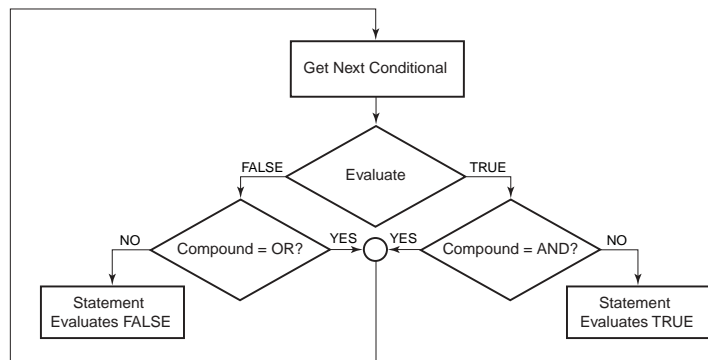
2. Compile the PLCP program (`PCOMP PLCPi`). A compiled program runs much faster than a standard program.
3. Execute the PLCP program (`SCANP PLCPi`). When the PLCP program is launched with the `SCANP` command, it is executed in the “PLC Scan Mode”. The advantage of the PLC Scan Mode is that the PLCP program is executed within a dedicated 0.5 ms time slot during every 2 ms system update period. This gives the PLCP program faster throughput for monitoring and manipulating I/O. *For more information on how the PLCP program is executed with SCANP, refer to the SCANP command description.*

An alternative execution method is to use the `PRUN` command (`PRUN PLCPi`). This method is similar to the `SCANP PLCPi` method, but will only run through the PLCP program once.

**Memory Requirements:** Most commands allowed in a PLCP program consume one segment of compiled memory after the program is compiled with `PCOMP`; the exceptions are `VARI` and `VARB` (each consume 2 segments) and `IF` statements. Each `IF` conditional evaluation compounded with either an `AND` or an `OR` operator consumes an additional segment (e.g., `IF (IN.1=b1 AND AS.1=b0)` consumes three segments of compiled memory). The number of compounds is limited only by the memory available.

**Conditional Expressions:**

- **Order of Evaluation.** Because only one level of parenthesis is allowed, the order of evaluation of `IF` conditionals is from left to right. Refer to the flowchart for the evaluation logic.
- Conditional expressions in a PLC program use the non-scaled integer (“raw”) operand values. Examples of the “raw” operand values are:



- The `PE` operator reports encoder counts not scaled by `SCLD` and not scaled by `ERES`.
- The `ANI` operator reports ADC counts from an analog input. Assuming the default `ANIRNG4` setting (+/-10V voltage range), 205 ADC counts = 1 volt.
- The `ANO` operator reports DAC counts commanded to an analog output. The range is 0-4095 counts over the -10 to +10V range. The calculation is:  $ANO\ value = (V_{out} + 10) \times 4095 / 20$ .

The only operands that are not allowed are: `SIN`, `COS`, `TAN`, `ATAN`, `VCVT`, `SQRT`, `VAR`, `TW`, `READ`, `DREAD`, `DREADF`, `DAT`, `DPTR`, and `PI`.

**Programming Example:** Refer to the example in the `SCANP` command description.

PLN	End of Loop, Compiled Motion	Product	Rev
Type	Compiled Motion	GT6K	6.0
Syntax	<a_>PLN	GV6K	6.0
Units	n/a		
Range	n/a		
Default	n/a		
Response	No response; instead ends loop for compiled motion		
See Also	GOBUF, PCOMP, PLOOP, PRUN, PUCOMP		

Use the `PLN` command to specify the end of a compiled motion profile loop, as initiated with the `PLOOP` command.

Programming Example: see `PLOOP`.

---

## **PLOOP**      **Beginning of Loop, Compiled Motion**

Type	Compiled Motion	<b>Product</b>	<b>Rev</b>
Syntax	<a_>PLOOP<i>	GT6K	6.0
Units	i = number of times to loop	GV6K	6.0
Range	0-2,147,483,647 ( $2^{31}-1$ ) 0 = infinite loop		
Default	n/a		
Response	No response; instead starts loop for compiled motion		
See Also	GOBUF, PCOMP, PLN, PRUN, PUCOMP		

---

The PLOOP command specifies the beginning of a profile loop. All subsequent segments defined between the PLOOP and PLN commands are included within that loop. The PLOOP value specifies the number of loops to be executed. If that number is a zero or blank, then the loop will be executed infinitely. The PLOOP command can be nested up to four levels deep within a program.

When using compiled loops (PLOOP and PLN), the last segment within the loop must end at zero velocity or there must be a final GOBUF segment placed outside (after) the loop. Otherwise, an error will result when the profile is compiled. The error is "ERROR: MOTION ENDS IN NON-ZERO VELOCITY".

The PLOOP command will consume one segment of compiled space.

### **Example:**

```
DEF prog1          ; Begin definition of prog1
V1                 ; Set velocity to 1 unit/sec
D1000              ; Set distance to 1000 units
GOBUF1             ; Segment of motion sent to buffer

PLOOP3             ; Start loop of the subsequent move profile
V10                ; Set velocity to 10 units/sec
D25000             ; Set distance to 25000 units
GOBUF1             ; First segment within loop sent to buffer

V2                 ; Set velocity to 2 units/sec
D1000              ; Set distance to 1000 units
GOBUF1             ; Second segment of motion within loop sent to buffer

V1                 ; Set velocity to 1 unit/sec
D25000             ; Set distance to 25000 units
GOBUF1             ; Third segment within loop sent to buffer

PLN1               ; Close loop

V.5                ; Set velocity to 0.5 units/sec
D100               ; Set distance to 100 units
GOBUF1             ; Segment of motion sent to buffer (outside loop)

END                ; End definition of prog1

PCOMP prog1        ; Compile prog1
PRUN prog1         ; Execute prog1
```

---

## **[ PMAS ]**      **Current Master Cycle Position**

Type	Following and Assignment or Comparison	<b>Product</b>	<b>Rev</b>
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	FMCNEW, FMCP, FOLMAS, FOLMD, [ FS ], GOWHEN, [ PCMS ], SCALE, SCLMAS, TPMAS, TFS		

---

The PMAS operator is used to assign the master position register value to a variable, or to make a comparison against another value. This value may be used for subsequent decision making, or for recording the cycle position corresponding to some other event.

PMAS is unique among position assignment variables, because its value rolls over to zero each time the entire master cycle length (FMCLLEN value) has been traveled. If it is desired to WAIT or GOWHEN on a master cycle position of the next master cycle, one master cycle length (value of FMCLLEN) should be added to the master cycle position specified in the argument. This allows commands that sequence follower events through a master cycle to be placed in a loop. The WAIT or GOWHEN command at the top of the loop could execute, even though the actual master travel had not finished the previous cycle. This is done to allow a PMAS value which is equal to the master cycle length to be specified and reliably detected. When using PMAS with IF, UNTIL, or WHILE arguments, the instantaneous PMAS value is used. Be careful to avoid specifying PMAS values that are nearly equal to the master cycle length (FMCLLEN), because rollover may occur before a PMAS sample is read.

**The master must be assigned first (FOLMAS command) before this command will be useful.**

If scaling is enabled (SCALE1), the PMAS value is scaled by the master scaling factor (SCLMAS). If scaling is disabled (SCALE0), the PMAS value is in counts.

**Syntax:** VARn=PMAS where “n” is the variable number; or PMAS can be used in an expression such as IF (PMAS>23450).

**Example:** (refer also to FOLEN example #2)

```
IF(PMAS>4.3)      ; If the master has traveled more than 4.3 master user units
                  ; then do the IF statement
OUT.2=b1          ; Set onboard output #2 to 1
NIF               ; End of IF statement
VAR14=PMAS        ; Set VAR14 to the master cycle position
```

---

## [ PME ]

### Position of Master Encoder

		Product	Rev
Type	Assignment or Comparison		
Syntax	See below	GT6K	6.0
Units	Master Encoder counts	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	MEPOL, MESND, [ PCME ], [ PE ], PMECLR, PMESET, TPCME, TPME		

---

Use the PME operator to assign the current master encoder position to a variable, or to make a comparison against another value. The master encoder is connected to the connector labeled “Master Encoder”. If you issue a PMESET command, the encoder position value will be offset by the PMESET command value. The PME value is always in encoder counts, it is never scaled.

**Syntax:** VARn=PME where “n” is the variable number, or PME can be used in an expression such as IF (PME>16000).

**Example:**

```
VAR1=PME          ; Master encoder position is assigned to variable 1
IF(PME<4000)     ; If the master encoder count is less than 4000,
                  ; do the IF statement
VAR2=PME+4000    ; Master encoder position plus 4000 is assigned to variable 2
NIF              ; End IF statement
```

---

## PMECLR Clear Master Encoder Absolute Position Reference

Type	Motion	Product	Rev
Syntax	<a_><!>PMECLR<r>	GT6K	6.0
Units	r = master encoder counts (not scalable)	GV6K	6.0
Range	±999,999,999.99999		
Default	n/a		
Response	n/a		
See Also	MEPOL, MESND, [ PCME ], [ PME ], PASET, PSET, TPCME, TPME		

---

Use the PMECLR command to remove any offset on the master encoder position reports (offset to master encoder position is established with the PASET command).

### Example:

```
TPME           ; Report master encoder position. For the sake of this example,
               ; let's say the response is *TPME10000 (master encoder is at
               ; absolute position 10,000).
PASET20000    ; Change relative position of master encoder with offset
TPME           ; Report new master encoder position with offset. New position
               ; response should now be: *TPME20000 (what was considered
               ; position 10,000 is now considered position 20,000).
PMECLR        ; Clear any offset applied to the master encoder position
TPME           ; Report master encoder position with no offsets.
               ; Response should be: *TPME10000.
```

---

## PASET Establish Master Encoder Absolute Position Reference

Type	Motion	Product	Rev
Syntax	<a_><!>PASET<r>	GT6K	6.0
Units	r = master encoder counts (not scalable)	GV6K	6.0
Range	±999,999,999.99999		
Default	n/a		
Response	n/a		
See Also	MEPOL, MESND, [ PCME ], [ PME ], PMECLR, PSET, TPCME, TPME		

---

Use the PASET command to offset the current absolute position of the master encoder (connected to the connector labeled “Master Encoder”) to establish an *absolute position reference*. To remove the offset, issue the PMECLR command.

All PASET values entered are in master encoder counts; this value is never scaled.

### Example:

```
TPME           ; Report master encoder position. For the sake of this example,
               ; let's say the response is *TPME10000 (master encoder is at
               ; absolute position 10,000).
PASET20000    ; Change relative position of master encoder with offset
TPME           ; Report new master encoder position with offset. New position
               ; response should now be: *TPME20000 (what was considered
               ; position 10,000 is now considered position 20,000).
PMECLR        ; Clear any offset applied to the master encoder position
TPME           ; Report master encoder position with no offsets.
               ; Response should be: *TPME10000.
```

---

## PORT

### Designate Destination Communication (“COM”) Port

		Product	Rev
Type	Communication Interface	GT6K	6.0
Syntax	<a_><!>PORT<i>	GV6K	6.0
Units	i = port number		
Range	0 (set to active port), 1 (COM1), or 2 (COM2) NOTE: “COM1” is the “RS-232/485” or “ETHERNET” connector. “COM2” is the “RS-232” connector.		
Default	1		
Response	PORT: *PORT1		
See Also	], [, BOT, DRPCHK, E, EOL, EOT, ERRDEF, ERRLVL, ERROK, ERBAD, LOCK, [READ], WRITE, XONOFF		

---

The PORT command is used to determine which COM port is affected by the DRPCHK, E, ECHO, BOT, EOL, EOT, ERROK, ERBAD, ERRDEF, ERRLVL, and XONOFF commands. It also specifies the port to which responses and prompts from stored programs should be sent.

The PORT command also selects the target port through which the WRITE and READ commands transmit ASCII text strings. The DWRITE command (as well as all other RP240 commands) will affect the RP240 regardless of the PORT command setting. If no RP240 is detected, the commands are sent to the COM2 port. DWRITE text strings are always terminated with a carriage return.

You can use the PORT0 command to set the PORT value to the port through which the command was received. For example, if your computer is connected to the RS-232/485 port (COM1) and you send the PORT0 command, the PORT value will be changed to PORT1 and all subsequent communication commands will affect the COM1 port.

**Example** (The PORT command can be used to designate EOT parameters for both ports. Assume that port COM1 is being used to communicate to the Gem6K drive.)

```
PORT1          ; Select COM1 for EOT setup
EOT45,49,10    ; EOT for COM1 is -1<lf>
TPE            ; Send "Transfer Position of Encoder" response to COM1
               ; using EOT 45,49,10
PORT2          ; Select COM2 for EOT setup
EOT45,50,10    ; EOT for COM2 is -2<lf>
TPC            ; Send "Transfer Commanded Position" response to COM2
               ; using EOT 45,50,10
```

**Example** (The PORT command specifies both port setups and response destinations in a stored program.)

```
DEF qwe        ; Begin definition of qwe
PORT1
EOT45,49,10    ; EOT for COM1 is -1<lf>
TPE            ; Send "Transfer Position of Encoder" response to COM1
               ; using EOT 45,49,10
PORT2
EOT45,50,10    ; EOT for COM2 is -2<lf>
TPC            ; Send "Transfer Commanded Position" response to COM2
               ; using EOT 45,50,10
END            ; End definition of qwe
```

---

## POUTA

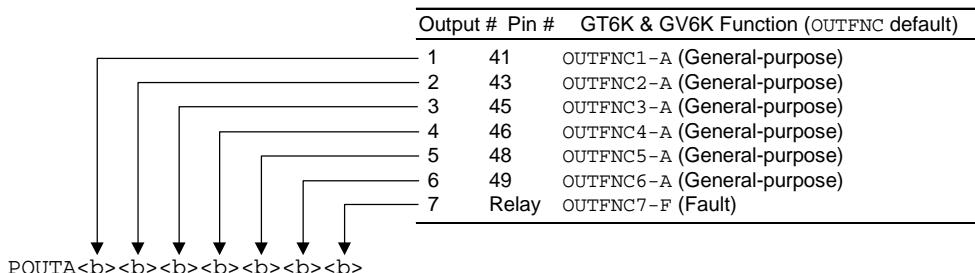
### Compiled Output

		Product	Rev
Type	Compiled Motion; Outputs	GT6K	6.0
Syntax	<a_><!><B>POUTA<b><b><b>...	GV6K	6.0
Units	b = enable bit for specific outputs (see page 8)		
Range	b = 0 (off), 1 (on), or X (don't change)		
Default	0		
Response	n/a		
See Also	GOBUF, OUT, OUTEN, OUTFNC, OUTLVL, PCOMP, PRUN, PUCOMP		

---

Use the POUTA command to control outputs during Compiled Motion. The outputs that can be controlled are the digital outputs on the DRIVE I/O connector, the relay output (labeled “RELAY COM” and “RELAY N.O.”) on the 4-pin removable connector, and any outputs on external I/O bricks. Refer to page 8 to

understand how to address the outputs (onboard and on optional expansion I/O bricks) available on the Gem6K.



**NOTE:** You may use POUTA to control only the outputs defined as “general-purpose” outputs with the OUTFNCi-A command. If you attempt to change the state of an output that is not defined as a general-purpose output, the drive will respond with an error message (“OUTPUT BIT USED AS OUTFNC”) and the POUTA command will not be executed (but command processing will continue).

If you wish to set only one output value, instead of all outputs, use the bit select (.) operator, followed by the number of the specific output. For example, 2POUTA.12-1 turns on only output 12 on I/O brick 2.

The POUTA command consumes one segment of compiled memory.

The programmable outputs are sampled once per “system update” (2 ms).

**Example:**

```

OUTFNC3-A      ; Default output function for onboard output 3
OUTFNC6-A      ; Default output function for onboard output 6
DEF P1         ; Define program P1
D1000          ; Set distance to travel
GOBUF1         ; Motion segment
POUTA.3-1      ; Turn on output 3 when the axis travels 1000 counts
D2000          ; New distance commanded
GOBUF1         ; Motion segment
POUTA.3-0      ; When the axis travels 2000 additional counts,
POUTA.6-1      ; turn off output 3 and turn on output 6
D1000          ; New distance commanded
GOBUF1         ; Motion segment
POUTA.6-0      ; Turn off output 6 when the axis travels 1000 additional counts
END            ; End program definition

PCOMP P1       ; Compiled program P1
PRUN P1        ; Execute program P1

```

When executing a Compiled Following profile, the POUTA statement is always executed as programmed. Therefore, in order to make sure an output is on for a given motion segment no matter what direction the master is traveling, you should use two POUTA statements (see example below).

```

POUTA.3-0      ; Turn off onboard output 3 - master going backwards
POUTA.3-1      ; Turn on onboard output 3 - master going forwards
GOBUF1         ; Motion segments for axis 1
POUTA.3-1      ; Turn on onboard output 3 for axis 1 - master going backwards
POUTA.3-0      ; Turn off onboard output 3 for axis 1 - master going forwards

```

If you desire to “pulse” an output (turn on for a given amount of time), then use the POUTA command along with the GOWHEN(T=n) command. For example:

```

POUTA.3-1      ; Turn on output 3
GOWHEN(T=120)  ; Wait for 120 milliseconds
POUTA.3-0      ; Turn off output 3

```



---

## PRUN

### Run a Compiled Profile

Type	Compiled Motion; PLC Program	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>PRUN<t>	GT6K	6.0
Units	t = text (name of path program)	GV6K	6.0
Range	text name of 6 characters or less		
Default	n/a		
Response	n/a		
See Also	COMEXC, DEF, END, PCOMP, PUCOMP, GOBUF, PLCP, PLOOP, PLN, SCANP		

---

Use the PRUN command to start execution of a previously compiled program (a GOBUF profile, or a PLCP program). All the required information about the program or path whose name is specified in the PRUN command has already been stored by the definition commands (DEF and END) and compiled by the PCOMP command.

Executing GOBUF profiles: If the axis is not ready, a GOBUF profile will not be executed. An axis is not ready if it is shutdown, moving, or in joystick mode. When execution of a pre-compiled program begins, the axis becomes busy until motion has completed.

Executing PLC programs: When using PRUN to execute a PLCP program, the PLCP program is run only once (as opposed to invoking a continual scan loop when executing PLCP programs with the SCANP command).

COMEXC1 mode must be enabled in order for command processing to continue once a motion invoking command has been initiated with PRUN. If you use the PRUN command within a program while in COMEXC1 mode, it functions as a GO and returns control back to the original program after the embedded program's motion is started (control is returned to the first command immediately following the PRUN command). If in COMEXC0 mode, command processing will not continue until the motion invoking command has completed its movement.

#### Example:

```
DEL prog1 ; Delete prog1
DEF prog1 ; Define prog1
MC0      ; Preset positioning mode
D50000   ; Distance is 50000
A10      ; Acceleration is 10 revs/sec/sec
AD10     ; Deceleration is 10 revs/sec/sec
V5       ; Velocity is 5 revs/sec
GOBUF1   ; 1st motion segment
D30000   ; Distance is 30000
V2       ; Velocity is 2 revs/sec
GOBUF1   ; 2nd motion segment
END      ; End definition of prog1
PCOMP prog1 ; Compile prog1
PRUN prog1 ; Execute prog1
```

---

## PS

### Pause Program Execution

Type	Program Flow Control	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>PS	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	C, COMEXR, COMEXS, K, S, [ SS ], TSS		

---

The 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:**

```

PS                ; Stop execution of command buffer until !C command
MA0               ; Select incremental positioning mode
D10000            ; Set distance to 10000 units
GO1               ; Initiate motion
; *****
; * NOTE:
; * No commands after the PS command will be executed until a !C
; * command is received.
; *****

```

---

**PSET**                      **Establish Absolute Position**

Type	Motion	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>PSET<r>	GT6K	6.0
Units	r = units (absolute position)	GV6K	6.0
Range	±999,999,999.99999		
Default	n/a		
Response	n/a		
See Also	D, [ FB ], GO, HOM, INFNC, MA, MC, [ PC ], [ PCC ], [ PCE ], [ PCMS ], [ PE ], PESET, PMESET, SCALE, SCLD, SFB, TFB, TPC, TPCC, TPCE, TPCMS, TPE		

---

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:** The PSET command will define the present commanded position to be the absolute position entered. To set an encoder absolute position, use the PESET command.

**Servos – without scaling:** The PSET command defines a new absolute position reference. If the drive is enabled (DRIVE1), the present commanded position is used as the reference point. If the drive is disabled (DRIVE0), the present feedback device position (selected with the SFB command) is used as the reference point.

<b>SERVO AXES</b>
-------------------

The PSET offset value is specific only to the feedback source selected with the last SFB command.

If your application requires switching between feedback sources, 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.)

---

**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:**

```

PSET0             ; Wherever the present actual or commanded position happens to be,
                  ; consider that position to have an absolute position of zero.

```

---

## [ PSHF ] Net Position Shift

Type	Following; Assignment or Comparison	<b>Product</b>	<b>Rev</b>
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	FOLEN, FOLMAS, FSHFC, FSHFD, SCALE, SCLD, TPSHF		

---

The PSHF operator is used to assign to a numeric variable the value of the net (absolute) follower axis position shift that has occurred since that last FOLEN1 command. The position value will be the sum of all shifts performed on that axis, including decelerations due to limits, kill, or stop. The shift value is set to zero each time a new FOLEN1 command or a FOLMAS command (with a value other than zero) is issued.

If scaling is enabled (SCALE1), the PSHF value is scaled by the distance scaling factor (SCLD). If scaling is not enabled, the value is in commanded counts.

**Syntax:** VARn=PSHF where “n” is the variable number, or PSHF can be used in an expression such as IF(PSHF>23450).

**Example:**

```
IF(PSHF>4.3)      ; If the follower axis has shifted more than 4.3 user units in the
                  ; positive direction, then do the IF statement
OUT.2=b1         ; Turn on onboard output #2
NIF              ; End of IF statement
VAR14=PSHF      ; Set VAR14 to the follower axis's position shift
```

---

## [ PSLV ] Current Commanded Position of Follower Axis

Type	Following; Assignment or Comparison	<b>Product</b>	<b>Rev</b>
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	FMCNEW, FMCP, SCLD, SCALE, TPSLV		

---

Use the PSLV operator to assign the follower axis commanded position register value to a variable, or to make a comparison against another value.

If scaling is enabled (SCALE1), the PSLV value is scaled by the distance scaling factor (SCLD). If scaling is not enabled, the value is in commanded counts.

**Syntax:** VARn=PSLV where “n” is the variable number, or PSLV can be used in an expression such as IF(PSLV>23450).

**Example:**

```
IF(PSLV>4.3)      ; If the follower axis has traveled more than 4.3 user units then
                  ; do the IF statement
OUT.2=b1         ; Turn on onboard output #2
NIF              ; End of IF statement
VAR14=PSLV      ; Set VAR14 to the follower axis's position
```

---

## PUCOMP Un-Compile a Compiled Profile

Type	Compiled Motion; PLC Program	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>PUCOMP<t>	GT6K	6.0
Units	t = text (name of program)	GV6K	6.0
Range	Text name of 6 characters or less		
Default	n/a		
Response	n/a		
See Also	DEF, END, GOBUF, MEMORY, PCOMP, PLCP, PRUN, SCANP, TDIR, TMEM, TSEG, GOBUF, PLOOP, PLN		

---

The PUCOMP command is used to delete a previously compiled (PCOMP) program from the compiled memory. The PUCOMP command does not delete the program from program memory.

### Example:

```
PUCOMP prog1      ; Delete compiled motion segments for prog1
DEL prog1         ; Delete prog1
DEF prog1         ; Begin definition of path named prog1
A50               ; Set acceleration to 50
V10               ; Set velocity to 10
D100000          ; Set D to 100000
GOBUF1
; *****
; * Add multiple GOBUF segment          *
; * definitions in this                 *
; * portion of the                      *
; * program                             *
; *****
END               ; End definition of prog1
PCOMP prog1      ; Compile prog1
PRUN prog1       ; Execute prog1
```

---

## RADIAN Radian Enable

Type	Operators (Trigonometric)	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>RADIAN<b>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	b = 0 (Disable), 1 (Enable) or X (don't care)		
Default	0		
Response	RADIAN: *RADIANO		
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:  $360^\circ = 2\pi$  radians.

### Example:

```
RADIAN1          ; Set trigonometric functions to radian mode
```

<b>RE</b>		<b>Registration Enable</b>	
Type	Registration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>RE<b>	GT6K	6.0
Units	b = 0 (disable), 1 (enable)	GV6K	6.0
Range	n/a		
Default	0		
Response	RE: *RE0		
See Also	[ AS ], COMEXC, [ ER ], INFNC, [ PCC ], [ PCE ], [ PCMS ], REG, REGLOD, REGSS, TAS, TER, TPCC, TPCE, TPCMS, TRGLOT, [ TRIG ], TTRIG		

The RE command enables the registration feature. When a registration input (an input assigned the “Trigger Interrupt” function with the INFNCi-H command) is activated, the motion profile currently being executed is replaced by a *registration profile* with its own distance (REG), acceleration (A & AA), deceleration (AD & ADA), and velocity (V) values. The registration move may interrupt any preset (MC0), continuous (MC1), or registration move in progress.

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 Gem6K is operating in the continuous command execution mode (COMEXC1).

Registration moves will not be executed while the motor is not performing a move, while in the joystick mode (JOY1), or while decelerating due to a stop, kill, soft limit, or hard limit.

### How to Set up a Registration Move

1. Use the INFNC1-H or INFNC2-H command to configure onboard input 1 or 2 as a trigger input (TRIG A or TRIG B, respectively). TRIG M, the MASTER TRIGGER, may not be used for registration.
2. Specify the distance of the registration move with the REG command. For servo axes, the distance refers to the encoder position. For stepper axes, the distance refers to commanded position.
3. Enable the registration function with the RE1 command. Registration is performed only when there is a non-zero distance specified by the REG command.

NOTE: The registration move is executed using the A, AA, AD, ADA, and V values that were in effect when the REG command was entered.

### Registration Move Accuracy (see also Registration Move Status below)

The accuracy of the registration move distance specified with the REG command is  $\pm 1$  count (servo axes: encoder count; stepper axes: commanded count)

RULE OF THUMB: To prevent position overshoot, make sure the REG distance is greater than 4 ms multiplied by the incoming velocity.

The lapse between activating the registration input and commencing the registration move (this does not affect the move accuracy) is less than one position sample period (2 ms).

The REG distance will be scaled by the distance scale factor (SCLD value) if scaling is enabled (SCALE1). See page 16 for details on scaling.

### Preventing Unwanted Registration Moves (methods)

- **Registration Input Debounce:** By default, the registration inputs are debounced for 24 ms before another input on the same trigger is recognized. (The debounce time is the time required between an input’s initial active transition and its secondary active transition.) If your application requires a shorter debounce time, you can change it with the TRGLOT command.
- **Registration Single-Shot:** The REGSS command allows you to program the Gem6K drive to ignore any registration commands after the first registration move has been initiated. Refer to the REGSS command description for further details and an application example.

- **Registration Lockout Distance:** The `REGLD` command specifies what distance an axis must travel before any input assigned as a registration input will be recognized. Refer to the `REGLD` command description for further details and an application example.

## Registration Move Status & Error Handling

**Axis Status — Bit #28:** This status bit is set when a registration move has been initiated by the registration input. This status bit is cleared with the next `GO` command.

`AS.28` ..... Assignment & comparison operator — use in a conditional expression (e.g., `WAIT(AS.28=b1)`).

`TASF` ..... Full text description of each status bit. (see “Reg Move Commanded” line item)

`TAS` ..... Binary report of each status bit (bits 1-32 from left to right). See bit #28.

**Axis Status — Bit #30:** If, when the registration input is activated, the registration move profile cannot be performed with the specified motion parameters, the Gem6K drive will kill the move in progress and set axis status bit #30. This status bit is cleared with the next `GO` command.

`AS.30` ..... Assignment & comparison operator — use in a conditional expression.

`TASF` ..... Full text description of each status bit. (see “Preset Move Overshot” line item)

`TAS` ..... Binary report of each status bit (bits 1-32 from left to right). See bit #30.

**Input Status — Bits #1-5:** Check the status of the assigned input to ascertain if it has been activated. The status reports the present state of the input.

`IN.n` ..... Comparison operand for `IF` and `WAIT` conditional expressions (e.g., `IF(IN.2=b1)`).

`TIN` ..... Binary report of each status bit (bits 1-5 from left to right).

**Error Status — Bit #10:** This status bit may be set if axis status bit #30 is set. The error status is monitored and reported only if you enable error-checking bit #10 with the `ERROR` command (e.g., `ERROR.10=1`). NOTE: When the error occurs, the Gem6K will branch to the error program (assigned with the `ERRORP` command). This status bit is cleared with the next `GO` command.

`ER.10` ..... Assignment & comparison operator — use in a conditional expression (e.g., `IF(ER.10=b1)`).

`TERF` ..... Full text description of each status bit. (see “Profile Impossible” line item)

`TER` ..... Binary report of each status bit (bits 1-32 from left to right). See bit #10.

**Trigger Status — Bits #1-5:** Trigger status bits are set when a registration move has been initiated by trigger inputs A or B. This also indicates that the position has been captured. As soon as the captured information is transferred or assigned/compared, the respective trigger status bit is cleared (set to  $\emptyset$ ).

`TRIG` ..... Assignment & comparison operator — use in a conditional expression.

`TTRIG` ..... Binary report of each status bit (bits 1-5). From left to right the bits represent trigger A and B (the 5<sup>th</sup> bit is master trigger M, the “MASTER TRIG” input terminal, which may not be used for registration) — see page 8.

## Example:

In this example, two-tiered registration is achieved. While the axis is executing its 50,000-unit move, Trigger 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 conveyor belt. After picking up the liquid and while registration move A is still in progress, Trigger B is activated and executes registration move B to slow the load to a gentle stop.

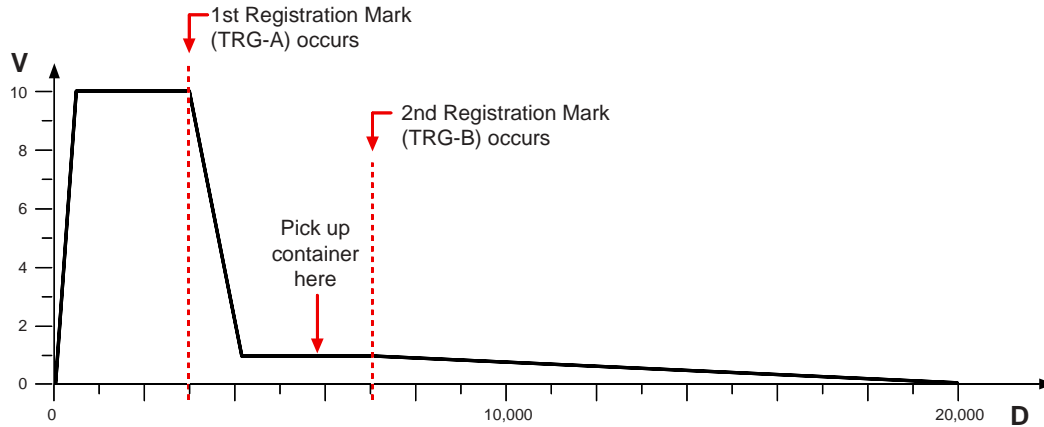
```
DEL REGI1      ; Delete program (assume program already resides in memory)
DEF REGI1      ; Begin program definition
INFNC1-H      ; Define input 1 as trigger A
INFNC2-H      ; Define input 2 as trigger B
A20           ; Set acceleration on to 20 units/sec/sec
AD40          ; Set deceleration on to 40 units/sec/sec
V1            ; Set velocity to 1 unit/sec
REGA4000      ; Set trigger A's registration distance to 4000 units
              ; (registration A move will use the A, AD, & V values above)
A5            ; Set acceleration to 5 units/sec/sec
AD2           ; Set deceleration to 2 units/sec/sec
V.5           ; Set velocity to 0.5 units/sec
REGB13000     ; Set trigger B's registration distance to 13,000 units
              ; (registration B move will use the A, AD, & V values above)
```

```

RE1          ; Enable registration
A50          ; Set acceleration to 50 units/sec/sec
AD50         ; Set deceleration to 50 units/sec/sec
V10         ; Set velocity to 10 unit/sec
D50000      ; Set distance to 50000 units
GO1         ; Initiate motion
END          ; End program definition

```

Registration Profile:



## [ READ ]

### Read a Value

Type	Communication Interface or Assignment	<b>Product</b>	<b>Rev</b>
Syntax	... READi ... (See below)	GT6K	6.0
Units	i = string variable number	GV6K	6.0
Range	1-50		
Default	n/a		
Response	n/a		
See Also	' , PORT, [ SS ], TSS, VAR, VARS, WRITE		

The READ command provides the user with an efficient way of storing numeric data read from the input buffer into a variable. The READ command can be used as part of a numeric variable assignment statement (e.g., VAR1=READ1) or in another command (A(READ1)). 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 sent out the Ethernet port or the RS-232 or RS-485 port, at the time this command is executed. The Gem6K drive 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>).

**Rule of Thumb for command value substitutions:** If the command syntax shows that the command field requires a real number (denoted by <r>) or an integer value (denoted by <i>), you can use the READ substitution (e.g., V(READ1)).

**Example:**

```

VARS1="Enter the count >" ; Place message in string variable #1
VAR2=READ1                ; Prompt with string variable #1, and read data
                           ; into variable #2
;

```

```

; The drive will send this message (string variable #1) to the screen:
;
;           "Enter the count >"
; The user must enter the numeric data preceded by the characters '!'.
; For example, '!82.5 assigns the value 82.5 to numeric variable 2

```

---

<b>REG</b>		<b>Registration Distance</b>	
Type	Registration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>REGc<r>	GT6K	6.0
Units	c = letter of trigger input (A or B) r = distance units (scalable by the SCLD value) servo axes: always encoder or resolver counts stepper axes: commanded counts	GV6K	6.0
Range	r = 0.00000 to 999,999,999 (after conversion to steps)		
Default	0 (do not make a registration move)		
Response	REG:       *REG0		
See Also	[ AS ], INFNC, [ ER ], [ PCC ], [ PCE ], [ PCMS ], RE, REGLOD, REGSS, SCALE, SCLD, [ SS ], TAS, TER, TPCE, TRGLOT, [ TRIG ], TTRIG		

---

The REG command specifies the distance to travel after receiving a registration input (trigger A or B). For example, REGA4000 sets up a 4000-count registration move to be initiated trigger A is activated.

Servo Axes: REG value always represents encoder or resolver counts.

Stepper Axes: REG value represents commanded counts.

**NOTE:** The registration move is executed using the A, AA, AD, ADA, and V values that were in effect when the REG command was entered. To see an example, refer to the programming sample in the RE command description.

**RULE OF THUMB:** To prevent position overshoot, make sure the REG distance is greater than 4 ms multiplied by the incoming velocity.

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 REG distance will be scaled by the distance scale factor (SCLD value) if scaling is enabled (SCALE1). See page 16 for details on scaling.

**UNITS OF MEASURE and SCALING:** refer to page 16.

**For additional details** on Registration (including programming examples), refer to the RE command description and to the Registration section in the *Programmer's Guide*.

---

<b>REGLOD</b>		<b>Registration Lock-Out Distance</b>	
Type	Registration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>REGLOD<r>	GT6K	6.0
Units	r = distance units (scalable by SCLD)	GV6K	6.0
Range	0.00000 to +999,999,999.99999		
Default	0		
Response	REGLOD:     *REGLOD0		
See Also	INFNC, RE, REG, REGSS, SCLD, TRGLOT		

---

The REGLOD command specifies the distance to travel before a registration input will be recognized. If scaling is enabled (SCALE1), the lock-out distance is scaled by the SCLD value.



**Stepper axes:** The lock-out distance is measured incrementally from the start of motion to the commanded position.

**Servo axes:** The lock-out distance is measured incrementally from the start of motion to the actual position (as measured by the position feedback device), not the commanded position.

**Example:**

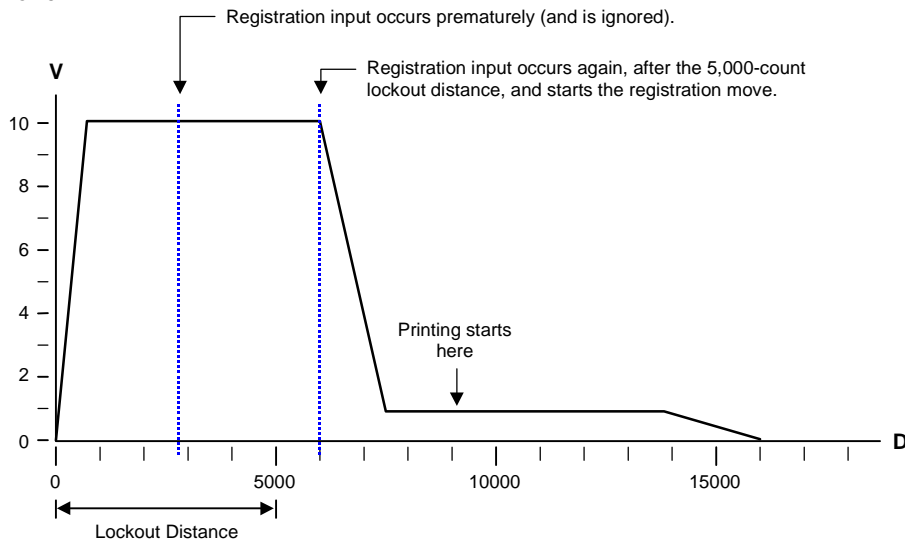
A print wheel uses registration to initiate each print cycle. From the beginning of motion, the Gem6K should ignore all registration marks before traveling 5000 counts.

```

DEL reg1      ; Delete program reg1
DEF reg1      ; Begin program definition
INFNC1-H     ; Define input #1 as a "trigger interrupt" input (TRG-A)
A40          ; Set acceleration to 40 revs/sec/sec
AD5          ; Set deceleration to 5 revs/sec/sec
V1           ; Set velocity to 1 rev/sec
REGA10000    ; Set registration distance to 10,000 counts
              ; (registration move will use the A, AD, & V values above)
REGL0D5000   ; Set registration lockout distance to 5000 counts
RE1          ; Enable registration
A50          ; Set acceleration to 50 revs/sec/sec
AD50         ; Set deceleration to 50 revs/sec/sec
V10          ; Set velocity to 10 revs/sec
D50000       ; Set distance to 50000 counts
GO1          ; Initiate motion
END           ; End program definition

```

Registration Profile:




---

**REGSS Registration Single-Shot**

Type	Registration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>REGSS<b>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	b=0 (Disable) or 1 (Enable)		
Default	0		
Response	REGSS: *REGSS0		
See Also	RE, REG, REGL0D		

---

The REGSS command sets the registration such that only one registration move will take place. This allows the user to prevent any other trigger from interrupting the registration move in progress. A GO command will reset the "one shot" condition.

**Example – Senario A:**

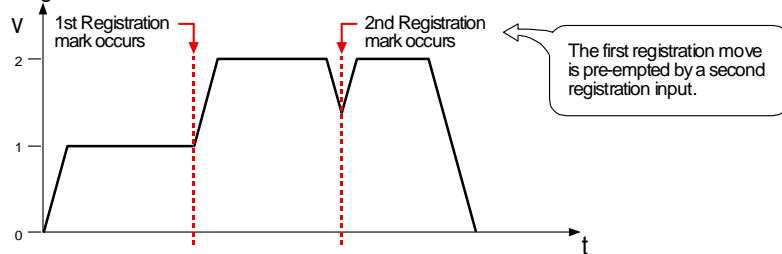
A user has a line of material with randomly spaced registration marks. It is known that the first mark must initiate a registration move, and that each registration move cannot be interrupted or the end product will be destroyed. Since the distance between marks is random, it is impossible to predict if a second registration mark will occur before the first registration move has finished.

```

DEL REGI2a      ; Delete program (in case program already resides in memory)
DEF REGI2a      ; Begin program definition
INFNC1-H       ; Define input #1 as a "trigger interrupt" input (TRG-A)
RE1            ; Enable registration
V2            ; Set registration move to a velocity of 2 rps
AD.5          ; a deceleration of 0.5 rev/sec/sec
REGA20000     ; and a distance of 20000 steps
MC1           ; Start a mode continuous
V1           ; move at a velocity of 1 rps
GO1          ; Initiate motion
END           ; End program definition

```

Registration Profile:



**Example – Scenario B** (introducing “single-shot” registration):

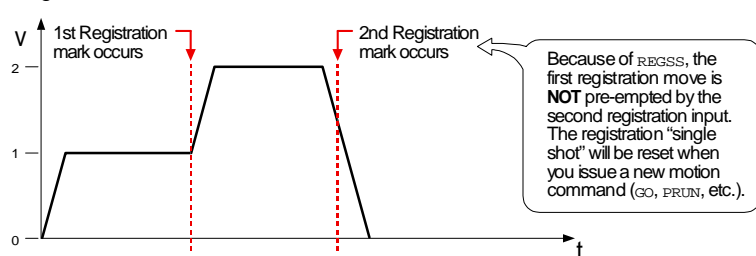
In order to stop the second registration from occurring, REGSS can be used:

```

DEL REGI2b      ; Delete program (in case program already resides in memory)
DEF REGI2b      ; Begin program definition
INFNC1-H       ; Define input #1 as a "trigger interrupt" input (TRG-A)
RE1            ; Enable registration
V2            ; Set registration move to a velocity of 2 rps
REGA20000     ; and a distance of 20000 steps
REGSS1        ; Enable registration single shot mode
MC1           ; Start a mode continuous
V1           ; move at a velocity of 1 rps
GO1          ; Initiate motion
END           ; End program definition

```

Registration Profile:



<b>REPEAT</b>		<b>Repeat Statement</b>	
Type	Program Flow Control or Conditional Branching	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>REPEAT	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	JUMP, UNTIL		

The 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, AS, D, ER, IN, INO, MOV, OUT, PC, PCE, PCME, PCMS, PE, PER, SS, TIM, US, V, VEL, etc.) can be used within the UNTIL( ) expression.

**Example:**

```

REPEAT          ; Beginning of REPEAT . . . UNTIL( ) loop
GOL            ; Initiate motion
VAR1=VAR1+1    ; Increment variable 1 by 1
UNTIL(VAR1=12) ; Repeat loop until variable 1 = 12

```

<b>RESET</b>		<b>Reset</b>	
Type	Communication Interface	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>RESET	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	RESET: (power-up message is displayed)		
See Also	DRESET, RFS, STARTP, TAS, TASX, TSTAT, TREV		

The RESET command affects the Gem6K drive the same as cycling power, or activating the hardware Reset input (pin 3 on the DRIVE I/O connector). The drive's programs, variables, scaling parameters, and motor configuration parameters are retained in non-volatile memory; however, all previously entered command values (not saved in programs or variables) will be reset to factory default values.

**NOTE:** After sending the RESET or !RESET command to the Gem6K, you must wait until you see the power-up message before communicating with the Gem6K.

**CAUTION:** The RESET command will disconnect an Ethernet connection. (See DRESET for a command that resets certain parameters, without disconnecting an Ethernet connection.)

---

<b>RFS</b>		<b>Return to Factory Settings</b>	
Type	Drive Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>RFS	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	ERROK, RESET, TDHRS, TREV		

---

The RFS command returns all settings to factory default, with the exception of the ERROK prompt and the TDHRS value. All stored programs and profiles are deleted. A RESET command is automatically issued following this command; therefore, no prompt will be returned.

**When is the RFS event finished?** The RFS process can take several seconds. When RFS is finished, the drive transmits the power-up message.

**Recommendation:** When you complete the drive configuration procedure in Motion Planner (see page 6), be sure to save the configuration file and program files to your PC's hard drive for safe keeping. If, after executing the RFS command, you need to restore the previous configuration or stored programs, re-download the configuration file and program files to your drive (REMEMBER: You must reset the drive to invoke new configuration settings).

---

<b>RUN</b>		<b>Begin Executing a Program</b>	
Type	Program or Subroutine Definition	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>RUN<t>	GT6K	6.0
Units	t = text (name of program)	GV6K	6.0
Range	Text name of 6 characters or less		
Default	n/a		
Response	n/a		
See Also	\$, DEF, DEL, END, GOSUB, GOTO		

---

The 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 similarly to a GOSUB command in that control returns to the original program when the called program finishes.

**Example:**

```
DEF pick          ; Begin definition of program named pick
GOL              ; Initiate motion
END              ; End program definition
RUN pick         ; Executes program named pick
pick            ; Executes program named pick
```

<b>S Stop Motion</b>		<b>Product</b>	<b>Rev</b>
Type	Motion	GT6K	6.0
Syntax	<a_><!>S<b>	GV6K	6.0
Units	n/a		
Range	b = 0 (do not stop) or 1 (stop)		
Default	1		
Response	!S: No response, instead motion is stopped.		
See Also	C, COMEXC, COMEXS, GO, K		

The Stop (S) command instructs the motor to stop motion. The S command will bring the axis to rest using the last deceleration values (AD and ADA) 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 must be enabled (COMEXC1). If the Stop (S) command is to be used external to the program, the immediate command identifier must be used (!S or !S1).

**Effect of COMEXS:**

COMEXS0: Under factory default conditions (COMEXS0), when the Gem6K receives a stop command (S, !S, S1, or !S1) or a stop input (input assigned a stop function with INFNCi-D), the following will happen:

- Motion decelerates to a stop, using the present AD and ADA deceleration values. The motion profile cannot be resumed.
- All commands in the Gem6K's command buffer are discarded.
- Program execution is terminated and cannot be resumed.

COMEXS1: Using the COMEXS1 mode, the Gem6K allows more flexibility in responding to stop conditions, depending on the stop method (see table below).

Stop Method	What Stops?		Resume Motion Profile. (Allow resume with a !C command or a resume input *)	Resume Program. (Allow resume with a !C command or a resume input *)	Save Command Buffer. (Save the commands that were in the command buffer when the stop was commanded)
	Motion	Program			
!S or S	Yes	Yes	Yes	Yes	Yes
!S1 or S1	Yes	No	No	No	Yes
Stop input	Yes	No	No	No	Yes
Pause input * (if COMEXR1)	Yes	Yes	Yes	Yes	Yes
Pause input * (if COMEXR0)	No	Yes	No	Yes	Yes

\* A Pause input is an input configured with the INFNCi-E command. This is also the Resume input that can be used to resume motion and program execution after a stop.

COMEXS2: Using the COMEXS2 mode, the Gem6K responds as it does in the COMEXS0 mode, with the exception that you can still use the program-select inputs to select programs (INSELP value is retained). The program-select input functions are: BCD select (INFNCi-B), and one-to-one select (INFNCi-P). For further details on program selection with inputs, refer to INFNC and INSELP.

**Example:**

```
GO1      ; Initiate motion
!S       ; Stop motion (must use "!" to stop motion in progress)
```

## [ SC ] Status Assignment/Comparison

Type	Assignment or Comparison	Product	Rev
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	SCANP, TSC, TSCF, VARB		

Use the SC operator to assign the drive 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, 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=SC where “n” is the binary variable number, or SC can be used in an expression such as IF(SC=b0010), or IF(SC=h7F). If it is desired to assign only one bit of the drive status value to a binary variable, instead of all 32, the bit select (.) operator can be used. For example, VARB1=SC.3 assigns drive status bit #3 to binary variable 1:  
 \*VARB1=XX0X\_XXXX\_XXXX\_XXXX\_XXXX\_XXXX\_XXX\_XXXX.

The function of each drive status bit is shown below.

Bit #	Function (1 = Yes; 0 = No)
1	RESERVED
2	RESERVED
3	A PLC program is being executed in Scan Mode (SCANP).
4	RESERVED
5	RESERVED
6	RESERVED
7	RESERVED
8-32	RESERVED

## SCALE Enable/Disable Scale Factors

Type	Scaling	Product	Rev
Syntax	<a_><!>SCALE<b>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	b = 0 (disable) or 1 (enable)		
Default	0		
Response	SCALE: *SCALE0		
See Also	DRES, ERES, SCLA, SCLD, SCLMAS, SCLV, SFB, TSTAT		

**NOTE:** All scaling settings (SCALE, SCLA, SCLD, SCLV and SCLMAS) are automatically saved in the Gem6K's battery-backed RAM.

Scaling allows you to program acceleration, deceleration, velocity, and position values in units of measure that are appropriate for your application. The SCALE command is used to enable or disable scaling (SCALE1 to enable, SCALE0 to disable). When scaling is enabled (SCALE1), all entered data is multiplied by the appropriate scale factor:

Type of Motion	Accel/Decel Scaling	Velocity Scaling	Distance Scaling
Standard Point-to-Point Motion	SCLA	SCLV	SCLD
Following	SCLA	SCLV	SCLD for follower distances SCLMAS for master distances

## When Should I Define Scaling Factors?

Scaling calculations are performed when a program is defined or downloaded. Consequently, you must enable scaling (SCALE1) and define the scaling factors (SCLD, SCLA, SCLV, SCLMAS) *prior* to defining (DEF), uploading (TPROG), or running (RUN or PRUN) the program.

**RECOMMENDATION:** Place the scaling commands at the beginning of your program file, *before* the location of any defined programs. This ensures that the motion parameters in subsequent programs in your program file are scaled correctly. When you use Motion Planner’s scaling setup wizard, the scaling commands are automatically placed in the appropriate location in your program file.

**ALTERNATIVE:** Scaling factors could be defined via a terminal emulator *just before* defining or downloading a program. Because scaling command values are saved in battery-backed RAM (remembered after you issue a RESET command or cycle power to the Gem6K product), all subsequent program definitions and downloads will be scaled correctly.

**RESTRICTIONS:** Scaling commands are not allowed in a program. If there are scaling commands in a program, the drive will report an error message (“COMMAND NOT ALLOWED IN PROGRAM”) when the program is downloaded. If you intend to upload a program with scaled motion parameters, be sure to use Motion Planner. Motion Planner automatically uploads the scaling parameters and places them at the beginning of the program file containing the uploaded program from the drive. This assures correct scaling when the program file is later downloaded.

**Units of Measure without Scaling** (Scaling is disabled (SCALEØ) as the factory default condition):

- Stepper axes: All distance values entered are in commanded counts (sometimes referred to as *motor steps*), and all acceleration, deceleration and velocity values entered are internally multiplied by the DRES command value.

Motion Attribute	Units of Measure
	Encoder
Accel/Decel	Revs/sec/sec *
Velocity	Revs/sec *
Distance	Counts **

\* All accel/decel & velocity values are internally multiplied by the ERES command value.

\*\* Distance is measured in the counts received from the feedback device.

**SCALING EXAMPLES:** Refer to page 16.

---

## [ SCAN ] PLC Scan Runtime

Type	PLC Scan Program; Assignment or Comparison	Product	Rev
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	PLCP, SCANP, TSCAN		

---

Use the SCAN operand to assign the PLCP runtime (the duration it took the last PLCP program scan to complete) to a variable, or to make a comparison against another value. A compiled PLCP program is launched into Scan mode using the SCANP command. During each 2 ms update, the PLCP program is scanned until 30 segments have been executed. If the PLCP program takes more than 30 segments to complete, the program will be paused and then resumed at the next 2 ms system update. The SCAN value is in multiples of the 2 ms system update period.

**Example:**

```
SCANP PLCP1 ; Start execution of compiled PLCP program 1 in Scan mode
VARI1=SCAN ; Assign the duration of the last scan to integer variable VARI1
```

## SCANP Scan Compiled PLCP Program

		Product	Rev
Type	PLC Scan Program	GT6K	6.0
Syntax	<a_><!>SCANP<t>	GV6K	6.0
Units	t = text (name of the PLCP program, or CLR)		
Range	t = PLCPi, where i is the number of the desired PLCP program, or t = CLR (to clear or stop the scan function)		
Default	n/a		
Response	n/a		
See Also	PLCP, PCOMP, PRUN, PUCOMP, [ SC ], [ SCAN ], TSC, TSCAN		

Use the SCANP command to initiate scanning a specific compiled PLCP program (PLCPi). For example, SCANP PLCP3 initiates scanning the program defined as PLCP3 (defined with DEF PLCP3) and compiled (PCOMP PLCP3).

The PLCP program is scanned/executed at the beginning of the 2 ms update period. The scan will stop after 30 segments have been executed and resume at the next 2 ms update. PLCP programs, when compiled, are comprised of a linked list of PLC segments. During a scan, each segment is counted until the total number of segments executed exceeds 30.

If the 30-segment limit occurs while executing a multi-segment statement, then that statement will finish no matter how many segments it executes. For example, if 29 segments have executed, then the next segment will cause the scan to pause until the next 2 ms update. If that next statement is VARI1=1PE, which executes 2 segments, then VARI1=1PE will complete its operations before pausing the scan.

To check how much time (in 2 ms increments) the last scan took to complete, issue the TSCAN command or use the SCAN operator. For example, if the last PLCP program took 3 system updates (2 ms each) to scan, then TSCAN would report \*TSCAN6, indicating that it took 6 ms to complete the scan. Each pass, if taking the same path through the conditional branches (IF statements), will always report the same TSCAN value.

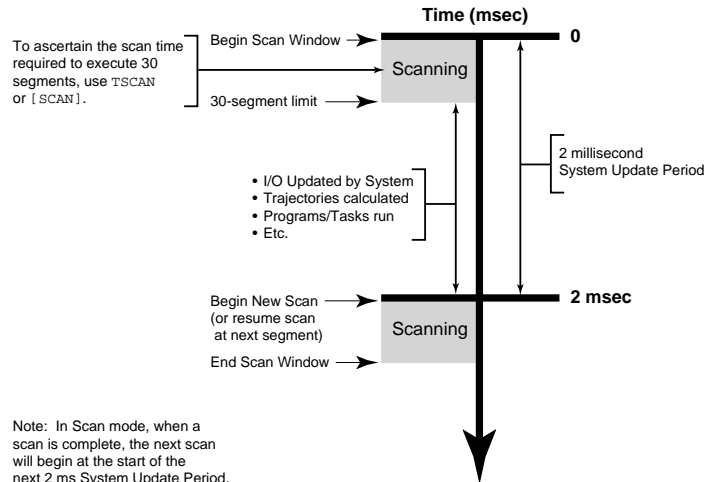
Controller Status bit #3 is set when a PLCP program is being executed in Scan Mode. To check the controller status, use SC, TSC, or TSCF.

**Launching programs external to the scan:** Using the EXE command or the PEXE command, a scan program can launch another program in a specified task. EXE launches a standard, non-compiled program; PEXE launches a compiled program.

**Stopping the scan:** The scan program can be stopped in several ways:

- Send the !K command to the Gem6K.
- Clear the scan program by issuing a SCANP CLR command.
- Placing a HALT or BREAK command in the PLCP program:
  - HALT terminates the current PLCP program scan and kills the SCANP mode.
  - BREAK will end only the current scan loop. At the next 2-millisecond update, the scan will restart at the first line of the PLCP program.

**Timing the PLCP program outputs:** It is not possible to control where the PLCP program will pause if the scan takes more than the allowed time. This means that there can be a time lag of several update periods before the outputs, analog outputs, and/or variables affected by the PLCP program are updated. The order





in which the scan takes place should be considered when creating PLCP programs to minimize the effects of such a lag. One way to avoid the lag is to create a binary variable as a temporary holding place for the desired output states. The last commands before the END statement of the PLCP program can set the outputs according to the final status of the variable, such that all output states are written at the same time, just as the scan completes. This method is demonstrated in the example program below.

For more information on defining a PLC program, refer to the PLCP command description.

**Example:**

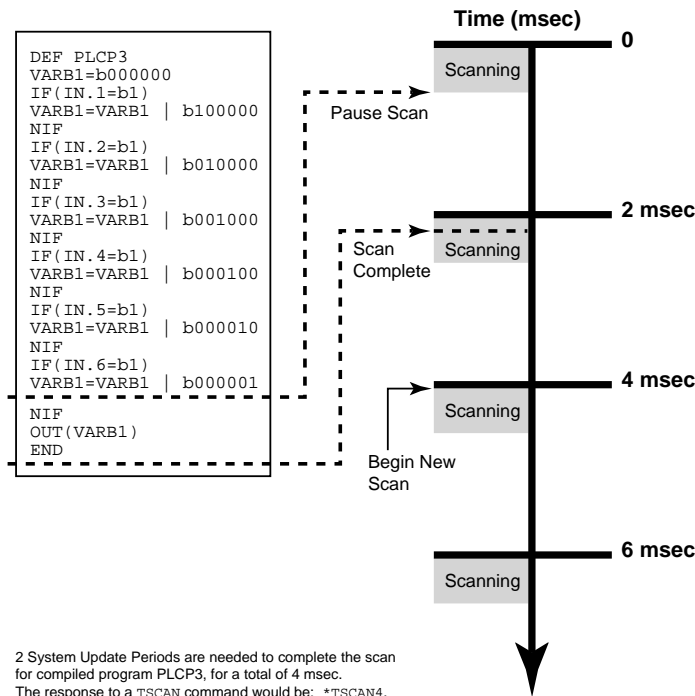
```

DEF PLCP3
; Binary states of outputs 1-6 are represented by VARB1 bit 1-6.
; Outputs 1-6 are set at the end of the program.
VARB1=b000000      ; Initialize binary variable 1
IF(IN.1=b1)        ; If Input 1 is ON, turn Output 1 ON
VARB1=VARB1 | b100000 ; Set binary bit for output 1 only to ON
NIF
IF(IN.2=b1)        ; If Input 2 is ON, turn Output 2 ON
VARB1=VARB1 | b010000 ; Set binary bit for output 2 only to ON
NIF
IF(IN.3=b1)        ; If Input 3 is ON, turn Output 3 ON
VARB1=VARB1 | b001000 ; Set binary bit for output 3 only to ON
NIF
IF(IN.4=b1)        ; If Input 4 is ON, turn Output 4 ON
VARB1=VARB1 | b000100 ; Set binary bit for output 4 only to ON
NIF
IF(IN.5=b1)        ; If Input 5 is ON, turn Output 5 ON
VARB1=VARB1 | b000010 ; Set binary bit for output 5 only to ON
NIF
IF(IN.6=b1)        ; If Input 6 is ON, turn Output 6 ON
VARB1=VARB1 | b000001 ; Set binary bit for output 6 only to ON
NIF
OUT(VARB1)         ; Turn on appropriate outputs
END

PCOMP PLCP3        ; Compile program PLCP3

SCANP PLCP3        ; Run compiled program PLCP3 in Scan mode
; The diagram below illustrates the scan.

```



## SCLA Acceleration Scale Factor

Type	Scaling	Product	Rev
Syntax	<a_><!>SCLA<i>	GT6K	6.0
Units	i = counts/unit	GV6K	6.0
Range	1 - 999,999		
Default	4000 (Servos auto-detect based on SFB: 4000 if encoder, 4096 if resolver)		
Response	SCLA: *SCLA4000		
See Also	ANIRNG, FMAXA, SCALE, SCLD, SCLMAS, SCLV, SFB, TSTAT		

**NOTE:** All scaling settings (SCALE, SCLA, SCLD, SCLV and SCLMAS) are automatically saved in the Gem6K's battery-backed RAM.

When scaling is enabled (SCALE1), all point-to-point acceleration values (A, AA, HOMA, HOMAA, JOGA, JOGAA, JOYA, JOYAA) and deceleration values (AD, ADA, LHAD, LHADA, LSAD, LSADA, HOMAD, HOMADA, JOGAD, JOGADA, JOYAD, JOYADA) are multiplied by the Acceleration Scale Factor (SCLA) command. Since the units are counts/unit, and all the acceleration values are in units/sec/sec, all accelerations will thus be internally represented as counts/sec/sec.

**Stepper axes:** 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/sec to commanded counts/sec/sec (sometimes referred to as “motor steps”/sec/sec).

If scaling is disabled (SCALE0), all accel and decel values are entered in commanded revs/sec/sec; these values are internally multiplied by the drive resolution (DRES) value to obtain accel and decel values in commanded counts/sec/sec for the motion trajectory calculations.

**Servo axes:** 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/sec to encoder counts/sec/sec.

If scaling is disabled (SCALE0), all accel and decel values are entered in encoder revs/sec/sec; encoder values are internally multiplied by the encoder resolution (ERES) value to obtain accel and decel values in counts/sec/sec 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 (counts/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.

Axis Type	Min. Accel or Decel (resolution)	Max. Accel or Decel
GT6K Stepper	$\frac{0.001 * DRES}{SCLA}$	$\frac{999.9999 * DRES}{SCLA}$
GV6K Servo	Encoder Feedback: $\frac{0.001 * ERES}{SCLA}$	Encoder Feedback: $\frac{999.9999 * ERES}{SCLA}$

### MORE ABOUT SCALING

For additional details on scaling, including scaling examples, refer to page 16.

## SCLD Distance Scale Factor

Type	Scaling	Product	Rev
Syntax	<a_><!*>SCLD<i>	GT6K	6.0
Units	i = counts/unit	GV6K	6.0
Range	1 - 999,999		
Default	1		
Response	SCLD: *SCLD1		
See Also	[ ANI ], D, [ FB ], FOLRN, FSHFD, [ PC ], [ PCC ], [ PCE ], [ PCMS ], [ PE ], [ PER ], PSET, [ PSHF ], [ PSLV ], REG, REGLOD, SCALE, SCLA, SCLV, SCLMAS, SFB, SMPER, TANI, TFB, TPC, TPCC, TPCE, TPCMS, TPE, TPER, TPSHF, TPSLV, TSTAT		

**NOTE:** All scaling settings (SCALE, SCLA, SCLD, SCLV and SCLMAS) are automatically saved in the Gem6K's battery-backed RAM.

If scaling is enabled (SCALE1), the Gem6K internally multiplies certain commands (see list below) by the Distance Scale Factor (SCLD) value.

Standard Motion	Following (Follower Axis only)
D	FOLRN
PESET	FSHFD
PSET	[ PCMS ]
REG	[ PSHF ]
REGLOD	[ PSLV ]
SMPER	TPCMS
STRGTD	TPSHF
[ ANI ]	TPSLV
[ FB ]	
[ PC ]	
[ PCC ]	
[ PCE ]	
[ PE ]	
[ PER ]	

Since the SCLD units are in terms of counts/unit, all distances will thus be internally represented in counts. For instance, if your distance scaling factor is 10000 (SCLD10000) and you enter a distance of 75 (D75), the actual distance moved will be 750,000 counts. (10000 x 75 = 750,000 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 SCLD250000, the D1.99999 command would be truncated to D1.9999).

SCLD (counts/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

### FRACTIONAL STEP TRUNCATION

If you are operating in the preset positioning mode (MC0), 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.

**More about scaling**, including scaling examples, refer to page 16.

## SCLMAS Master Scale Factor

Type	Following; Scaling	Product	Rev
Syntax	<a_><!>SCLMAS<i>	GT6K	6.0
Units	i = scaling factor	GV6K	6.0
Range	i = 1 - 999999		
Default	1		
Response	SCLMAS *SCLMAS1		
See Also	FMCLEN, FMCP, FOLEN, FOLMD, FOLRD, FOLRN, GOWHEN, [ PCMS ], [ PMAS ], SCALE, SCLD, TPMAS, TPCMS		

**NOTE:** All scaling settings (SCALE, SCLA, SCLD, SCLV and SCLMAS) are automatically saved in the Gem6K's battery-backed RAM.

The SCLMAS command internally multiplies all Following master values by the specified scale factor value. Since the SCLMAS units are in terms of counts/unit, all distances will thus be internally represented in counts. For instance, if your master scaling factor is 10000 (SCLMAS10000) and you enter a master parameter of 75 (e.g., FOLMD75), the internal value will be 750,000 counts. (10000 x 75 =750,000)

**NOTE:** The SCLMAS command will not take effect unless scaling is enabled (SCALE1).

This command allows you to specify distances in any unit. For example, if you had a 4000 step/revolution encoder as the master and wanted master units in terms of revolutions, then SCLMAS should be set to 4000.

As the master scaling factor (SCLMAS) changes, the resolution of all master parameter values and the number of positions to the right of the decimal point also change (see table below). A master parameter value with greater resolution than allowed will be truncated (e.g., if scaling is set to SCLD4000, the FOLMD1.9999 command would be truncated to FOLMD1.999).

SCLMAS (counts/unit)	Master Resolution (units)	Master 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

### FRACTIONAL STEP TRUNCATION

If you are specifying master distance values (FOLMD), when the master scaling factor (SCLMAS) and the distance value are multiplied, a fraction of one count may possibly be left over. This fraction is truncated when the distance value is used in the move algorithm. This truncation error can accumulate when performing several moves over the specified master distance. To eliminate this truncation problem, set the master scale factor (SCLMAS) to 1, or a multiple of 10.

**Example:** (refer also to the FOLEN examples, and page 16)

The commands below are a subset of the set-up parameters for an application in which the axis is following the Master encoder input at a 1-to-1 ratio.

```
SCALE1           ; Enable parameter scaling
SCLA25000        ; Set follower acceleration scale factor to 25000
SCLV25000        ; Set follower velocity scale factor to 25000
SCLD25000        ; Set follower distance scale factor to 25000
SCLMAS4000       ; Set master scale factor to 4000
FOLMAS1          ; Axis 1 using Master encoder input as master
FOLRN1           ; Set follower-to-master Following ratio numerator to 1
                 ; (scaled by SCLD)
FOLRD1           ; Set follower-to-master Following ratio denominator to 1.
                 ; This sets the ratio to 1:1 (scaled the SCLMAS).
                 ; The actual ratio in counts = 25000 to 4000 = 6.25 follower
                 ; axis counts per master count.
```

---

## SCLV

### Velocity Scale Factor

Type	Scaling	Product	Rev
Syntax	<a_><! >SCLV<i>	GT6K	6.0
Units	i = counts/unit	GV6K	6.0
Range	1 - 999,999		
Default	4000 (Servos auto-detect based on SFB: 4000 if encoder, 4096 if resolver)		
Response	SCLV: *SCLV4000		
See Also	ANIRNG, FMAXV, HOMV, HOMVF, JOGVH, JOGVL, JOYVH, JOYVL, SCALE, SCLA, SCLD, SFB, TSTAT, V		

---

**NOTE:** All scaling settings (SCALE, SCLA, SCLD, SCLV and SCLMAS) are automatically saved in the Gem6K's battery-backed RAM.

When scaling is enabled (SCALE1), all velocity values (HOMV, HOMVF, JOGVH, JOGVL, JOYVH, JOYVL, V) are multiplied by the Velocity Scale Factor (SCLV) command. Since the units are counts/unit, all velocities will thus be internally represented in counts/sec.

**Steppers:** If scaling is enabled (SCALE1), the entered velocity values are internally multiplied by SCLV to convert user units/sec to commanded counts/sec.

If scaling is disabled (SCALE0), all velocity values are entered in commanded revs/sec; these values are internally multiplied by the drive resolution (DRES) value to obtain velocity values in commanded counts/sec for the motion trajectory calculations.

**Servos:** If scaling is enabled (SCALE1), the entered velocity values are internally multiplied by SCLV to convert user units/sec to encoder counts/sec.

If scaling is disabled (SCALE0), all velocity values are entered in encoder revs/sec; encoder values are internally multiplied by the encoder resolution (ERES) value to obtain velocity values in counts/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

### MORE ABOUT SCALING

For additional details on scaling, including scaling examples, refer to page 16.

---

## [ SEG ]                      Number of Free Segment Buffers

Type	Compiled Motion; Assignment or Comparison	<b>Product</b>	<b>Rev</b>
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	MEMORY, [SS], TDIR, TMEM, TSEG, TSS		

Use the `SEG` operator to assign the number of free memory segment buffers in *compiled memory* to a variable (`VAR`), or to make a comparison against another value. “Compiled memory” is the partition of the Gemini6K’s non-volatile memory that stores compiled profiles & PLC programs. Compiled profiles/programs could be a *motion profile* (a series of `GOBUF` commands), or a *PLC program* (for PLC Scan Mode).

System status bit (see `TSSF`, `TSS`, and `SS`) 29 is set when the compiled memory is 75% full, and bit 30 is set if the compiled memory is 100% full.

**Syntax:** `VARn=SEG` where “n” is the variable number,  
or `SEG` can be used in an expression such as `IF(SEG=1)`

## SFB                              Select Servo Feedback Source

Type	Drive Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_>SFB<i>	GT6K	n/a
Units	i = feedback source identifier	GV6K	6.0
Range	i = 1 (encoder), or 4 (resolver)		
Default	1		(applicable only to servo axes)
Response	SFB    *SFB1		
See Also	ERES, SRSET, TPE		

**AUTO-SETUP:** This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6). If you did not use the configuration utility or are not using a Parker Motor, this command is set to 1. You may have to manually set this parameter. (Refer to `DMTR` for a list of auto-configured commands.)

Use the `SFB` command to select the servo feedback source to be used. The options are:

Options	Physical Location	Measurement*	Resolution Command
1—Encoder	<b>MOTOR FEEDBACK</b> connector only	Encoder counts	ERES (default is 4000 counts/rev)
4—Resolver	<b>MOTOR FEEDBACK</b> connector (GV6K must have Resolver option)	Resolver counts	ERES (fixed at 4096 counts/rev)

\* With scaling enabled (`SCALE1`), feedback is scaled by the `SCLD` value.

---

## SGAF Acceleration Feedforward Gain

Type	Tuning	Product	Rev
Syntax	<a_><!>SGAF<i>	GT6K	n/a
Units	i = percent	GV6K	6.0
Range	0 - 500		
Default	100		
Response	SGAF: *SGAF100		(applicable only to servo axes)
See Also	DMTJ, LJRAT, SFB, SGENB, SGSET, SGVF, TGAIN, TSGSET		

---

Use the 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. Acceleration feedforward control does not affect the servo system's stability, nor does it have any effect at constant velocity.

The SGAF value is multiplied by the *commanded acceleration* calculated by the GV6K's move profile routine to produce an estimated torque command that is added to the servo control signal. The value is normalized to the current setting of both the motor inertia and load inertia ratio (DMTJ and LJRAT, respectively) as shown in the equation below.

$$\text{Estimated acceleration torque} = \underbrace{\text{DMTJ} (1 + \text{LJRAT})}_{\text{SGAF value} = 100\%} \cdot \text{acceleration command}$$

Setting SGAF to 100% will theoretically produce zero following error during the acceleration and deceleration part of a move profile. This assumes that the drive or motor are not being current limited, the values for inertia are accurate and the models used for analysis are correct. You can adjust the value of SGAF from zero to as high as 5 times or 500% (SGAF500) of the theoretical value. For example, SGAF200 sets the acceleration feedforward to 200% of theoretical value.

### Working with servo gains.

- Servo tuning process: refer to the *Hardware Installation Guide* or to the Motion Planner help system.
- Check the values of all active gains (SGAF is one of many servo gains): use TGAIN.
- Creating and invoking gain sets: see SGSET, SGENB, TGAIN, TSGSET.

---

## SGENB Enable a Servo Gain Set

Type	Tuning	Product	Rev
Syntax	<a_><!>SGENB<i>	GT6K	n/a
Units	i = gain set identification number (see SGSET command)	GV6K	6.0
Range	1 - 3		
Default	n/a		
Response	n/a		(applicable only to servo axes)
See Also	SFB, SGAF, SGSET, SGVF, TGAIN, TSGSET		

---

SGENB allows you to enable one of the three gain sets. The gain sets are established 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.

### Gain Set Elements:

- DIBW (current loop bandwidth)
- DMTLIM (torque/force limit)
- DMVLIM (velocity limit)
- DNOTAD (notch filter A depth)
- DNOTAF (notch filter A frequency)
- DNOTAQ (notch filter A quality factor)
- DNOTBD (notch filter B depth)
- DNOTBF (notch filter B frequency)
- DNOTBQ (notch filter A quality factor)
- DNOTLD (notch lead filter break frequency)
- DNOTLG (notch lag filter break frequency)
- DPBW (position loop bandwidth)
- LDAMP (load damping)
- LJRAT (load-to-rotor inertia ratio or load-to-force mass ratio)
- SGAF (acceleration feedforward gain)
- SGINTE (integrator enable)
- SGIRAT (current damping ratio)
- SGPRAT (position loop ratio)
- SGPSIG (velocity/position bandwidth ratio)
- SGVF (velocity feedforward gain)
- SGVRAT (velocity damping ratio)

For help with servo tuning procedures, refer to the *Hardware Installation Guide* or to the Motion Planner help system.

#### Example:

```
SGVF5           ; Set the velocity feedforward gain
SGAF1           ; Set the acceleration feedforward gain
SGSET3         ; Assign SGVF & SGAF gains to servo gain set #3
SGVF9           ; Set the velocity feedforward gain
SGAF18         ; Set the acceleration feedforward gain
SGSET1         ; Assign SGAF & SGVF gains to servo gain set #1
SGENB1         ; Enable gain set #1 to be in effect
TSGSET3        ; Display the value for all gains in gain set #3.
                ; (see sample response in TSGSET command description)
```

---

## SGINTE Integrator Enable

Type	Tuning	Product	Rev
Syntax	<a_><!>SGINTE<b>	GT6K	n/a
Units	b = enable bit	GV6K	6.0
Range	0 (disable) or 1 (enable) or 2 (enable only at end of move)		
Default	1 (enabled)		(applicable only to servo axes)
Response	SGINTE: *SGINTE1		
See Also	LDAMP, SGPRAT, SGVRAT, TGAIN, TSGSET		

---

The `SGINTE` command enables/disables the integrator in the velocity and position loops. When enabled (the default), the integrator progressively reduces the velocity error in proportion to how long the error has persisted. This helps to reduce the effects of load friction on performance. This effect is most noticeable when a constant velocity is commanded for a significant period of time.

Because position error causes the generation of a corrective velocity command, the integrator also reduces steady-state position error to zero, and will result in more accurate final positioning in applications where friction is present. If you use the `SGINTE2` setting, the integrator will be disabled during the move, and will be enabled only at the end of the move.

In some situations, the integrator can reduce stability, especially if there is high stiction (non-linear friction). In this case, a limit-cycle can result. This will typically take the form of a square-wave oscillation around the final position. (Most other causes of instability result in nearly sinusoidal oscillations.) In such cases, disabling the integrator can improve performance.

### Working with servo gains.

- Servo tuning process: refer to the *Hardware Installation Guide* or to the Motion Planner help system.
- Check the values of all active gains (`SGINTE` is one of many servo gains): use `TGAIN`.
- Creating and invoking gain sets: see `SGSET`, `SGENB`, `TGAIN`, `TSGSET`.



---

## SGIRAT Current Damping Ratio

Type	Tuning	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>SGIRAT<r>	GT6K	n/a
Units	r = ratio	GV6K	6.0
Range	0.500 to 2.000 : ±0.001		
Default	1.000		(applicable only to servo axes)
Response	SGIRAT: *SGIRAT.775		
See Also	DIBW, SGPSIG, SGPRAT, SGVRAT, TGAIN, TSGSET		

---

The SGIRAT command sets the damping ratio (Zeta) of the current loop. Higher values produce a more stable current loop response. Lower values provide faster current response but increase current overshoot.

### Working with servo gains.

- Servo tuning process: refer to the *Hardware Installation Guide* or to the Motion Planner help system.
- Check the values of all active gains (SGIRAT is one of many servo gains): use TGAIN.
- Creating and invoking gain sets: see SGSET, SGENB, TGAIN, TSGSET.

---

## SGPRAT Position Damping Ratio

Type	Tuning	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>SGPRAT<r>	GT6K	n/a
Units	r = ratio	GV6K	6.0
Range	0.500 to 2.000 : ±0.001		
Default	1.000		(applicable only to servo axes)
Response	SGPRAT: *SGPRAT.542		
See Also	DPBW, LDAMP, LJRAT, SGIRAT, SGPSIG, SGVRAT, TGAIN, TSGSET		

---

The SGPRAT command sets the damping ratio (Zeta) of the position loop. Higher values produce a more stable position loop. Lower values provide faster position response but increase position overshoot.

### Working with servo gains.

- Servo tuning process: refer to the *Hardware Installation Guide* or to the Motion Planner help system.
- Check the values of all active gains (SGPRAT is one of many servo gains): use TGAIN.
- Creating and invoking gain sets: see SGSET, SGENB, TGAIN, TSGSET.

---

## SGPSIG Velocity/Position Bandwidth Ratio

Type	Tuning	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>SGPSIG<r>	GT6K	n/a
Units	r = ratio	GV6K	6.0
Range	0.100 to 2.000 : ±0.001		
Default	1.000		(applicable only to servo axes)
Response	SGPSIG: *SGPSIG1.250		
See Also	DMODE, DPBW, SGIRAT, SGPRAT, SGVRAT, TGAIN, TSGSET		

---

The SGPSIG command sets the ratio (frequency spread) between the internal velocity loop bandwidth and the position loop bandwidth (DPBW). When operating in one of the position modes (see list below), the velocity loop bandwidth will track position loop bandwidth adjustments, with the ratio set by SGPSIG.

Position modes are DMODE12 and DMODE17 (see DMODE for descriptions).

### Working with servo gains.

- Servo tuning process: refer to the *Hardware Installation Guide* or to the Motion Planner help system.
- Check the values of all active gains (SGPSIG is one of many servo gains): use TGAIN.
- Creating and invoking gain sets: see SGSET, SGENB, TGAIN, TSGSET.

---

## SGSET Save a Servo Gain Set

Type	Tuning	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>SGSET<i>	GT6K	n/a
Units	i = gain set identification number	GV6K	6.0
Range	1-3		
Default	n/a		
Response	n/a		(applicable only to servo axes)
See Also	SFB, SGAF, SGENB, SGVF, TGAIN, TSGSET		

---

The SGSET command saves the presently active gain values (see list below) as a set of gains. Up to three sets of gains can be saved. Any gain set can be displayed using the TSGSET command. To report the presently active gain values, enter the TGAIN 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.

### Gain Set Elements:

- DIBW (current loop bandwidth)
- DMTLIM (torque/force limit)
- DMVLIM (velocity limit)
- DNOTAD (notch filter A depth)
- DNOTAF (notch filter A frequency)
- DNOTAQ (notch filter A quality factor)
- DNOTBD (notch filter B depth)
- DNOTBF (notch filter B frequency)
- DNOTBQ (notch filter A quality factor)
- DNOTLD (notch lead filter break frequency)
- DNOTLG (notch lag filter break frequency)
- DPBW (position loop bandwidth)
- LDAMP (load damping)
- LJRAT (load-to-rotor inertia ratio or load-to-force mass ratio)
- SGAF (acceleration feedforward gain)
- SGINTE (integrator enable)
- SGIRAT (current damping ratio)
- SGPRAT (position loop ratio)
- SGPSIG (velocity/position bandwidth ratio)
- SGVF (velocity feedforward gain)
- SGVRAT (velocity damping ratio)

For help with servo tuning procedures, refer to the *Hardware Installation Guide* or to the Motion Planner help system.

### Example:

```
SGVF5           ; Set the velocity feedforward gain
SGAF1           ; Set the acceleration feedforward gain
SGSET3         ; Assign SGVF & SGAF gains to servo gain set #3
SGVF9           ; Set the velocity feedforward gain
SGAF18         ; Set the acceleration feedforward gain
SGSET1         ; Assign SGAF & SGVF gains to servo gain set #1
SGENB1         ; Enable gain set #1 to be in effect
TSGSET3        ; Display the value for all gains in gain set #3.
                ; (see sample response in TSGSET command description)
```

---

## SGVF Velocity Feedforward Gain

Type	Tuning	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>SGVF<i>	GT6K	n/a
Units	i = percent	GV6K	6.0
Range	0 - 500		
Default	100		
Response	SGVF: *SGVF100		(applicable only to servo axes)
See Also	DMTD, LDAMP, SFB, SGAF, SGENB, SGSET, TGAIN, TSGSET		

---

Use the SGVF command to set the velocity feedforward gain. Velocity feedforward control improves *position tracking performance* when the system is commanded to move at constant velocity. The velocity tracking error is mainly attributed to viscous friction.

The SGVF value is multiplied by the *commanded velocity* calculated by the GV6K's move profile routine to produce an estimated torque command that gets added to the servo control signal. The value is normalized

to the current setting of both the motor and load viscous damping terms (DMTD and LDAMP, respectively) as shown in the equation below.

$$\text{Estimated velocity torque} = \underbrace{(\text{DMTD} + \text{LDAMP})}_{\text{SGVF value}=100\%} \cdot \text{velocity command}$$

Setting SGVF to 100% will theoretically produce zero following error during the constant velocity portion of a move profile. This assumes that the drive or motor are not being current limited, the values for viscous damping are accurate and the models used for analysis are correct. You can adjust the value of SGVF from zero to as high as 5 times or 500% (SGVF500) of the theoretical value.

**Working with servo gains.**

- Servo tuning process: refer to the *Hardware Installation Guide* or to the Motion Planner help system.
- Check the values of all active gains (SGVF is one of many servo gains): use TGAIN.
- Creating and invoking gain sets: see SGSET, SGENB, TGAIN, TSGSET.

**Example:**

SGVF200 ; Set velocity feedforward to 200% of theoretical value

---

<b>SGVRAT</b>		<b>Velocity Damping Ratio</b>	
Type	Tuning	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>SGVRAT<r>	GT6K	n/a
Units	r = ratio	GV6K	6.0
Range	0.500 to 2.000 : ±0.001		
Default	1.000		(applicable only to servo axes)
Response	SGVRAT: *SGVRAT.224		
See Also	LDAMP, LJRAT, SGIRAT, SGPSIG, SGPRAT, SGSET, TGAIN, TSGSET		

---

The SGVRAT command sets the damping ratio (Zeta) of the velocity loop. Higher values produce a more stable velocity loop. Lower values provide faster velocity response but increase velocity overshoot.

**Working with servo gains.**

- Servo tuning process: refer to the *Hardware Installation Guide* or to the Motion Planner help system.
- Check the values of all active gains (SGVRAT is one of many servo gains): use TGAIN.
- Creating and invoking gain sets: see SGSET, SGENB, TGAIN, TSGSET.

---

<b>SHALL</b>		<b>Hall Sensor Inversion</b>	
Type	Drive Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>SHALL<i> (does not take effect until RESET, DRESET or cycle power)	GT6K	n/a
Units	i = control option #	GV6K	6.0
Range	0 (do not invert) 1 (invert)		(applicable only to servo axes)
Default	0		
Response	SHALL: *SHALL0		
See Also	THALL		

---

**NOTE:** This command does not take effect until you cycle power to the drive, or issue a RESET or DRESET command.

**Encoder Motors (with SFB1):** The SHALL command controls the logic sense of the Hall sensors. To invert the sensors, use the SHALL1 command. To check the present value of the Hall sensors, use the THALL command.

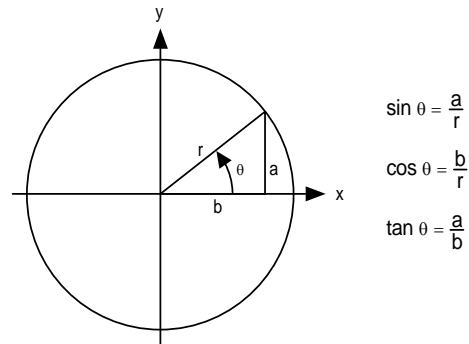
For a complete description on how to troubleshoot Hall sensors, especially for non-Compumotor motors, refer to the *Hardware Installation Guide* section on using non-Compumotor motors.

**Resolver Motors (with SFB4):** If you use a resolver motor, SHALL is not applicable.

## [ SIN( ) ] Sine

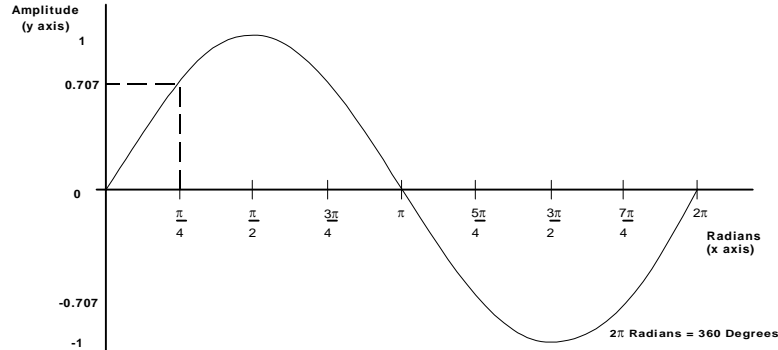
Type	Operator (Trigonometric)	<b>Product</b>	<b>Rev</b>
Syntax	... SIN(r) (See below)	GT6K	6.0
Units	r = value in radian or degrees based on RADIAN command	GV6K	6.0
Range	±17500.0000000 radians		
Default	n/a		
Response	n/a		
See Also	[ ATAN ], [ COS ], [ PI ], RADIAN, [ TAN ], VAR		

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:  $\text{sine } \theta = \frac{a}{r}$



If a value is given in radians and a conversion is needed to degrees, use the formula:  $360^\circ = 2\pi$  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:**

```
RADIAN1
VAR1=5 * SIN(PI/4) ; Set variable 1 equal to 5 times the sine of Pi divided by 4
```

---

## SINAMP Virtual Master Sine Wave Amplitude

Type	Following	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>SINAMP<i>	GT6K	6.0
Units	i = amplitude	GV6K	6.0
Range	0-8192 (max. peak to peak is 16384)		
Default	0		
Response	SINAMP: *SINAMP0		
See Also	FOLMAS, FVMACC, FVMFRQ, SINANG, SINGO		

---

Use the SINAMP command to define the amplitude of the internal sine wave when it has been designated as the virtual master. By designating the internal sine wave as a master, the user may produce a sinusoidally oscillating motion, with control of the phase, amplitude, and center of oscillation.

The SINAMP command allows a change in follower amplitude without changing the center of oscillation. It affects the sine wave immediately, without any built-in ramp in amplitude. If a gentle change in amplitude is desired, write a user program which repeatedly issues the command with small changes in value until the desired value is reached.

The peak-to-peak amplitude of a virtual master sine wave is twice the value specified with the SINAMP command.

---

## SINANG Virtual Master Sine Wave Angle

Type	Following	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>SINANG<r>	GT6K	6.0
Units	degrees	GV6K	6.0
Range	0.0-360.0		
Default	0		
Response	SINANG: *SINANG0		
See Also	FOLMAS, FVMACC, FVMFRQ, SINAMP, SINGO		

---

The SINANG command is used to define the phase angle when the internal sine wave is designated as the virtual master. By designating the internal sine wave as a master, the user may produce a sinusoidally oscillating motion, with control of the phase, amplitude, and center of oscillation.

The internal sine wave uses the variable count frequency command (FVMFRQ) to increase or decrease the angle from which the sine is calculated. Each count of the count frequency changes the angle by one-tenth (0.1) of a degree. For example, a FVMFRQ value of 3600 would create an angular frequency of 3600 tenths of degrees per second, or 1 cycle per second. When used as a source for the sine wave, the maximum value for FVMFRQ is 144000. This results in a maximum of 40 Hz angular frequency. Frequencies higher than this are not allowed because they may be subject to aliasing.

---

## SINGO Virtual Master - Initiate Internal Sine Wave

Type	Following	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>SINGO<b>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	b = 1 (restart sine wave from previous angle & amplitude) or 0 (stop sine wave)		
Default	0		
Response	SINGO *SINGO0		
See Also	FOLMAS, FVMACC, FVMFRQ, SINAMP, SINANG		

---

The SINGO command is used to restart the internal sine wave from zero degrees. By designating the internal sine wave as a master, the user may produce a sinusoidally oscillating motion, with control of the phase, amplitude, and center of oscillation.

The SINGO command with a “0” parameter abruptly stops the sine wave, without changing its current magnitude. Using the SINGO command with a “1” parameter abruptly starts the sine wave, also without changing its current magnitude. To gently pause the follower output, change the FVMFRQ value to zero with a moderate FVMACC value; to resume the follower output, restore the original FVMFRQ value.

The SINGO command with a “1” parameter always starts at the previous angle, which may not be the desired start of oscillation. The SINANG command will instantly change the angle and corresponding sine of the angle. This represents an abrupt change in master position. If the follower axis is still following when this occurs, there will be an abrupt change in commanded follower position. To start the follower properly, move the follower to the desired start position first (using MCO, D, GO), then issue SINANG, then MC1, GO1, and finally SINGO.

---

<b>SMPER</b>		<b>Maximum Allowable Position Error</b>	
Type	Servo	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>SMPER<r>	GT6K	n/a
Units	r = feedback device counts (scalable with SCLD)	GV6K	6.0
Range	0-200,000,000 (after conversion to steps) (0 = do not monitor position error condition)		(applicable only to servo axes)
Default	4000		
Response	SMPER: *SMPER4000		
See Also	[ AS ], [ ER ], ERES, ERROR, ERRORP, SCALE, SCLD, SFB, TANI, TAS, TER, TERRLG, TFB, TPC, TPE, TPER		

---

SMPER determines the maximum position error (in feedback device counts) allowed before an error condition occurs. The position error is the difference between the commanded position and the actual position as read by the feedback device. When the position error exceeds the value entered by the SMPER command, an error condition is latched (see TAS or AS bit #23 and TER bit #12) and the drive faults (issues a shutdown – DRIVE0). The DRIVE1 command re-enables the drive, clears TAS bit #23 and TER bit #12, and sets the commanded position (TPC) equal to the actual feedback device position (TPE). Incremental devices will be zeroed. The position error is monitored once per system update period.

**CAUTION:** If the SMPER value is set to zero (SMPERØ), the position error condition is not monitored, allowing the position error to accumulate without causing a fault.

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:**

```

ERES4000           ; Set feedback resolution to 4000 counts/rev
SMPER4000          ; Set maximum allowable position error to 1 rev before
                   ; a fault condition will occur.

```

---

<b>SMVER</b>		<b>Maximum Allowable Velocity Error</b>	
Type	Drive Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>SMVER<r>	GT6K	n/a
Units	r = feedback device revs/sec (linear motors: see DMEPIT for linear/rotary conversion)	GV6K	6.0
Range	0.000000 to 200.000000 : ±0.000001		(applicable only to servo axes)
Default	0.000000 (do not monitor velocity error condition)		
Response	SMVER: *SMVER10.000000		
See Also	DMEPIT, TASX, TVE, TVEL, TVELA		

---

SMVER determines the maximum velocity error allowed before an error condition occurs. The velocity error is the difference between the commanded velocity (TVEL) and estimated actual velocity (TVELA). If the error exceeds this value, a fault will result in which the drive is shut down (DRIVE0) and TASX bit #9 is set. The DRIVE1 command re-enables the drive, clears TASX bit #9, and sets TVEL equal to TVELA.

You can check the actual velocity error with the TVE command.

If the SMVER value is set to zero (SMVER0), the velocity error condition is not monitored, allowing the velocity error to accumulate without causing a fault.

---

## [ SQRT() ] Square Root

Type	Operator (Mathematical)	<b>Product</b>	<b>Rev</b>
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
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:**

VAR1=SQRT(25) ; Set variable 1 equal to the square root of 25 (result = 5)

---

## SRSET Resolver Offset Angle

Type	Drive Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>SRSET<r>	GT6K	n/a
Units	r = angle in degrees	GV6K	6.0
Range	-180.0 to +180.0 : ±0.1 (no value = use auto align if in DMODE11)		(applicable only to servo axes)
Default	0		
Response	SRSET: (returns the calculated value only if in DMODE11)		
See Also	DMODE, SFB, TSROFF		

---

**AUTO-SETUP:** This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6). If you did not use the configuration utility or are not using a Parker Motor, this command is set to zero. (Refer to DMTR for a list of auto-configured commands.)

The SRSET value becomes the new resolver offset angle. When no value is specified, and the drive is in DMODE11 (feedback alignment mode), then a routine is executed to automatically set the resolver angle.

**WARNING:** Motion (less than 1 rev) will occur when you initiate the auto alignment mode. To ascertain the actual offset angle, use the TSROFF command.

The SRSET value is saved in non-volatile memory.

---

## [ SS ] System Status

Type	Assignment or Comparison	<b>Product</b>	<b>Rev</b>
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	IF, TCMDER, TRGFN, [ TRIG ], TTRIG, TSS, TSSF, TSTAT, VARB		

---

Use the SS operator 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 0 through 9.

**Syntax:** VARBn=<i%>SS where “n” is the binary variable number, or SS can be used in an expression such as IF(SS=b1101), or IF(SS=h7F). **NOTE:** If you are using multi-tasking, be aware that each task has its own system status register. If you wish to check the system status of an external task (a task other than the task that is executing the SS operator), then you must prefix the SS operator to address the targeted task (e.g., 2%SS for the system status of Task 2).

The function of each system status bit is shown below.

BIT (Left to Right)	Function (1 = yes, 0 = no)	BIT (Left to Right)	Function (1 = yes, 0 = no)
1	System Ready	17	Loading Thumbwheel Data (TW)
2	RESERVED	18	External Program Select Mode (INSELP)
3	Executing a Program (see EXE)	19	Dwell in Progress (T command)
4	Immediate Command (set if last command was immediate)	20	Waiting for RP240 Data—DREAD or DREADF
5	In ASCII Mode	21	RP240 Connected — current PORT setting only
6	In Echo Mode — current PORT setting only	22	Non-volatile Memory Error
7	Defining a Program	23	GV6K: Servo data gathering transmission in progress; GT6K: Reserved
8	In Trace Mode	24	Suspended on Swap (see LOCK)
9	In Step Mode	25	RESERVED
10	In Translation Mode	26	Suspended on COM1 (see LOCK)
11	Command Error Occurred (bit is cleared when TCMER is issued)	26	Suspended on COM2 (see LOCK)
12	Break Point Active (BP)	28	Program Pending (see EXE command)
13	Pause Active	29	Compiled memory is 75% full
14	Wait Active (WAIT)	30	Compiled memory is 100% full
15	Monitoring On Condition (ONCOND)	31 *	Compile operation failed (PCOMP) **
16	Waiting for Data (READ)	32	EXE Failed (see EXE command)

- \* Bit #31: failed PCOMP compile is cleared on power up, RESET, or after successful compile. Possible causes include:
- Errors in profile design (e.g., change direction while at non-zero velocity; distance & velocity equate to < 1 count per system update; preset move profile ends in non-zero velocity)
  - Profile will cause a Following error (see TFSF, TFS, or FS command descriptions)
  - Out of memory (see SS bit #30)
  - Axis already in motion at the time of the PCOMP command
  - Loop programming errors (e.g., no matching PLOOP or PLN; more than 4 embedded PLOOP/END loops)
  - PLCP program contains invalid commands.

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\_XXX0\_XXXX\_XXXX\_XXXX\_XXXX\_XXXX.

**Example:**

```
VARB1=SS ; System status assigned to binary variable 1
IF(SS=b111011X11) ; If the system status contains 1s in bit locations 1, 2, 3,
; 5, 6, 8, & 9, and a 0 in bit location 4, do the IF
; statement
IF(SS=h7F00) ; If the system status contains 1s in bit locations 1, 2, 3,
; 5, 6, 7, & 8, and 0s in every other bit location, do the IF
; statement
NIF ; End of second IF statement
NIF ; End of first IF statement
```



---

## STARTP      Start-Up Program

Type	Program Definition	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>STARTP<t>	GT6K	6.0
Units	t = text (name of program)	GV6K	6.0
Range	Text name of 6 characters or less		
Default	n/a		
Response	STARTP:    *STARTP MAIN		
See Also	DEF, RESET, SCALE		

---

The STARTP command assigns an existing program to be executed automatically when the Gem6K drive is powered up or reset. A reset may be invoked by sending the RESET command, or activating the hardware Reset input (pin 3 on the DRIVE I/O connector). For example, STARTP MAIN assigns program MAIN (previously defined with DEF MAIN) as the startup program.

If the program that is identified as the STARTP program is deleted with the DEL command, the STARTP assignment 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.

### Example:

```
DEL WakeUp           ; Delete program WakeUp
DEF WakeUp           ; Begin definition of program WakeUp
INFNC1-B             ; Assign input #1 as a BCD program select input
INFNC2-B             ; Assign input #2 as a BCD program select input
INFNC3-B             ; Assign input #3 as a BCD program select input
END                  ; End program definition
STARTP WakeUp        ; Assign program WakeUp as the startup program
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)
```

---

## STEP            Single Step Mode Enable

Type	Program Debug Tool	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>STEP<b>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	b = 0 (disable), 1 (enable) or X (don't care)		
Default	0		
Response	STEP:        *STEP0		
See Also	[ # ], BP, [ SS ], TRACE, TRACEP, TRANS, TSS		

---

The 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. Enable trace mode (TRACE1)
4. Run the program (RUN)
5. Use the immediate pound (!#) to step through the program

Each step (!#) command will initiate the next command to be processed.

### Example:

```
DEF tester           ; Begin definition of program named tester
V1                   ; Set velocity to 1 unit/sec
                    ; (Note: This command will not be executed until a !# sign
                    ; is received.)
A10                  ; Set acceleration to 10 units/sec/sec
D1                   ; Set distance to 1 unit
GO1                  ; Initiate motion
OUT11X1              ; Turn on outputs 1, 2, and 4, leave 3 unchanged
```

```

END                ; End program definition
STEP1             ; Enable single step mode
RUN tester        ; Execute program named tester
; *****
; * At this point no action will occur because single step mode      *
; * has been enabled. Here's how to execute commands:                *
; * !#2 (Execute 1st 2 commands in the program: V1 & A10)           *
; * !#  (Execute 1 command: command to be executed is D1)         *
; * !#1 (Execute 1 command: command to be executed is G01)         *
; * !#2 (Execute 2 commands: commands to be executed are OUT11X1 & END) *
; *****

```

---

## STRGTD Target Distance Zone

Type	Servo	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>STRGTD<r>	GT6K	n/a
Units	r = distance units (scalable with SCLD)	GV6K	6.0
Range	0-999,999,999.99999		
Default	50		(applicable only to servo axes)
Response	STRGTD: *STRGTD50		
See Also	[ AS ], SCLD, STRGTE, STRGTT, STRGTV, TAS, TSTLT		

---

STRGTD 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 load 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 Settling Mode, the load'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. Axis status bit #24 (see TASF, TAS, or AS) indicates when the axis is within the zone specified with STRGTD and STRGTV; this status bit is usable even if the Target Zone Mode is not enabled (STRGTE0).

If the load does not settle into the target zone before the timeout period set by STRGTT, the Gem6K drive detects an error (see TASF, TAS, or AS bit #25, and TER bit #11). If this error occurs, you can prevent subsequent command and/or move execution by enabling bit #11 in 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 Programmer's Guide.

**Example** (see STRGTE):

---

## STRGTE Enable Target Zone Settling Mode

Type	Servo	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>STRGTE<b>	GT6K	n/a
Units	n/a	GV6K	6.0
Range	b = 0 (disable) or 1 (enable)		
Default	0		(applicable only to servo axes)
Response	STRGTE: *STRGTE0		
See Also	COMEXC, ERROR, STRGTD, STRGTT, STRGTV, TAS, TER, TSTLT		

---

STRGTE enables or disables the Target Zone Settling Mode. When using the target zone settling criterion, the load's actual position (TPE) and actual velocity (TVELA) must be within the *target zone* (that is, within the position zone defined by STRGTD and within the velocity zone defined by STRGTV) before motion can be determined complete. Axis status bit #24 (see TASF, TAS, or AS) indicates when the axis is within the

zone specified with STRGTD and STRGTV; this status bit is usable even if the Target Zone Mode is not enabled (STRGTE0).

If the load does not settle into the target zone before the timeout period set by STRGTT, the Gem6K drive detects an error (see TASF, TAS, or AS bit #25, and TER bit #11). If this error occurs, you can prevent subsequent command and/or move execution by enabling bit #11 in 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 Programmer's Guide.

**Example:**

```
STRGTD5           ; Sets the distance target zone to +/-5 units
STRGTV.01        ; Sets the velocity target zone to <= 0.01 units/sec
STRGTT10         ; Sets the timeout period to 10 milliseconds
STRGTE1          ; Enables the target zone criteria
;
; Given these target zone commands, a move with a distance of, for example,
; 8,000 units (D8000) must end up between position 7,995 and 8,005 and settle
; to <=0.01 units/sec within 10 ms after the commanded profile is complete.
```

---

<b>STRGTT</b>		<b>Target Settling Timeout Period</b>	
Type	Servo	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>STRGTT<i>	GT6K	n/a
Units	i = milliseconds	GV6K	6.0
Range	0-5000		
Default	1000		
Response	STRGTT: *STRGTT1000		(applicable only to servo axes)
See Also	[ AS ], [ ER ], ERROR, ERRORP, STRGTD, STRGTE, STRGTV, TAS, TER, TSTLT		

---

STRGTT sets the maximum time allowed for the load to settle within the defined target zone; exceeding this time period will generate an error condition. This command is useful only if Target Zone Settling Mode is enabled with the STRGTE command.

When using the Target Zone Settling Mode, the load's actual position (TPE) and actual velocity (TVELA) 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. Axis status bit #24 (see TASF, TAS, or AS) indicates when the axis is within the zone specified with STRGTD and STRGTV; this status bit is usable even if the Target Zone Mode is not enabled (STRGTE0).

If the load does not settle into the target zone before the timeout period set by STRGTT, the Gem6K drive detects an error (see TASF, TAS, or AS bit #25, and TER bit #11). If this error occurs, you can prevent subsequent command and/or move execution by enabling bit #11 in 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 Programmer's Guide.

**Example** (see STRGTE):

---

## STRGTV Target Velocity Zone

Type	Servo	Product	Rev
Syntax	<a_><!>STRGTV<r>	GT6K	n/a
Units	r = units/sec (scalable by SCLV) (linear motors: see DMEPIT for linear/rotary conversion)	GV6K	6.0
Range	0.0000-200.0000	(applicable only to servo axes)	
Default	1.00000		
Response	STRGTV: *STRGTV1.0000		
See Also	[ AS ], DMEPIT, ERROR, ERRORP, SCLV, STRGTD, STRGTE, STRGTT, TAS, TER, TSTLT		

---

STRGTV sets the target velocity zone for use in the Target Zone Settling Mode. The target velocity zone is a velocity range that the load 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 load'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. Axis status bit #24 (see TASF, TAS, or AS) indicates when the axis is within the zone specified with STRGTD and STRGTV; this bit is usable even if the Target Zone Mode is not enabled (STRGTE0).

If the load does not settle into the target zone before the timeout period set by STRGTT, the Gem6K drive detects an error (see TASF, TAS, or AS bit #25, and TER bit #11). If this error occurs, you can prevent subsequent command and/or move execution by enabling bit #11 in 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 Programmer's Guide.

**Example** (see STRGTE):

---

## [ SWAP ] Task Swap Assignment

Type	Assignment or Comparison; Multi-Tasking	Product	Rev
Syntax	See Below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	%, [ SS ], TTASK, TSWAP, TSKTRN, TSKAX, TSS		

---

The SWAP command allows a binary bit pattern indicating the tasks that are currently active to be assigned to a binary variable, or evaluated in a conditional statement such as IF or WAIT. This is useful for ascertaining which tasks have any activity. To ascertain exactly what activity a specific task has at a given time, use the system status commands (SS or TSS).

SWAP's binary 10-bit pattern represents tasks 1-10, from left to right. A "1" indicates that the task is active, and a "0" indicates that the task is inactive. The "Task Supervisor", represented by task Ø, is always active and is therefore not included in the SWAP and TSWAP status.

**Syntax:** VARBn=SWAP where "n" is the binary variable number, or SWAP can be used in an expression such as IF(SWAP=b1001000000) or IF(SWAP.3=b1) or IF(SWAP=h7F0).

To check the status of only one task, you may use the bit select (.) operator. For example, VARB1=SWAP.2 assigns the binary state of Task2 to binary variable 1; or WAIT(SWAP.2=b1) establishes a wait condition that evaluates true when Task2 becomes active.

---

<b>T</b>		<b>Time Delay</b>			
Type	Program Flow Control	<b>Product</b>		<b>Rev</b>	
Syntax	<a_><!>T<r>	GT6K		6.0	
Units	r = seconds	GV6K		6.0	
Range	0.001-999.999				
Default	n/a				
Response	n/a				
See Also	GOWHEN, PS, [ SS ], [ 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 milliseconds (ms). 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.005 produces a 6 ms time delay.

VARI variables may be substituted for the T command value (e.g., T(VARI5)).

**Example:**

```
T5           ; Wait 5 seconds before executing TPE command
TPE         ; Transfer position of encoder to the terminal
```

---

<b>TACC</b>		<b>Transfer Commanded Acceleration</b>			
Type	Transfer	<b>Product</b>		<b>Rev</b>	
Syntax	<a_><!>TACC	GT6K		6.0	
Units	revs/sec/sec (linear motors: see DMEPIT for linear/rotary conversion)	GV6K		6.0	
Range	n/a				
Default	n/a				
Response	TACC: *TACC100				
See Also	A, AD, DMEPIT, HOMA, LHAD, LSAD, TVEL				

---

TACC reports the commanded acceleration.

## [ TAN( ) ] Tangent

Type	Operator (Trigonometric)	Product	Rev
Syntax	... TAN(r) (See below)	GT6K	6.0
Units	r = radians or degrees depending on RADIAN command	GV6K	6.0
Range	±17500.0000000 radians		
Default	n/a		
Response	n/a		
See Also	[ ATAN ], [ COS ], [ PI ], RADIAN, [ SIN ], VAR		

The 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:

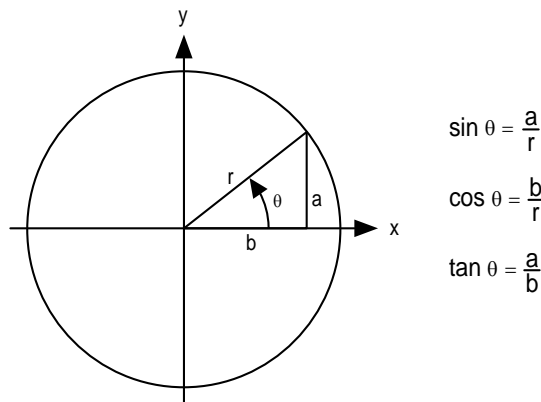
$$\tan \theta = \frac{a}{b}$$

If a value is given in radians and a conversion is needed to degrees, use the following formula:  
 $360^\circ = 2\pi$  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:

VAR1=5 \* TAN(PI/4) ; Set variable 1 = 5 times the tangent of Pi divided by 4



## TANI Transfer Analog Input Voltage

Type	Transfer	Product	Rev
Syntax	<a_><!><@><B>TANI<.i>	GT6K	6.0
Units	B = I/O brick i = location on I/O brick	GV6K	6.0
Range	i = 1-32		
Default	n/a		
Response	1TANI *1TANIx,x,x,x,x,x,x,x +5.802,-4.663,-4.972, +6.023,+2.126,+2.223, ... x,x,x,x,x,x,x,x x,x,x,x,x,x,x,x 1TANI.10 *-4.663		
See Also	ANIRNG, [ ANI ], [ FB ], TFB,		

The TANI command returns the voltage level present at the ANI analog inputs. The value reported with the TANI command is measured in volts.

If you omit the brick number, the value of the onboard analog input (ANI .1) will be reported. To determine the analog value from a specific input on an external I/O brick, use the bit select operator (.). For example, to check the voltage of the 2<sup>nd</sup> analog input on the 3<sup>rd</sup> SIM (I/O location 18) of I/O brick 2, use the 2TANI .18 command. To understand more about the location of I/O points on external I/O bricks, see page 8.

The TANI value is derived from the voltage applied to the corresponding analog input and ground. The analog value is determined from a 12-bit analog-to-digital converter (ADC). Under the default ANI voltage range, set with ANIRNG, the range of the ANI operator is -10.000VDC to +10.000VDC (see ANIRNG command for optional voltage ranges).

## TANO Transfer Analog Output Value

Type	Transfer	Product	Rev
Syntax	<a_><!><@><B>TANO.i	GT6K	6.0
Units	B = I/O brick number i = input/output location on brick "B"	GV6K	6.0
Range	B = 1-8 (depending on I/O brick configuration) i = 1-32 (depending on I/O brick configuration)		
Default	n/a		
Response	!TANO.9: *1TANO.9=0.00		
See Also	[ ANO ], ANO, TIO		

Use the TANO command to report the value of an analog output channel.

### Example:

```
>!TANO.9 ; Report the commanded analog output voltage at bit 9 of I/O brick 1.
*1TANO.9=0.00
```

## TAS Transfer Axis Status

Type	Transfer	Product	Rev
Syntax	<a_><!>TAS<.i>	GT6K	6.0
Units	i = bit location for the axis (See below)	GV6K	6.0
Range	1-32		
Default	n/a		
Response	TAS: *TAS0000_0000_0000_0000_0000_0000_0000_0000 <div style="display: flex; justify-content: space-between; align-items: center;"> <span>bit 1 <math>\uparrow</math></span> <span><math>\uparrow</math> bit 32</span> </div> TAS.5: *0 (bit 5 of status register)		
See Also	[ AS ], [ ASX ], DSTALL, ESTALL, GOBUF, GOWHEN, HOM, JOG, JOY, MA, MC, SMPER, STRGTD, STRGTE, STRGTT, STRGTV, TASF, TASX, TSTAT		

The TAS command returns the current status of the axis.

### FULL-TEXT STATUS REPORT AVAILABLE

The TAS status command reports a binary bit report. If you would like to see a more descriptive text-based report, use the TASF command description.

Bit # (left to right)	Function (1 = Yes; 0 = No)	GT6K	GV6K	To Clear This Bit:
1	Motion is <u>commanded</u> . (CAUTION: This bit could be cleared even while the motor is still "moving" – e.g., due to end-of-move settling.)	X	X	-----
2	Negative/positive-direction (1 = negative, 0 = positive).	X	X	-----
3	Accelerating (commanded acceleration only – TACC). This bit does not indicate commanded deceleration (bit is set to 0 during decel); to check if the axis is decelerating, the state of TAS bits 1, 3 and 4 should be: TAS1x00.	X	X	-----
4	At commanded velocity (TVEL).	X	X	-----
5	Home Successful (HOM) (YES/NO)	X	X	-----
6	Absolute/Incremental (MA) position mode selected. (1 = MA1, 0 = MA0)	X	X	-----
7	Continuous/Preset (MC) position mode selected. (1 = MC1, 0 = MC0)	X	X	-----
8	Jog Mode selected (JOG) (1 = JOG1, 0 = JOG0)	--	--	-----

9	Joystick Mode selected (JOY) (1 = JOY1, 0 = JOY0)	--	--	-----
10	RESERVED	--	--	-----
11	Position Maintenance Mode	X	--	-----
12	Stall Detected (ESTALL)	X	--	-----
13	Drive shut down. <b>NOTE:</b> If operating in the FLTDSB1 mode, a shutdown (DRIVE0 or open the Enable input interlock) will cause a "fault condition" – see note * below.	X	X	DRIVE1
14 *	Drive Faults occurred. Check the TASX response to identify which fault(s) occurred.	X	X	DRIVE1
15	Positive-direction hardware limit hit. This is <u>not</u> a "fault condition".	X	X	LH0 or GO in opposite direction.
16	Negative-direction hardware limit hit. This is <u>not</u> a "fault condition".	X	X	LH0 or GO in opposite direction.
17	Positive-direction Software Limit (LSPOS) Hit.	X	X	LS0 or GO in opposite direction.
18	Negative-direction Software Limit (LSNEG) Hit.	X	X	LS0 or GO in opposite direction.
19	Position Error	X	--	-----
20	RESERVED	--	--	-----
21	RESERVED	--	--	-----
22	RESERVED	--	--	-----
23 *	Position error exceeded (SMPEP).	--	X	DRIVE1
24	In Target Zone (defined with STRGTD & STRGTV). This bit is set only after the <i>successful completion</i> of a move (if STRGTD and STRGTV criteria have been satisfied). This bit is usable even if the Target Zone mode is not enabled (STRGTE0).	--	X	GO1
25	Target Zone Timeout occurred (STRGTT).	--	X	GO1
26	Change in motion is suspended pending GOWHEN (YES/NO). This bit is cleared if the GOWHEN condition is true, or if STOP (!S) or KILL (!K or ^K) is executed	X	X	(see description)
27	RESERVED	--	--	-----
28	Registration move initiated by trigger since last GO command. This bit is cleared with the next GO command.	X	X	GO1
29	GOWHEN error occurred. The input state or the position relationship specified in a GOWHEN command was already true when the subsequent GO, FGADV, FSHFC or FSHFD command was executed.	X	X	-----
30	Pre-emptive (OTF) GO or Registration profile not possible	X	X	GO1
31	Compiled GOBUF profile is executing.	X	X	-----
32	RESERVED	--	--	-----

\* **FAULT CONDITIONS:** If one or more of these conditions exist, the drive automatically disables (DRIVE0), it activates any output configured as a fault output, and it opens the dry contact relay (labeled "RELAY COM" and "RELAY N.O.") on the 4 pin removable connector.



---

## TASF Transfer Axis Status (full-text report)

Type	Transfer	Product	Rev
Syntax	<a_><!>TASF	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TASF: (see example below)		
See Also	[ AS ], [ ASX ], DSTALL, ESTALL, GOBUF, GOWHEN, HOM, JOG, JOY, MA, MC, SMPER, STRGTD, STRGTE, STRGTT, STRGTV, TAS, TASX, TSTAT		

---

The TASF command returns a text-based status report. This is an alternative to the binary report (TAS).  
Example TASF response:

```
*TASF          AXIS #
*              1
*Moving        NO
*Direction NEG NO
*Accelerating  NO
*At Velocity   NO
*
*Home successful NO
*Mode Absolute NO
*Mode Continuous NO
*Jog Mode      NO
*
*Joystick Mode NO
*RESERVED     NO
*Position Maint Mode NO (RESERVED for GV6K)
*Stall Detected NO (RESERVED for GV6K)
*
*Drive Shutdown NO
*Drive Faulted NO
*POS Hard Limit Hit NO
*NEG Hard Limit Hit NO
*
*POS Sftwr Limit Hit NO
*NEG Sftwr Limit Hit NO
*Within Deadband NO (RESERVED for GV6K)
*RESERVED     NO
*
*RESERVED     NO
*RESERVED     NO
*Pos Error Exceeded NO (RESERVED for GT6K)
*In Target Zone YES (RESERVED for GT6K)
*
*Target Zone Timeout NO (RESERVED for GT6K)
*Gowhen is Pending NO
*RESERVED     NO
*Reg Move Commanded NO
*
*Gowhen Error   NO
*Preset Move Overshot NO
*Executing Profile NO
```

## [ TASK ] Task Number Assignment

Type	Assignment or Comparison	Product	Rev
Syntax	See below	GT6K	6.0
Units	The TASK value is the number of the controlling task.	GV6K	6.0
Range	0-10		
Default	n/a		
Response	n/a		
See Also	%, TTASK, VAR, VARI		

The TASK operator allows the program itself to determine which task is executing it. The current task number TASK may be assigned to a numeric or integer variable or evaluated in a conditional statement, such as IF or WAIT.

**Syntax:** VARn=TASK or VARIn=TASK where “n” is the variable number; or TASK can be used in an expression, such as IF(TASK=3).

The TASK operator allows a single program to be used as a subroutine called from programs running in all tasks, yet this routine could contain sections of statements which are executed by some tasks and not others. The example below demonstrates statements used to execute different WAIT-for-input conditions, depending on the task that is executing the program.

**Example:**

```
IF (TASK=1)           ; Check if this program is operating in task 1  
WAIT(1IN.3=B1)       ; If in task 1, wait for input at location 3 on I/O brick 1  
NIF  
  
IF(TASK=2)           ; Check if this program is operating in task 2  
WAIT(2IN.11=B1)      ; If in task 2, wait for input at location 11 on I/O brick 2  
NIF
```

## TASX Transfer Extended Axis Status

Type	Transfer	Product	Rev
Syntax	<a_><!>TASX<.i>	GT6K	6.0
Units	i = bit location for the axis (See below)	GV6K	6.0
Range	1-32		
Default	n/a		
Response	TASX: *1TASX0000_0000_0000_0000_0000_0000_0000_0000 bit 1     ↑  ↑     bit 32 TASX.5: *0 (bit 5 of status register)		
See Also	[AS], [ASX], DCLRLR, DRIVE, DSTALL, [ER], TAS, TASXF, TER		

The TASX command reports axis status conditions.

<b>FULL-TEXT STATUS REPORT AVAILABLE</b>
--

The TASX status command reports a binary bit report. If you would like to see a more descriptive text-based report, use the TASXF command description.

Bit #	Function (1 = Yes; 0 = No)	GT6K	GV6K	To Clear This Status Bit:
1 *	Motor temperature fault (hardware switch in the motor).	X	X	DRIVE1
2 *	Low voltage fault	X	X	DRIVE1
3 *	Drive over-temperature fault (hardware thermal sensor in drive).	X	X	DRIVE1
4	RESERVED	--	--	-----
5 *	Resolver failure.	--	X	RESET or cycle power
6	RESERVED	--	--	-----
7 *	Motor configuration error occurred (checked on power up). Use TCS to ascertain the cause of the error. This is a fault condition which causes a drive shutdown (DRIVE0).	X	X	Resolve error condition (see TCS) and issue RESET or DRESET or cycle power
8	RESERVED	--	--	-----
9 *	Velocity error limit (SMVER value) has been exceeded.	--	X	DRIVE1
10 *	Bridge fault (hardware signal from the bridge)	X	X	RESET or DRESET or cycle power
11 *	Bridge temperature fault (this is a software control).	--	X	DRIVE1
12 *	Over-voltage.	--	X	RESET or DRESET or cycle power
13-16	RESERVED	--	--	-----
17	Stall detected (DSTALL or ESTALL)	X	--	DRIVE1
18	Override mode was invoked. (See DMVLIM command.)	X	X	DCLRLR or RESET or DRESET or cycle power
19	Bridge is in foldback mode.	--	X	DCLRLR or RESET or DRESET or cycle power
20	Power Dissipation Circuit has been active – not applicable to GV6K-U3, GV6K-U6, and GV6K-U12 drives	X	X	DCLRLR or RESET or cycle power
21 *	Bad Hall state detected. (Use THALL for diagnostics.)	--	X	RESET or cycle power
22 *	Unrecognized hardware — consult factory	X	X	RESET or cycle power
23 *	User Fault input activated (INFNCi-F)	X	X	Deactivate the input
24	Keep Alive active (User supplied +24VDC)	X	X	-----
25	Power dissipation circuit fault (excessive power dissipation)	X	X	DRIVE1
26	RESERVED	--	--	-----
27	RESERVED	--	--	-----
28	Motor configuration warning resulting from trying to run the motor beyond acceptable limits. Use TCS to ascertain the cause of the warning.	X	X	Resolve error condition (see TCS) and issue RESET or DRESET or cycle power
29 *	ORES failure (encoder output or step & direction output exceeds the maximum output frequency).	X	X	DRIVE1
30 *	Motor Thermal model fault.	--	X	Let the motor cool, then issue DRIVE1.
31	Commanded torque/force is at limit (TTRQ = DMTLIM).	--	X	DCLRLR or RESET or DRESET or cycle power
32	RESERVED	--	--	-----

\* **FAULT CONDITIONS:** If one or more of these conditions exist, the drive automatically disables (DRIVE0), it activates any output configured as a fault output, and it opens the dry contact relay (labeled “RELAY COM” and “RELAY N.O.”) on the 4 pin removable connector.

---

## TASXF Transfer Extended Axis Status, (full-text report)

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TASXF	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TASXF: (see example below)		
See Also	[AS], [ASX], DSTALL, [ER], TAS, TASX, TER		

---

The TASXF command returns a text-based status report. This is an alternative to the binary report (TASX).  
Example TASXF response:

```
*TASXF                AXIS #
*                      1
*Motor Fault          NO
*Low-voltage          NO
*Over-temperature     NO
*RESERVED             NO
*
*Resolver Failure     NO (RESERVED for GT6K)
*RESERVED             NO
*Motor Config Error   NO
*RESERVED             NO
*
*SMVER Exceeded       NO (RESERVED for GT6K)
*Bridge Fault         NO
*Bridge Temp Fault    NO (RESERVED for GT6K)
*Over-voltage         NO (RESERVED for GT6K)
*
*RESERVED             NO
*RESERVED             NO
*RESERVED             NO
*RESERVED             NO
*
*Drive Stall Detected NO (RESERVED for GV6K)
*Override Mode Active NO
*Bridge in Foldback   NO (RESERVED for GT6K)
*Power Dissip Active  NO
*
*Bad Hall State       NO (RESERVED for GT6K)
*Unrecognized Hardware NO
*User Fault           NO
*Keep Alive Active    NO
*
*Power Dissipat Fault NO
*RESERVED             NO
*RESERVED             NO
*Motor Config Warning NO
*
*ORES Failure         NO
*Motor Thermal Fault  NO (RESERVED for GT6K)
*Torque at Limit      NO (RESERVED for GT6K)
*RESERVED             NO
```

---

## TCMDER      Transfer Command Error

Type	Transfer or Program Debug Tool	Product	Rev
Syntax	<a_><!>TCMDER	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TCMDER: *(incorrect command)		
See Also	ERRBAD, [ SS ], TSS		

---

For program debugging, use TCMDER to transfer the command that the drive 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 drive 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 drive 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 system status bit #11 (reported in the TSSF, TSS, and SS commands) is set to 1. The status bit remains set until the TCMDER command is issued.

### Example:

```
DEL badprg           ; Delete program called badprg
DEF badprg           ; Begin definition of program called badprg
MA1                  ; Select the absolute preset positioning mode
A25                  ; Set acceleration
AD11                 ; Set deceleration
V5                   ; Set velocity
VAR1=0               ; Set variable #1 equal to zero
GO1                  ; Initiate move
IF(VAR1<)16          ; Mistyped IF statement--should be typed as: IF(VAR1<16)
VAR1=VAR1+1          ; If variable #1 is less than 16, increment the counter by 1
NIF                  ; End IF statement
END                  ; End programming of program called badprg
RUN badprg           ; Run the program called badprg
                     ; (this will cause an error --see comment box below)
; *****
; * 1. When you run the badprg program, you should see this error *
; * message on your screen: "*INCORRECT DATA" (this error message *
; * indicates incorrect command syntax). *
; * 2. Type "TCMDER" and press enter. This queries the drive to *
; * display the command that caused the error. In this case, the *
; * response will be "*IF(VAR1<)16". *
; *****
```

## TCS Transfer Configuration Status

Type	Transfer	Product	Rev
Syntax	<a_><!>TCS	GT6K	6.0
Units	Fault/Warning code (see table below)	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TCS: *TCS46		
See Also	DMTR, TASX		

TASX bit #7 is set when a motor configuration error occurs. TASX bit #28 is set when a motor configuration warning occurs. An error causes the drive to be shut down (DRIVE0). A warning means the drive attempted to control the motor outside of safe operating limits (in this case, the maximum safe configuration value is used). To help ascertain the cause of the error or warning, the TCS command reports any existing configuration error or warning conditions (refer to the table below).

**NOTE:** TCS reports only one code. If there is more than one error or warning condition present, **errors will overwrite warnings**. Therefore, to resolve multiple error or warning conditions:

1. Resolve the known error.
2. Cycle power to the drive, issue a RESET command, or activate the Reset input.
3. If another error condition presents itself (e.g., the drive will not enable), check for subsequent errors with the TCS command.

Code	Fault / Warning	Drive Type	Condition	Method to clear	TASX Bit Set
-32228	Fault	GV6K	ERES $\neq$ 4096 during SFB4	Change ERES to 4096 (ERES4096).	Bit 7
-32238	Fault	GV6K	Internal position loop gains $\leq$ 0.	Increase one or all of the following: DPBW, SGPRAT, SGPSIG. Recheck DMTKE, DMTJ, LJRAT, DMTD, LDAMP.	Bit 7
-32248	Fault	GV6K	Internal velocity loop gains $\leq$ 0.	Increase SGVRAT. Recheck DMTKE, DMTJ, LJRAT, DMTD, DVBW, LDAMP.	Bit 7
-32259	Fault	GV6K	Internal current loop gains $<$ 0.	Increase DIBW and/or re-check DMTRES, DMTLMN, DMTLMX and SGIRAT.	Bit 7
-32367	Fault	GT6K	Drive resolution (DRES) is too low for the number of motor pole pairs.	Increase DRES value to be at least 4 X DPOLE.	Bit 7
-32710	Fault	Both	DMTJ = 0	Enter non zero value.	Bit 7
-32714	Fault	GV6K	DMTLMX = 0	Enter non zero value.	Bit 7
-32715	Fault	GV6K	DMTLMN = 0	Enter non zero value.	Bit 7
-32718	Fault	GV6K	DMTW = 0	Enter non zero value.	Bit 7
-32723	Fault	Both	DPOLE = 0	Enter non zero value.	Bit 7
-32725	Fault	Both	DMTRES = 0	Enter non zero value.	Bit 7
-32726	Fault	GT6K	DMTIND = 0	Enter non zero value.	Bit 7
-32727	Fault	GV6K	DMTKE = 0	Enter non zero value.	Bit 7
-32729	Fault	GT6K	DMTSTT = 0	Enter non zero value.	Bit 7
40	Warning	Both	DMTIC = 0	Enter non zero value.	Bit 28
46	Warning	GT6K	DIGNA = 0	Enter non zero value.	Bit 28
47	Warning	GT6K	DIGNB = 0	Enter non zero value.	Bit 28
48	Warning	GT6K	DIGNC = 0	Enter non zero value.	Bit 28
49	Warning	GT6K	DIGND = 0	Enter non zero value.	Bit 28
51	Warning	GV6K	DMTIP = 0	Enter non zero value.	Bit 28

Code	Fault / Warning	Drive Type	Condition	Method to clear	TASX Bit Set
400	Warning	Both	DMTIC is too high for power level of drive. The drive's continuous current rating is used instead.	Is drive sized properly? Lower DMTIC.	Bit 28
500	Warning	GV6K	DMTLIM is too high for the drive's peak current rating.	Lower DMTLIM and/or recheck DMTKE.	Bit 28
503	Warning	GV6K	DMTIP is set too high. Drive is set to maximum value.	Is drive sized properly? Lower DMTIP.	Bit 28
551	Warning	GV6K	Notch filter calculation error.	Check DNOTAF, DNOTBF, DNOTAQ, DNOTBQ.	Bit 28
560	Warning	GV6K	Lead frequency (DNOTLD) < 20 Hz. The last good value is used. (0 = disable)	Increase DNOTLD.	Bit 28
561	Warning	GV6K	Lead frequency (DNOTLD) > 4 times the lag frequency (DNOTLG). The last good value is used. (0 = disable)	Lower DNOTLD or turn off Lead Filter (DNOTLD = 0).	Bit 28

---

## TDHRS      Transfer Operating Hours

Type	Transfer	Product	Rev
Syntax	<a_><!>TDHRS	GT6K	6.0
Units	Lifetime operating hours (resolution is ¼ or 0.25 hours)	GV6K	6.0
Range	Hour counter rolls over at 16384.00 hours		
Default	n/a		
Response	TDHRS:    *TDHRS16.5		
See Also	RESET, TERRLG		

---

The TDHRS command reports the lifetime number of hours (to the nearest ¼ hour) that the Gem6K drive has had power applied.

**NOTE:** The hour count rolls over at 16,384.00 hours.

---

## TDICNT      Transfer Continuous Current Rating

Type	Transfer	Product	Rev
Syntax	<a_><!>TDICNT	GT6K	6.0
Units	Amps peak	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TDICNT: *TDICNT10		
See Also	DMTIC, DMTIP, TDIMAX		

---

The TDICNT command reports the continuous current rating of the drive in amps peak. Note that most other current-related parameters (e.g., DMTIC, DMTIP) are in amps RMS.

---

## TDIMAX      Transfer Maximum Current Rating

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TDIMAX	GT6K	n/a
Units	Amps peak	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TDIMAX:    *TDIMAX10		(applicable only to servo axes)
See Also	DMTIP, DMTIP, TDICNT		

---

The TDIMAX command reports the maximum current rating of the drive in amps peak.

Note that most other current-related parameters (e.g., DMTIC, DMTIP) are in amps RMS.

---

## TDIR              Transfer Program Directory

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TDIR	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TDIR:    *NO PROGRAMS DEFINED *150000 OF 150000 BYTES (100%) PROGRAM MEMORY REMAINING *1973 of 1973 SEGMENTS (100%) COMPILED MEMORY REMAINING		
See Also	DEF, INFNC, MEMORY, PLCP, [ SEG ] TMEM, TSEG		

---

TDIR 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
*14322 OF 150000 BYTES (98%) PROGRAM MEMORY REMAINING
*1973 of 1973 SEGMENTS (100%) COMPILED MEMORY REMAINING
```

(In the above example, PROG1 and PROG2 are names of programs.)

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 intended to be a compiled profile and has been successfully compiled (PCOMP), then the line item for a compiled GOBUF program is amended with "COMPILED AS A PATH", and the line item for a compiled PLCP program is "COMPILED AS A PLC PROGRAM."



---

## **TDPTR**      **Transfer Data Pointer Status**

Type	Data Storage	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TDPTR	GT6K	6.0
Units	n/a	GV6K	6.0
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**

```
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.
```

---

## **TDTEMP**      **Transfer Drive Temperature**

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TDTEMP	GT6K	6.0
Units	Degrees C	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TDTEMP:      *TDTEMP50		
See Also	DMONAV, DMONBV, TERRLG		

---

The TDTEMP reports the measured temperature (internal) of the drive.

---

## **TDVBUS**      **Transfer Bus Voltage**

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TDVBUS	GT6K	6.0
Units	Volts	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TDVBUS:      *TDVBUS170		
See Also			

---

**GT6K:** TDVBUS reports the currently measured bus voltage of the drive.

**GV6K:** TDVBUS reports the nominal DC bus voltage (see calculation below) when the drive is first powered up.

$$169V (= \sqrt{2} * 120) \quad \text{or} \quad 339V (= \sqrt{2} * 240)$$

## TER Transfer Error Status

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!><%>TER<.i>	GT6K	6.0
Units	i = specific error status bit (specific to the task)	GV6K	6.0
Range	1-32		
Default	n/a		
Response	TER: *TER0000_0000_0000_0000_0000_0000_0000_0000 <div style="text-align: center;"> <span style="margin-right: 100px;">bit 1</span> <span style="margin-left: 100px;">bit 32</span> </div> TER.4: *0 (bit 4 of error status register for Task 0)		
See Also	[ ASX ], DSTALL, [ ER ], ERROR, ERRORP, ESTALL, GOWHEN, INFNC, LH, LIMFNC, LS, SMPER, STRGTT, TASX, TCMER, TERF		

The TER command returns the status of the 32 error bits. There is one error status register per Task. The TER status command reports a binary bit report. If you would like to see a more descriptive text-based report, use the TERF command description.

### NOTES

- 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.
- **Multi-tasking:** If you are using multi-tasking, be aware that each task has its own error status register. Therefore, to check a specific task's error status, you must prefix the TER command with the task identifier (e.g., 2%TER to check error status for Task 2). If no task identifier is given, the TER response is for the task supervisor (Task 0).

The function of each error status bit is shown below.

Bit # (left to right)	Function (1 = Yes; 0 = No)	GT6K	GV6K	To Clear This Bit:
1	Stall detected: Functions when stall detection has been enabled (ESTALL or DSTALL), and Kill on Stall is enabled (ESK1).	X	--	GO
2	Hardware end-of-travel limit hit. This condition is detectable only when the limits are enabled (LH3).	X	X	LH0 or GO in opposite direction.
3	Software end-of-travel limit hit. This condition is detectable only when the limits are enabled (LS3).	X	X	LS0 or GO in opposite direction.
4	Drive fault: Detected only if the drive is enabled (DRIVE).	X	X	Varies with Fault – refer to TASX
5	RESERVED (see note at end of table; refer to ERROR command)	X	X	-----
6	Kill input activated. An input defined as a Kill input (INFNCi-C) was activated.	X	X	Deactivate input
7	User Fault input activated. Input must be assigned the user fault function (INFNCi-F).	X	X	Deactivate input
8	Stop input activated. An input defined as a Stop input (INFNCi-D) was activated.	X	X	Deactivate input
9	Enable input is not grounded.	X	X	Ground the Enable Input
10	Pre-emptive (on-the-fly) GO or registration move profile not possible.	X	X	Issue another GO command
11	Target Zone Settling Timeout Period (set with the STRGTT command) is exceeded.	--	X	STRGTE0 : D0 : GO : STRGTE1
12	Maximum position error (set with the SMPER command) is exceeded.	--	X	DRIVE1
13	RESERVED	--	--	-----
14	GOWHEN condition already true when a subsequent GO, FSHFC, or FSHFD is executed.	X	X	-----
15	RESERVED	--	--	RESERVED
16	Bad command detected	--	X	TCMTER

17	RESERVED	--	--	RESERVED
18	Cable to an expansion I/O brick is disconnected, or power to the I/O brick is lost. (Multi-tasking: This condition also kills all tasks.)	X	X	Reconnect I/O brick, issue ERROR.18-0, then ERROR.18-1
19	RESERVED	--	--	-----
20	RESERVED	--	--	-----
21	RESERVED	--	--	-----
22	RESERVED	--	--	-----
23	Ethernet Client connection error.	X	X	Clear error bit (ERROR.23-0), then re-establish Ethernet connection, and issue ERROR.23-1
24	Ethernet Client polling error.	X	X	Clear error bit (ERROR.24-0), then re-establish Ethernet connection, and issue ERROR.24-1
25-32	RESERVED	--	--	-----

**Error Handling:** Each TER status bit has a corresponding error-checking bit that can be enabled with the ERROR command. If an error-checking bit is enabled and the error occurs, the Gem6K drive will branch to the “error program,” which is assigned with the ERRORP command. For additional details on handling errors, refer to the ERROR and ERRORP command descriptions, and the *Error Handling* section in the *Programmer’s Guide*.

**NOTE:** 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 drive to GOSUB or GOTO to the error program (ERRORP). Within the error program the cause of the error will need to be determined. The 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.

---

## TERF Transfer Error Status (full-text report)

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!><%>TERF	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TERF: (see example below)		
See Also	[ ASX ], DSTALL, [ ER ], ERROR, ERRORP, ESTALL, GOWHEN, INFNC, LH, LS, SMPER, STRGTT, TASX, TCMDER, TER		

---

The TERF command returns a text-based status report. This is an alternative to the binary report (TER). Example TERF response:

```
*TERF
*Stall Detected      NO
*Hard Limit Hit     NO
*Soft Limit Hit     NO
*Drive Fault Active NO
*
*RESERVED           NO
*Kill Input Active  NO
*User Fault Input   NO
*Stop Input Asserted NO
*
*Enable Input Open  NO
*Profile Impossible NO
*Target Zone Timeout NO
*Max Position Error NO
*
*RESERVED           NO
*GOWHEN cond true   NO
*RESERVED           NO
*Bad command        NO
*
*RESERVED           NO
*I/O Brick Failure  NO
*RESERVED           NO
*RESERVED           NO
*
*RESERVED           NO
*RESERVED           NO
*Client Connect Err NO
*Client Polling Err NO
```

---

## TERRLG      Transfer Error Log

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TERRLG	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TERRLG:    *TAS0000_0000_0000_0000_0000_0000_0000_0000 *TASX0000_0000_0000_0000_0000_0000_0000_0000 *TDHRS104.5 *TDTEMP45 *TMTEMP100		
See Also	CERRLG, ERRORL, TAS, TASX, TDHRS, TDTEMP, TMTEMP		

---

The error log is updated every time an event occurs, as selected by the ERRORL command. The TERRLG command displays the last ten error conditions, most recent at the top, which the drive has experienced, as recorded in these status registers:

- TAS (axis status binary report)
- TASX (extended axis status binary report)
- TDHRS (number of lifetime hours the drive has been powered)
- TDTEMP (temperature of the drive in degrees centigrade)
- TMTEMP (temperature of the motor in degrees centigrade – GV6K only)

**NOTE:** After an error event occurs and the drive updates the error log, bits may still be recorded in TAS or TASX for another 120 milliseconds. To find all possible causes of an error event, examine TAS and TASX in addition to viewing the error log.

The CERRLG command erases the stored contents of the error log. Clearing the error log is a helpful diagnostic tool; it allows you to start the diagnostic process when the error log is in a known state so that you can check the error log in response to subsequent events.

**NOTE:** TERRLG may not be stored in a program.

---

## TEX              Transfer Program Execution Status

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!><%>TEX	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	!TEX:        *PROGRAM NOT EXECUTING		
See Also	DEF		

---

The TEX command reports the status of any programs in progress (in the specified task). If using multi-tasking, you must prefix the TEX with the task you want to check (e.g., 2%TEX).

If the program PAUL was in progress (in task 0), 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 Device

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TFB	GT6K	n/a
Units	Response is position of the selected feedback device	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TFB *TFB+0	(applicable only to servo axes)	
See Also	[ FB ], [ PE ], PSET, SCALE, SCLD, SFB, TPCE, TPE		

---

Use the TFB command to return the value of the feedback source selected with the SFB command. If you do not change the default SFB selection, the response will indicate the encoder position.

If scaling is **not** enabled, the position values returned will be counts. If scaling is enabled (SCALE1), the 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:

```
SFB1 ; Select encoder feedback
TFB ; Report encoder's position.
      ; Sample response is *TFB2.436
```

---

## TFS Transfer Following Status

Type	Following; Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TFS	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TFS *TFS0000_0000_0000_0000_0000_0000_0000_0000 bit 1 ← ↑ bit 32		
See Also	FGADV, FMAXA, FMAXV, FMCLN, FMCP, FOLEN, FOLMAS, FPPEN, [ FS ], FSHFC, FSHFD, MC, [ NMCY ], [ PMAS ], TFSF		

---

The TFS command returns the current Following status. The response for TFS is as follows:

**FULL-TEXT STATUS REPORT AVAILABLE**

The TFS status command reports a binary bit report. If you would like to see a more descriptive text-based report, use the TFSF command description.

---

---

### Bit Assignment

(left to right)	Function (YES = 1; NO = ∅)
1	Follower in Ratio Move      A Following move is in progress.
2	Ratio is Negative            The current ratio is negative (i.e., the follower counts are counting in the opposite direction from the master counts).
3	Follower Ratio Changing     The follower is ramping from one ratio to another (including a ramp to or from zero ratio).
4	Follower At Ratio            The follower is at constant non-zero ratio.

Bits 1-4 indicate the status of Following motion. They mimic the meaning and organization of Axis Status (TAS & AS) bits 1-4, except that each bit indicates the current state of the ratio, rather than the current state of the velocity.

---

\* 5      FOLMAS Active      A master is specified with the FOLMAS command.

---

* 6	FOLEN Active	Following has been enabled with the FOLEN command.
* 7	Master is Moving	The specified master is currently in motion.
8	Master Dir Neg	The current master direction is negative. (bit must be cleared to allow Following move in preset mode—MCØ).

Bits 5-8 indicate the status required for Following motion (i.e., a master must be assigned, Following must be enabled, the master must be moving, and for many features, the master direction must be positive).

Unless the master is a commanded position of another axis, it is likely that minor vibration of the master will cause bits 7-8 to toggle on and off, even if the master is nominally "at rest". These bits are meant primarily as a quick diagnosis for the absence of master motion, or master motion in the wrong direction. Many features require positive master counting to work properly.

9	OK to Shift	Conditions are valid to issue shift commands (FSHFD or FSHFC).
10	Shifting now	A shift move is in progress.
11	Shift is Continuous	An FSHFC-based shift move is in progress.
12	Shift Dir is Neg	The direction of the shift move in progress is negative.

Bits 9-12 indicate the shift status of the follower. Shifting is super-imposed motion, but if viewed alone, can have its own status. In other words, bits 10-12 describe only the shifting portion of motion.

13	Master Cyc Trig Pend	A master cycle restart is pending the occurrence of the specified trigger.
14	Mas Cyc Len Given	A non-zero master cycle length has been specified with the FMCLEN command.
15	Master Cyc Pos Neg	The current master cycle position (PMAS) is negative. This could be caused by a negative initial master cycle position (FMCP), or if the master is moving in the negative direction.
16	Master Cyc Num > 0	The master position (PMAS) has exceeded the master cycle length (FMCLEN) at least once, causing the master cycle number (NMCY) to increment.

Bits 13-16 indicate the status of master cycle counting. If a Following application is taking advantage of master cycle counting, these bits provide a quick summary of some important master cycle information.

17	Mas Pos Prediction On	Master position prediction has been enabled (FPPEN).
18	Mas Filtering On	A non-zero value for master position filtering (FFILT) is in effect.
19	RESERVED	
20	RESERVED	

Bits 17-18 indicate the status of master position measurement features.

21	RESERVED	
22	RESERVED	
23	OK to do FGADV move	OK to do Geared Advance move (master assigned with FOLMAS, Following enabled with FOLEN, and follower axis is either not moving, or moving at constant ratio in continuous mode).
24	FGADV move underway	Geared Advance move profile is in progress.

Bits 23-24 indicate the status of geared advance features.

25	RESERVED	
26	FMAXA/FMAXV limited	The present Following move profile is being limited by FMAXA or FMAXV.
27-32	RESERVED	

\* All these conditions must be true before Following motion will occur.

---

## TFSF                      Transfer Following Status (full-text report)

Type	Following; Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TFSF	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TFSF:    (see example below)		
See Also	FGADV, FMAXA, FMAXV, FMCLN, FMCP, FOLEN, FOLMAS, FPPEN, [ FS ], FSHFC, FSHFD, MC, [ NMCY ], [ PMAS ], TFS		

---

The TFSF command returns a text-based status report. This is an alternative to the binary report (TFS).

Example TFSF response:

```
*TFSF                      AXIS #
*                            1
*Follower in Ratio Move    NO
*Ratio is Negative        NO
*Followr Ratio Changing   NO
*Follower At Ratio        NO
*
*FOLMAS Active            NO
*FOLEN Active            NO
*Master is Moving        NO
*Master Dir Neg          NO
*
*OK to Shift              NO
*Shifting now            NO
*Shift is Continuous     NO
*Shift Dir is Neg        NO
*
*Master Cyc Trig Arm     NO
*Mas Cyc Len Given      NO
*Master Cyc Pos Neg     NO
*Master Cyc Num > 0     NO
*
*Pos Prediction On        YES
*Master Filtering On     NO
*Performing FRC move    NO
*RESERVED                NO
*
*Master Window given     NO
*Inside Master Window    NO
*OK to do FGADV move    NO
*FGADV move underway    NO
*
*RESERVED                NO
*FMAXA/FMAXV limited    NO
*RESERVED                NO
```



---

## TGAIN      Transfer Active Gains

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TGAIN	GT6K	n/a
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	(list of all gain values)		(applicable only to servo axes)
See Also	SGENB, SGSET, TSGSET, (see list below)		

---

This command allows you to display the value of each of the gains presently in effect (see list below). Each time an individual gain is entered, the TGAIN register is updated accordingly. When a gain set is enabled with the SGENB command, the present value of each gain is set to the values saved in that particular gain set.

- DIBW (current loop bandwidth)
- DMTLIM (torque/force limit)
- DMVLIM (velocity limit)
- DNOTAD (notch filter A depth)
- DNOTAF (notch filter A frequency)
- DNOTAQ (notch filter A quality factor)
- DNOTBD (notch filter B depth)
- DNOTBF (notch filter B frequency)
- DNOTBQ (notch filter A quality factor)
- DNOTLD (notch lead filter break frequency)
- DNOTLG (notch lag filter break frequency)
- DPBW (position loop bandwidth)
- LDAMP (load damping)
- LJRAT (load-to-rotor inertia ratio or load-to-force mass ratio)
- SGAF (acceleration feedforward gain)
- SGINTE (integrator enable)
- SGIRAT (current damping ratio)
- SGPRAT (position loop ratio)
- SGPSIG (velocity/position bandwidth ratio)
- SGVF (velocity feedforward gain)
- SGVRAT (velocity damping ratio)

---

## THALL      Transfer Hall Sensor Values

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>THALL	GT6K	n/a
Units	n/a	GV6K	6.0
Range	1-6 (0 or 7 is a fault condition - see TASX bit #21)		
Default	n/a		
Response	THALL: *THALL6		(applicable only to servo axes)
See Also	SHALL, TASX		

---

**Encoder Motors (with SFB1):** The THALL command reports the present Hall sensor value. There are six distinct Hall states, from 1 to 6. Rotating the motor shaft clockwise, the Hall state order should be 6, 2, 3, 1, 5, 4, 6, 2, 3, 1, 5, 4, 6 (and so on).

THALL values 0 and 7 are invalid and will fault the drive and set TASX bit #21 (in case of this fault, check the Hall wiring or grounding/noise conditions).

For a complete description on how to troubleshoot Hall sensors, especially for non-Compumotor motors, refer to the *Hardware Installation Guide* section on using non-Compumotor motors.

**Resolver Motors (with SFB4):** If you use a resolver motor, THALL will always report a value of 7.

---

## [ TIM ] Current Timer Value

Type	Assignment or Comparison	<b>Product</b>	<b>Rev</b>
Syntax	See below	GT6K	6.0
Units	Milliseconds	GV6K	6.0
Range	Maximum count is 999,999,999 (approx. 11 days, 13 hours)		
Default	n/a		
Response	n/a		
See Also	TIMINT, TIMST, TIMSTP, TTIM		

---

The 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=<n%>TIM where x is a numeric variable number, or TIM can be used in an expression such as IF(TIM<2400). Multi-tasking: If addressing the timer of a specific task, include the n% prefix.

**Example:**

```
VAR1=TIM           ; Timer value is assigned to variable 1
IF(TIM<1000)       ; If timer value is < 1000 milliseconds, do the IF statement
VAR1=TIM + 10      ; Timer value plus 10 assigned to variable 1
NIF                ; End IF statement
```

---

## TIMINT Timer Value to Cause Alarm Event

Type	Timer; Alarm Event	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TIMINT<b>,<i>	GT6K	6.0
Units	i = milliseconds	GV6K	6.0
Range	b = 0 (reset and start) or 1 (stop) i = 0 - 999,999,999		
Default	0,0		
Response	TIMINT: *TIMINT0,0		
See Also	INTHW, [ TIM ], TIMST, TIMSTP, TTIM		

---

The TIMINT command sets the timer value upon which the Gem6K drive will trigger an Alarm Event. The time value at which the alarm event will occur is specified by the second field in the command.

<b>NOTES</b>
--------------

- To use TIMINT, you must first issue the INTHW.21-1 command to enable checking for the alarm event.
  - When using multi-tasking, this feature only works with the timer for Task zero.
- 

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 alarm value, a one should be specified for the first field. If the timer is to be reset and restarted upon reaching the alarm value, a zero should be specified for the first field. By specifying a zero in the first field, an alarm will occur repeatedly.

**Example:**

```
INTHW.21-1         ; Enable checking for the timer-driven alarm event
TIMINT1,10000      ; Trigger alarm once after 10000 ms, do not restart the timer
TIMST0             ; Reset and start timer
```

---

## TIMST

### Start Timer

Type	Timer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TIMST<b><r>	GT6K	6.0
Units	b = Enable bit r = time (milliseconds) if b = 0, task # if b = 1	GV6K	6.0
Range	b = 0 (reset and start) or 1 (start from previous TIMSTP) r = absolute time 0-999,999,999 if b = 0, or r = task # 0-10 if b = 1		
Default	0		
Response	TIMST: No response, acts as if TIMST1 command was issued		
See Also	[ TIM ], TIMINT, TIMSTP, TTIM		

---

The TIMST command is used to start the timer.

- If TIMST0, you can start the timer at a specific time in milliseconds (e.g., TIMST0, 500).
- If TIMST1, you can resume the timer (after stopping it with the TIMSTP command) with the value of the time of the specified task (e.g., TIMST1, 3).

The timer resolution is 2 ms. The delay for executing TIMST and TIMSTP in combination is 4-6 ms.

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.

**Multi-Tasking:** Each task has its own timer.

**Example:**

```
TIMST0      ; Reset and start timer
GO1         ; Initiate motion
TIMSTP     ; Stop timer
TTIM       ; Transfer time required for move
```

---

## TIMSTP

### Stop Timer

Type	Timer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!><%>TIMSTP	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	[ TIM ], TIMINT, TIMST, TTIM		

---

The 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.

The timer resolution is 2 ms. The delay for executing TIMST and TIMSTP in combination is 4-6 ms.

**Multi-Tasking:** Each task has its own timer.

**Example:**

```
TIMST0      ; Reset and start timer
GO1         ; Initiate motion
TIMSTP     ; Stop timer
TTIM       ; Transfer time required for move
```

## TIN Transfer Input Status

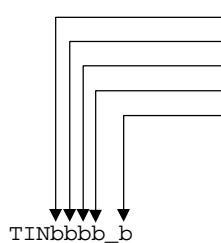
Type	Transfer	Product	Rev
Syntax	<a_><!><@><B>TIN<.i>	GT6K	6.0
Units	i = input number on the specified I/O brick (B) – see page 8	GV6K	6.0
Range	1-32 (dependent on I/O brick and options installed)		
Default	n/a		
Response	TIN: *0000_0 (onboard inputs) 1TIN *0000_0000_0000_0000_0000_0000_0000_0000 1TIN.4: *1 (status of I/O point 4 on I/O brick 1)		
See Also	IN, [ IN ], INFNC, INLVL, TINO, TLIM		

The TIN command returns the status (active or inactive) of the digital inputs on the DRIVE I/O connector and the programmable inputs on external I/O bricks. The inputs are numbered 1 to *n* from left to right (*n* is the maximum number of I/O points on the I/O brick— refer to page 8 for details.)

If the status of a specific input is required, use the bit select operator (.). For example, 1TIN.9 reports the status of the 1<sup>st</sup> I/O point on the 2<sup>nd</sup> SIM of I/O brick 1.

TIN response for inputs on the DRIVE I/O connector (bits are numbered 1-5 from left to right):

Input #	Pin #	GT6K & GV6K Function (INFNC default)
1	37	INFNC1-A (General-purpose. If assigned, TRG-A must be on Input #1)
2	38	INFNC2-A (General-purpose. If assigned, TRG-B must be on Input #2)
3	39	INFNC3-A (General-purpose)
4	34	INFNC4-A (General-purpose)
5	35	INFNC5-A (General-purpose. If assigned, MASTER TRIGGER must be on Input #5)



### Input State Relationships (inputs on the DRIVE I/O connector):

Active Level	Sinking/Sourcing *	Switch	IN/TIN/TIO Report
INLVL0 (active low)	Sourcing	Closed (connected to ground)	1
INLVL0 (active low)	Sourcing	Open	0
INLVL1 (active high)	Sourcing	Closed (connected to ground)	0
INLVL1 (active high)	Sourcing	Open	1
INLVL0 (active low)	Sinking	Closed (connected to +V)	0
INLVL0 (active low)	Sinking	Open	1
INLVL1 (active high)	Sinking	Closed (connected to +V)	1
INLVL1 (active high)	Sinking	Open	0

\* The inputs are factory configured to source current. If you wish the inputs to sink current, connect the pull-up terminals (pins 27 and 33) on the DRIVE I/O connector to ground (see the *Hardware Installation Guide* for wiring instructions). Pin 27 is the pull up for limit inputs 1-3, and pin 33 is the pull up for general-purpose inputs 1-5.

### Input State Relationships (external SIM8-IN-EVM32 inputs on an EVM32 I/O brick):

Active Level *	Jumper Selection *	Switch	Voltage at Input	LED	IN/TIN/TIO Report
INLVL0 (active low)	Position 3 (sourcing)	Open	≥ 2/3 of V+	On	0
INLVL0 (active low)	Position 3 (sourcing)	Closed	< 1/3 of V+	Off	1
INLVL1 (active high)	Position 3 (sourcing)	Open	≥ 2/3 of V+	On	1
INLVL1 (active high)	Position 3 (sourcing)	Closed	< 1/3 of V+	Off	0
INLVL0 (active low)	Position 1 (sinking)	Open	< 1/3 of V+	Off	1
INLVL0 (active low)	Position 1 (sinking)	Closed	≥ 2/3 of V+	On	0
INLVL1 (active high)	Position 1 (sinking)	Open	< 1/3 of V+	Off	0
INLVL1 (active high)	Position 1 (sinking)	Closed	≥ 2/3 of V+	On	1

\* Factory default: INLVL0 (active low) and jumper in position 3 (sourcing). Move jumper to position 1 for sinking.

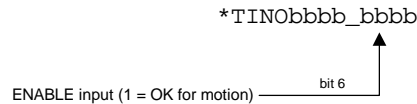
## TINO

### Transfer Other Input Status

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TINO<.i>	GT6K	6.0
Units	(bit #6 = ENABLE input. 1 = OK for motion)	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TINO: *TINO0000_0100 TINO.6: *1 (status of input number 6 - ENABLE input)		
See Also	[ INO ], TINOF		

The TINO command returns the status of the Enable input (bit #6). This bit is set (1) when the input is grounded and motion is allowable. All other TINO status bits (1-5 and 7-8) are not used and are always cleared (0). The Enable input is located at pin #1 on the DRIVE I/O connector.

TINO response (bits are numbered 1-8 from left to right):



#### FULL-TEXT STATUS REPORT AVAILABLE

The TINO status command reports a binary bit report. If you would like to see a more descriptive text-based report, use the TINOF command description.

## TINOF

### Transfer Other Input Status (full-text report)

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TINOF	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TINOF: (see example below)		
See Also	[ INO ], TINO		

The TINOF command returns a text-based status report. This is an alternative to the binary report (TINO).

Example response:

```
*TINOF
*Enable input OK   YES
```

## TIO

### Transfer Current Expansion I/O Status

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!><@><B>TIO	GT6K	6.0
Units	B = I/O brick number	GV6K	6.0
Range	1-8		
Default	n/a		
Response	(see example below)		
See Also	IN, KIOEN, TANO, TIN, TINO, TOUT, TANI		

The TIO command displays the status of the current I/O configuration for the drive's expansion I/O bricks. If an I/O brick is not connected, it will not be included in the status report. Onboard I/O is not reported.

The I/O bricks are connected in series to the "EXPANSION I/O" connector (see the *Hardware Installation Guide* for instructions). The 1<sup>st</sup> I/O brick in the series (closest to the Gem6K drive) is BRICK 1. The next is BRICK 2, and so on.

Each I/O brick has 4 SIM slots and can hold from 1 to 4 I/O SIM modules. A SIM slot may hold a digital input SIM, a digital output SIM, an analog input SIM, or an analog output SIM. Each SIM provides 8 inputs or outputs; therefore, each I/O brick has 32 I/O addresses, referenced as absolute I/O point locations:

- SIM slot 1 = I/O points 1-8
- SIM slot 2 = I/O points 9-16
- SIM slot 3 = I/O points 17-24
- SIM slot 4 = I/O points 25-32

The TIO response for each I/O brick is separated into four lines, one for each SIM. I/O points 1-8 represent SIM #1, 9-16 represents SIM #2, 17-24 represents SIM #3, and 25-32 represents SIM #4. When digital outputs are detected, the report also indicates the type (NPN outputs are SINKING, PNP outputs are SOURCING). When digital inputs and digital or relay outputs are detected, TIO displays the present hardware state and programmed function (INFNC for inputs and OUTFNC for outputs). When analog inputs are detected, TIO reports the voltage present on each input. When analog outputs are detected, TIO reports the voltage commanded on each output.

**Example** TIO responses (in this example, 2 I/O bricks are connected to the drive):

```

>TIO
*BRICK 1:  SIM Type      Status      Function
  1-8:  DIGITAL INPUTS  0000_0000  AAAA_AAAA
  9-16: DIGITAL INPUTS  0000_0000  AAAA_AAAA
 17-24: DIGITAL INPUTS  0000_0000  AAAA_AAAA
 25-32: ANALOG INPUTS   0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000
*BRICK 2:  SIM Type      Status      Function
  1-8:  DIGITAL OUTPUTS  0000_0000  AAAA_AAAA -- SINKING
  9-16: DIGITAL INPUTS  0000_0000  AAAA_AAAA
 17-24: NO SIM PRESENT
 25-32: DIGITAL OUTPUTS  0000_0000  AAAA_AAAA -- SOURCING

>1TIO
*BRICK 1:  SIM Type      Status      Function
  1-8:  DIGITAL INPUTS  0000_0000  AAAA_AAAA
  9-16: DIGITAL INPUTS  0000_0000  AAAA_AAAA
 17-24: DIGITAL INPUTS  0000_0000  AAAA_AAAA
 25-32: ANALOG INPUTS   0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000

>2TIO
*BRICK 2:  SIM Type      Status      Function
  1-8:  DIGITAL OUTPUTS  0000_0000  AAAA_AAAA -- SINKING
  9-16: DIGITAL INPUTS  0000_0000  AAAA_AAAA
 17-24: NO SIM PRESENT
 25-32: DIGITAL OUTPUTS  0000_0000  AAAA_AAAA -- SOURCING

```

## TLABEL

### Transfer Labels

Type	Transfer	Product	Rev
Syntax	<a_><!>TLABEL	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TLABEL: *NO LABELS DEFINED		
See Also	\$		

The 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

```

## TLIM Transfer Limits

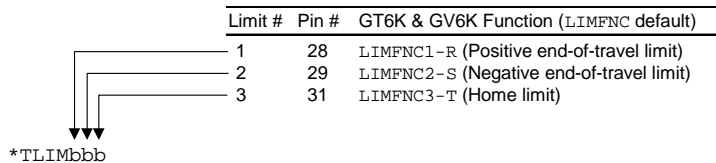
Type	Transfer	Product	Rev
Syntax	<!><a>TLIM<.i>	GT6K	6.0
Units	i = limit input number	GV6K	6.0
Range	System configuration dependant		
Default	n/a		
Response	TLIM: *TLIM110 TLIM.2: *0 (status of negative-direction limit)		
See Also	HOM, INDEB, INFNC, [ LIM ], LIMFNC, LIMLVL, TAS, TASF, TIN		

The TLIM command returns the current hardware state of the limit inputs located on the “DRIVE I/O” connector. This command reports the state of the limit inputs, regardless of their assigned function with the LIMFNC command. There are 3 limit inputs. To determine if an end-of-travel limit has been hit, refer to the TAS or TASF command response, bits 15 through 18.

This command does not report the status of onboard or external inputs configured as limit inputs with the INFNC command. For status on such inputs, refer to the TIN command.

The TLIM value is the debounced version of the limits status (debounced with the ØINDEB value). Axis status (TAS) bits 15 and 16 reports the non-debounced version of the end-of-travel limits.

TLIM response for limits on the DRIVE I/O connector (bits are numbered 1-3 from left to right):



### Input State Relationships:

Active Level Setting	Required Switch Type *	State	LIM/TLIM Report
Active low (LIMLVL0) <i>This is the default setting.</i>	End-of-travel limit: N.C.	<b>Grounded</b> — sinking current (device driving the input is on)	1 (no limit hit)
	Home limit: N.O.	<b>Not Grounded</b> — not sinking current (device driving the input is off)	0 (FAULT: limit hit)
Active high (LIMLVL1)	End-of-travel limit: N.O.	<b>Grounded</b> — sinking current (device driving the input is on)	0 (FAULT: limit hit)
	Home limit: N.C.	<b>Not Grounded</b> — not sinking current (device driving the input is off)	1 (no limit hit)

\* Compumotor recommends that all end-of-travel limit switches be normally-closed, because with normally-closed limit switches the drive detects a limit fault condition (and stops or inhibits motion) when the switch contact is open or if the wiring to the switch is broken.

## TMEM Transfer Memory Usage

Type	Transfer	Product	Rev
Syntax	<a_><!>TMEM	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TMEM: *150000 OF 150000 BYTES (100%) PROGRAM MEMORY REMAINING *1973 OF 1973 SEGMENTS (100%) COMPILED MEMORY REMAINING		
See Also	DEF, MEMORY, PCOMP, [ SEG ], TDIR, TSEG		

The TMEM command returns the amount of available memory for storing programs and for storing compiled profile path segments. A path segment is one element of a compiled profile path. 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.

---

## TMTEMP      Transfer Motor Temperature

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TMTEMP	GT6K	n/a
Units	Degrees C	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TMTEMP:    *TMTEMP45		(applicable only to servo axes)
See Also	DMONAV, DMONBV, DMTRWC, DMTTCM, DMTCW, TASX, TERRLG		

---

The TMTEMP reports the predicted temperature of the motor winding for Parker motors. The temperature is estimated using the winding and motor time constants, the rated continuous current, and the winding thermal resistance. The motor will fault (reported with TASX bit 30) at an estimated winding temperature of 125°C, assuming the ambient temperature is 40°C.

If you are using a non-Parker motor, the TMTEMP value depends on parameters you supply for DMTRWC, DMTTCM and DMTCW.

---

## TNMCY      Transfer Master Cycle Number

Type	Following; Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TNMCY	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TNMCY      *TNMCY0		
See Also	FMCLen, FMCNEW, [ FS ], TRGFN, TFS		

---

The TNMCY command displays the current master cycle number. The value represents the current cycle number, not the position of the master (or the follower). The master cycle number is set to zero when master cycle counting is restarted, and is incremented each time a master cycle finishes (i.e., rollover occurs). It will often correspond to the number of complete parts in a production run. This value may be used for subsequent decision making, or simply recording the cycle number corresponding to some other event.

**The master must be assigned first (FOLMAS command) before this command will be useful.**

For a complete discussion of master cycles, please refer to the Following chapter in the *Programmer's Guide*.

---

## TNT          Transfer Ethernet Status

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TNT	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TNT:      (see sample response below)		
See Also	NTADDR, NTFEN, TNTMAC		

---

The TNT command reports the current Ethernet conditions (see sample response below).

```
*GEM6K ETHERNET STATUS
*Ethernet enabled: NTFEN2
*GEM6K IP address: 172.20.34.156
*GEM6K Network Mask: 255.255.255.0
*GEM6K Ethernet address: 0-144-85-0-0-1 (decimal)
*GEM6K Ethernet address: 0-90-55-0-0-1 (hex)
*GEM6K Ethernet connected
```



## How to interpret the status report

- **Ethernet enabled (NTFEN):** NTFEN2 is recommended as the standard Ethernet communication mode, even if you are using a closed network and no file sharing. NTFEN2 is especially helpful if you are using Windows NT, or Windows 95/98 with File Sharing and/or an open network. **NOTE:** When using NTFEN2, you must also follow the ARP -S Static Mapping procedure (refer to the instructions in the *Programmer's Guide*).
- **GEM6K IP address:**  
This is the IP address that you specified (NTADDR command) in the configuration process (refer to the *Programmer's Guide*)
- **GEM6K Network Mask:**  
This is the network mask that you specified (NTMASK command) in the configuration process.
- **GEM6K Ethernet address:**  
This is the Ethernet “MAC” address that is given to the Gem6K unit at the factory.
- **GEM6K Ethernet connected/not connected:**  
Ethernet is “Connected” if the Ethernet hardware connection is good and your PC is communicating to the Gem6K over a valid Communication Server/Motion Planner connection.

**Error Status:** If the Ethernet connection fails, the Gem6K will set error status bit #22 (see ER, TER, and TERF) if error-checking bit #22 is enabled with the ERROR command. Also, if this error occurs, the Gem6K will branch to the ERRORP program.

---

<b>TNTMAC</b>		<b>Transfer Ethernet Address</b>	
Type	Transfer; Communications Interface	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!><B>TNTMAC	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TMAC: *0,144,85,0,0,1		
See Also	NTADDR, NTFEN, NTMASK		

---

The TNTMAC command reports the Gem6K's Ethernet address.

---

<b>TOUT</b>		<b>Transfer Output Status</b>	
Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!><@><B>TOUT<.i>	GT6K	6.0
Units	i = input number on the specified I/O brick (B) – see page 8	GV6K	6.0
Range	1-32 (dependent on I/O brick and options installed)		
Default	n/a		
Response	TOUT: *0000_000 (onboard outputs) 1TOUT *0000_0000_0000_0000_0000_0000_0000_0000 1TOUT.4: *1 (status of I/O point 4 on I/O brick 1)		
See Also	OUT, OUTFNC, OUTLVL, TAS, TASX, TIN, TINO, TIO		

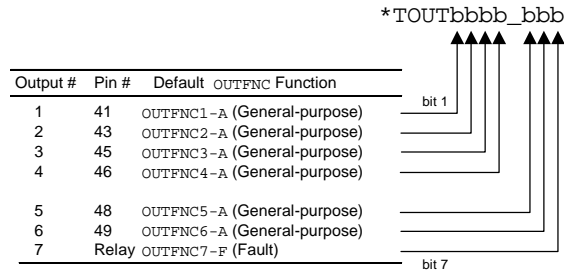
---

The TOUT command returns the present status (active or inactive) of the outputs on the DRIVE I/O connector, the dry contact relay output (labeled “RELAY COM” and “RELAY N.O.”) on the 4-pin removable connector, and the programmable outputs on external I/O bricks. The outputs are numbered 1 to *n* from left to right (*n* is the maximum number of I/O points on the I/O brick— refer to page 8 for details.)

If the status of a specific output is required, use the bit select operator (.). For example, 1TOUT.9 reports the status of the 1<sup>st</sup> I/O point on the 2<sup>nd</sup> SIM of I/O brick 1.

TOUT response for the outputs on the DRIVE I/O connector and the relay (bits are numbered 1-7 from left to right):

**GT6K & GV6K:**



**Output State Relationships** (outputs on the DRIVE I/O connector):

OUTLVL Setting	OUT State *	Current	OUT/TOUT status
OUTLVL0 (default)	OUT1	Sinking current	1
OUTLVL0	OUT0	No current flow	0
OUTLVL1	OUT1	No current flow	1
OUTLVL1	OUT0	Sinking current	0

\* The output is "active" when it is commanded by the OUT, OUTP, or POUT command (for example, OUTxx1 activates output #3).

**Output State Relationships** (external outputs on an EVM32 I/O brick):

Output Type	OUTLVL Setting	OUT State *	Current	OUT/TOUT status	LED
NPN (sinking)	OUTLVL0 (default)	OUT1	Sinking current	1	ON
NPN	OUTLVL0 (default)	OUT0	No current flow	0	OFF
NPN	OUTLVL1	OUT1	No current flow	1	OFF
NPN	OUTLVL1	OUT0	Sinking current	0	ON
PNP (sourcing)	OUTLVL0	OUT1	No current flow	1	OFF
PNP	OUTLVL0	OUT0	Sourcing current	0	ON
PNP	OUTLVL1 (default)	OUT1	Sourcing current	1	ON
PNP	OUTLVL1 (default)	OUT0	No current flow	0	OFF

\* The output is "active" when it is commanded by the OUT, OUTP, or POUT command (for example, OUTxx1 activates output #3).

TPC		Transfer Position Commanded		Product	Rev
Type	Transfer			GT6K	6.0
Syntax	<a_><!>TPC			GV6K	6.0
Units	Reported value represents distance units (scalable by SCLD)				
Range	Range of the reported value is ±2,147,483,648				
Default	n/a				
Response	TPC: *TPC+0				
See Also	DRES, ERES, [ PC ], [ PCC ], PSET, SCALE, SCLD, SMPER, TAS, TFB, TPCC, TPER				

Use the TPC command to display the *commanded position*. The TPC value is scaled by the distance scaling factor (SCLD) if scaling is enabled with the SCALE1 command.

If you issue a PSET command, the commanded position value will be offset by the PSET command value.

**GT6K:** The reported value is measured in commanded counts ("motor counts").

**GV6K:** The reported TPC value is measured in feedback device counts (resolution set with ERES). The commanded position (TPC) and the actual position (TPE) are used in the position error calculation (TPC - TPE = TPER).

**Example (GV6K only):**

```

TPC           ; Display the commanded position. Example is *TPC4000 (setpoint
              ; displayed in counts)
TPE           ; Display the actual position. Example is *TPE4004 (actual
              ; position displayed in counts)
TPER          ; Display position error. Example is *TPER-4 (error displayed
              ; in counts)

```

---

**TPCC**                      **Transfer Captured Commanded Position**

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TPCCc	GT6K	6.0
Units	c = trigger input letter (A, B or M) (Reported value is commanded counts, scalable by SCLD)	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TPCCA:    *TPCCA+0		
See Also	ENCCNT, INFNC, [ PC ], [ PCC ], [ PCMS ], PSET, SCALE, SCLD, SFB, TFB, TPC, [ TRIG ], TRGLOT, TTRIG		

---

Use the TPCC command to display the current captured commanded position, captured with the specific “trigger interrupt” input—A, B, or M.

General-purpose inputs on the DRIVE I/O connector:

Input #	Pin #	GT6K & GV6K Function (INFNC default)
1	37	INFNC1-A (General-purpose. If assigned, TRG-A must be on Input #1)
2	38	INFNC2-A (General-purpose. If assigned, TRG-B must be on Input #2)
3	39	INFNC3-A (General-purpose)
4	34	INFNC4-A (General-purpose)
5	35	INFNC5-A (General-purpose. If assigned, MASTER TRIGGER must be on Input #5)

**About Position Capture:** The commanded position can be captured only by an input that is defined as a “trigger interrupt” input with the INFNCi-H command (see INFNC for details). When a “trigger interrupt” input is activated, the commanded position is captured and the position is available through the use of the PCC operator and the TPCC display command.

**GT6K:** By default, the GT6K captures only the commanded position. However, if the drive has Encoder Capture Mode enabled with the ENCCNT command, only the encoder position is captured.

**Position Capture Status, Longevity of Captured Position:** Use the TTRIG and TRIG commands to ascertain if a trigger interrupt input has been activated. TTRIG displays the status as a binary report, and TRIG is an assignment/comparison operator for using the status information in a conditional expression (e.g., in an IF statement). Once the captured commanded position value is displayed with the TPCC command, the TTRIG/TRIG status bit for that trigger input is cleared; but the position information remains available until it is overwritten by a subsequent position capture from the same trigger input.

**Position Capture Accuracy:** The commanded position capture accuracy is  $\pm 1$  count.

**Scaling and Position Offset:** If scaling is enabled (SCALE1), the commanded position is scaled by the distance scaling factor (SCLD). If scaling is not enabled (SCALE0), the value reported will be actual commanded counts. If you issue a PSET (establish absolute position reference) command, any previously captured commanded positions will be offset by the PSET command value.

**Example:**

```

TPCCA        ; Report the captured command position, which was captured
              ; when trigger A (TRG-A) was activated
TPCCB        ; Report captured command position, which was captured
              ; when trigger B (TRG-B) was activated
TPCCM        ; Report captured command position, which was captured
              ; when the master trigger (TRG-M) was activated

```

---

## TPCE Transfer Position of Captured Encoder

Type	Transfer	Product	Rev
Syntax	<a_><!>TPCEc	GT6K	6.0
Units	c = trigger input letter (A, B or M) (Reported value represents encoder counts, scalable by SCLD)	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TPCEA: *TPCEA+0		
See Also	ENCCNT, INFNC, [ PCE ], PESET, PSET, SCALE, SCLD, SFB, TPE		

---

Use the TPCE command to display the current captured encoder position, from the time of the last trigger interrupt.

General-purpose inputs on the DRIVE I/O connector:

---

Input #	Pin #	GT6K & GV6K Function (INFNC default)
1	37	INFNC1-A (General-purpose. If assigned, TRG-A must be on Input #1)
2	38	INFNC2-A (General-purpose. If assigned, TRG-B must be on Input #2)
3	39	INFNC3-A (General-purpose)
4	34	INFNC4-A (General-purpose)
5	35	INFNC5-A (General-purpose. If assigned, MASTER TRIGGER must be on Input #5)

---

**About Position Capture:** The encoder position can be captured only by an input that is defined as a “trigger interrupt” input with the INFNCi-H command (see INFNC command). When a “trigger interrupt” input is activated, the encoder position is captured and the position is available through the use of the PCE operator and the TPCE display command.

**GT6K:** By default, the GT6K captures only the commanded position. To capture the encoder position, the axis must be in the Encoder Capture Mode (see ENCCNT command).

**Position Capture Status, Longevity of Captured Position:** Use the TTRIG and TRIG commands to ascertain if a trigger interrupt input has been activated. TTRIG displays the status as a binary report, and TRIG is an assignment/comparison operator for using the status information in a conditional expression (e.g., in an IF statement). Once the captured encoder position value is reported with the TPCE command, the TTRIG/TRIG status bit for that trigger input is cleared; but the position information remains available until it is overwritten by a subsequent position capture from the same trigger input.

**Position Capture Accuracy:** The encoder position capture accuracy is  $\pm 1$  encoder count.

**Scaling and Position Offset:** If scaling is enabled (SCALE1), the encoder position is scaled by the distance scaling factor (SCLD). If scaling is not enabled (SCALE0), the value reported will be actual encoder counts. If you issue a PSET (establish absolute position reference) command, any previously captured encoder positions will be offset by the PSET command value.

**Example:**

```
TPCEA      ; Report captured encoder position, which was captured
            ; when trigger A (TRG-A) was activated
TPCEB      ; Report captured encoder position, which was captured
            ; when trigger B (TRG-B) was activated
TPCEM      ; Report captured encoder position, which was captured
            ; when the master trigger (TRG-M) was activated
```

---

## TPCME      Transfer Captured Master Encoder Position

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TPCME	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TPCME      *TPCME+0		
See Also	INFNC, MEPOL, MESND, [ PME ], [ PCME ], [ PCMS ], PMECLR, PMESET, TPME, TPCMS		

---

Use the TPCME command to display the current captured master encoder position. The master encoder is connected to the connector labeled “Master Encoder.”

**Syntax:** VARn=PCME where n is the variable number; or PCME can be used in an expression such as IF(PCME>23450).

**About Position Capture:** The master encoder position can be captured only by the Master Trigger input (input #5), and only when that input is defined as a “trigger interrupt” input with the INFNC5-H command (see INFNC command). When the “trigger interrupt” input is activated (active edge), the master encoder position is captured and the position is available through the use of the PCME operator and the TPCME display command.

**Position Capture Status, Longevity of Captured Position:** Use the TTRIG and TRIG commands to ascertain if a trigger interrupt input has been activated. TTRIG displays the status as a binary report, and TRIG is an assignment/comparison operator for using the status information in a conditional expression (e.g., in an IF statement). Once the captured master encoder position value is displayed with the TPCME command, TTRIG/TRIG status bit #5 is cleared; but the position information remains available until it is overwritten by a subsequent position capture from the master trigger input.

**Position Capture Accuracy:** The master encoder position capture accuracy is  $\pm 1$  encoder count.

**Scaling and Position Offset:** The TPCME value is always in master encoder counts; it is never scaled. If you issue a PMESET (establish absolute position reference) command, any previously captured master encoder positions will be offset by the PMESET command value.

---

## TPCMS      Transfer Captured Master Cycle Position

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TPCMSc	GT6K	6.0
Units	c = trigger input letter (A, B or M) (Reported value represents master counts, scalable by SCLMAS)	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TPCMSA      *TPCMSA+0		
See Also	ENCCNT, FOLMAS, INFNC, [ PCMS ], PSET, SCALE, SCLMAS, SFB, TPCME, [ TRIG ], TRGLOT, TTRIG		

---

The TPCMS command transfers the captured position of the master within its current master cycle.

TPCMS (and TPMAS) is unique among position transfers, because its value rolls over to zero each time the entire master cycle length (FMCLen) has been traveled. Thus, the captured TPCMS value is essentially a snap-shot of the position relative to the master cycle at the time of the capture.

**The master must be assigned first (FOLMAS command) before this command will be useful.**

For a complete discussion of master cycles, refer to the Following chapter in the *Programmer’s Guide*.

General-purpose inputs on the DRIVE I/O connector:

Input #	Pin #	GT6K & GV6K Function (INFNC default)
1	37	INFNC1-A (General-purpose. If assigned, TRG-A must be on Input #1)
2	38	INFNC2-A (General-purpose. If assigned, TRG-B must be on Input #2)
3	39	INFNC3-A (General-purpose)
4	34	INFNC4-A (General-purpose)
5	35	INFNC5-A (General-purpose. If assigned, MASTER TRIGGER must be on Input #5)

**About Position Capture:** The master cycle position can be captured only by a trigger input that is defined as “trigger interrupt” input with the `INFNCi-H` command (see `INFNC` command). When a “trigger interrupt” input is activated, the master cycle position of the dedicated axis is captured and the position is available through the use of the `PCMS` operator and the `TPCMS` display command.

**Position Capture Status, Longevity of Captured Position:** Use the `TTRIG` and `TRIG` commands to ascertain if a trigger interrupt input has been activated. `TTRIG` displays the status as a binary report, and `TRIG` is an assignment/comparison operator for using the status information in a conditional expression (e.g., in an `IF` statement). Once the captured master cycle position value is reported with the `TPCMS` command, the `TTRIG/TRIG` status bit for that trigger input is cleared; but the position information remains available until it is overwritten by a subsequent position capture from the same trigger input.

**Position Capture Accuracy:** The master cycle position is interpolated; the capture accuracy is 50  $\mu$ s multiplied by the velocity of the axis at the time the trigger input was activated.

**Scaling and Position Offset:** If scaling is enabled (`SCALE1`), the master cycle position is scaled by the distance scaling factor (`SCLMAS`). If scaling is not enabled (`SCALE0`), the value assigned will be actual counts from the commanded or encoder master source as selected with the `FOLMAS` command. If you issue a `PSET` (establish absolute position reference) command, any previously captured master cycle positions will be offset by the `PSET` command value.

<b>TPE</b>		<b>Transfer Position of Encoder/Resolver</b>	
Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TPE	GT6K	6.0
Units	Encoder or resolver counts, or scaled by <code>SCLD</code>	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TPE: *TPE+0		
See Also	<code>ENCCNT</code> , <code>ERES</code> , <code>[ FB ]</code> , <code>[ PE ]</code> , <code>PESET</code> , <code>PSET</code> , <code>SCALE</code> , <code>SCLD</code> , <code>SFB</code> , <code>TFB</code> , <code>TPC</code> , <code>TPER</code>		

**GV6K:** The `TPE` command reports the present feedback device position, based on the encoder/resolver resolution (`ERES`).

**GT6K:** If the `ENCCNT1` mode is enabled `TPE` reports the encoder position, but in `ENCCNT0` mode (the factory default setting) the `TPE` report represents the commanded position.

**UNITS OF MEASURE and SCALING:** refer to page 16 or to the `SCLD` command.

If you issue a `PSET` command, the encoder position value will be offset by the `PSET` command value. If you are using a stepper axis in the `ENCCNT1` mode, use the `PESET` command instead.

The position error calculation is:  $TPER = TPC - TPE$ .

---

## TPER

### Transfer Position Error

Type	Transfers	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TPER	GT6K	n/a
Units	Reported value represents distance units (scalable by SCLD)	GV6K	6.0
Range	Range of the reported value is ±2,147,483,648		
Default	n/a		(applicable only to servo axes)
Response	TPER: *TPER+0		
See Also	ENCCNT, DMODE, DMONAV, DMONEV, DRES, ERES, [ FB ], [ PC ], [ PE ], [ PER ], SFB, SMPER, TANI, TAS, TFB, TPE, TPC		

---

The TPER command reports the present position error. The error is reported in feedback device counts and is based on the encoder/resolver resolution (ERES). It is scaled by the distance scaling factor (SCLD), if scaling is enabled with the SCALE1 command.

The position error is the difference between the commanded position and the actual position read by the feedback device. (TPER = TPC - TPE).

The position error is calculated every sample period.

#### Example:

```
TPC           ; Display the commanded position. Example is: *TPC4000
TPE           ; Display the actual position. Example is: *TPE4004
TPER          ; Display position error. Example is: *TPER-4
```

---

## TPMAS

### Transfer Current Master Cycle Position

Type	Following; Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TPMAS	GT6K	6.0
Units	Reported value represents master counts, scalable by SCLMAS.	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TPMAS *TPMAS0		
See Also	FMCLen, FMCNEW, FMCP, FOLMAS, FOLMD, [ FS ], MEPOL, [ NMCY ], [ PMAS ], SCALE, SCLMAS, TFS		

---

The TPMAS command transfers the current position of the master within its current master cycle. **The master must be assigned first (FOLMAS command) before this command will be useful.**

TPMAS is unique among position transfers, because master cycle position rolls over to zero each time the entire master cycle length (FMCLen value) has been traveled.

If scaling is enabled (SCALE1), the value returned is scaled by the master scaling factor (SCLMAS). If scaling is disabled (SCALE0), the value returned is in master counts (encoder/resolver counts, commanded counts, or analog input counts).

For a complete discussion of master cycles, refer to the Following chapter in the *Programmer's Guide*.

---

## TPME

### Transfer Position of Master Encoder

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TPME	GT6K	6.0
Units	Reported value represents master encoder counts.	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TPME *TPME+0		
See Also	MEPOL, MESND, [ PCME ], [ PE ], [ PME ], PMECLR, PMESET, TPCME		

---

Use the TPME command to display the current master encoder position. The master encoder is connected to the connector labeled “Master Encoder”. If you issue a PMESET command, the master encoder position value will be offset by the PMESET command value. The TPME value is always in encoder counts, it is never scaled.

---

## TPRA Transfer Absolute Resolver Position

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TPRA	GT6K	n/a
Units	counts	GV6K	6.0
Range	0 - (ERES-1)		
Default	n/a		
Response	TPRA: *TPRA145		(applicable only to servo axes)
See Also	ERES, SFB		

---

TPRA returns the actual resolver reading. For resolver drives, the command will be valid even if encoder feedback has been selected with the SFB command. For servo drives without resolver feedback, this command will always return 0.

---

## TPROG Transfer Program Contents

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TPROG<t>	GT6K	6.0
Units	t = text (name of program)	GV6K	6.0
Range	Text name of 6 characters or less		
Default	n/a		
Response	n/a		
See Also	DEF, TDIR, TMEM		

---

The TPROG command displays the contents of the program specified. If there is no such program, the drive responds with the error message: \*INVALID DATA. To see which programs have been created, use the TDIR command.

---

## TPSHF Transfer Net Position Shift

Type	Following; Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TPSHF	GT6K	6.0
Units	(Reported value represents commanded counts, scalable by SCLD)	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TPSHF *TPSHF+0		
See Also	FMCNEW, FMCP, FOLEN, FSHFC, FSHFD, [ PSHF ], SCALE, SCLD		

---

The TPSHF command transfers the net (absolute) follower axis position shift that has occurred since that last FOLEN1 command. The position returned will be the sum of all shifts performed on that axis, including decelerations due to limits, kill, or stop. The shift value is set to zero each time a new FOLEN1 command or a FOLMAS command (with a value other than zero) is issued.

If scaling is enabled (SCALE1), the PSHF value is scaled by the distance scaling factor (SCLD). If scaling is not enabled, the value is in commanded counts.



<b>TPSLV</b>		<b>Transfer Current Commanded Position of Follower Axis</b>	
Type	Following; Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TPSLV	GT6K	6.0
Units	(Reported value represents commanded counts, scalable by SCLD)	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TPSLV *TPSLV+0		
See Also	FMCNEW, FMCP, [ PSLV ], SCALE, SCLD		

The TPSLV command transfers the current commanded position of the follower axis. The master must be assigned first (FOLMAS command) before this command will be useful.

If scaling is enabled (SCALE1), the PSLV value is scaled by the distance scaling factor (SCLD). If scaling is not enabled, the value is in commanded counts.

<b>TRACE</b>		<b>Program Trace Mode Enable</b>	
Type	Program Debug Tool	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TRACE<b>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	b = 0 (disable) or 1 (enable)		
Default	0		
Response	TRACE: *TRACE0		
See Also	[ , ], [ # ], PORT, [ SS ], STEP, TRACEP, TRANS, TSS		

The TRACE command enables program trace mode. When in program trace mode, all commands executed are transferred out the Ethernet, RS-232 or RS-485 port, and the program from which the command came is identified.

**Example:**

```

DEL prog1          ; Delete program prog1
DEF prog1          ; Begin definition of program prog1
L                  ; Begin loop
  IF(IN.3=b1)      ; If input #3 is activated ...
  JUMP prog2       ; Jump to program prog2
  ELSE             ; If input #3 is not activated, continue with loop
  MC0              ; Select preset positioning mode
  MA0              ; Select incremental positioning mode
  A45              ; Set acceleration to 45 revs/sec/sec
  AD20             ; Set deceleration to 20 revs/sec/sec
  V7               ; Set velocity to 7 revs/sec
  GO1              ; Initiate motion
  NIF
LN                 ; End loop
END                ; End definition of program

DEL prog2          ; Delete program prog2
DEF prog2          ; Begin definition of prog2
K1                 ; Kill motion only
TERRLG            ; Display the error log
END                ; End definition of program

TRACE1            ; Enable trace mode
RUN prog1         ; Run program prog1

```

After enabling the trace mode, executing RUN prog1 places the following information in the output buffer: (assume input #3 is not activated)

```

*L
*IF( IN.3=b1 )
*ELSE
*MC0
*MA0
*A45
(continues)

```

---

## TRACEP Program Flow Mode Enable

Type	Program Debug Tool	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TRACEP<b>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	b = 0 (disable), 1 (enable) or X (don't care)		
Default	0		
Response	TRACEP: *TRACEP0		
See Also	TRACE		

---

The TRACEP command provides a debug tool to monitor the entry and exit of programs and their associated nest-levels.

### Example:

```
DEF PICK1
GOSUB PICK2
GOTO PICK3
END

DEF PICK2
GOSUB PICK4
END

DEF PICK3
END

DEF PICK 4
END

>TRACEP1
>PICK1
*INITIATE PROGRAM:    PICK1    NEST=1
*INITIATE PROGRAM:    PICK2    NEST=2
*INITIATE PROGRAM:    PICK4    NEST=3
*END:                 PROGRAM NOW:    PICK2    NEST=2
*END:                 PROGRAM NOW:    PICK1    NEST=1
*INITIATE PROGRAM:    PICK3    NEST=1
*END:                 PROGRAM EXECUTION TERMINATED
```

---

## TRANS Translation Mode Enable

Type	Program Debug Tool	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TRANS<b>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	b = 0 (disable), 1 (enable) or X (don't care)		
Default	0		
Response	TRANS: *TRANS0		
See Also	[ # ], [ SS ], STEP, TSS		

---

The TRANS command enables the program translation mode, in which all commands processed by the Gemini drive 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:

```
TRANS1          ; Enable translation mode
A10             ; Translate acceleration command A10. Response displayed
               ; is: 07 01 00 00 01 86 A0. Note that 07 hex represents a command
               ; memory requirement of 19 bytes.
GO1            ; Translate initiate motion command GO1. Response displayed
               ; is: 04 07 03 01. Note that 04 hex represents a
```

; command memory requirement of 4 bytes.

## TREV

### Transfer Revision Level

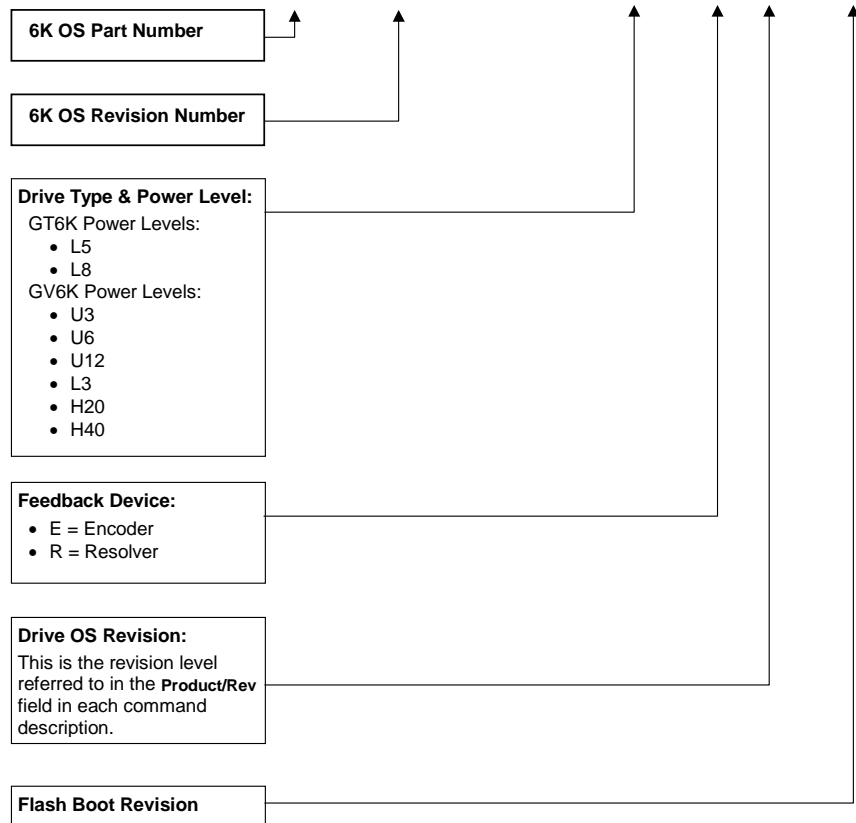
Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TREV	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TREV: *TREV92-016740-01-5.3.25 GEM6K GV6K-L3E D0.11 F1.3 (response varies by product)		
See Also	RESET, RFS		

The Transfer Revision Level (TREV) command reports the following information:

- 6K operating system part number
- 6K operating system revision number
- Drive type
- Drive power level
- Feedback device (for GV6K only)
- Drive operating system
- Flash boot revision (for hardware identification only)

Example Response:

```
*TREV92-016740-01-5.3.25 GEM6K GV6K-L3E D0.11 F1.3
```



**To update your drive operating system:** The operating system file is located in the software download section of the *Compumotor Online* web site (<http://www.compumotor.com>). Download the file to your hard drive and follow the relevant download procedure:

- Motion Planner users: refer to page 6.
- Communications Server (COM6SRVR.EXE) users: use the SendOS method described on page 6.

TRGFN		Trigger Functions	Product	Rev
Type	Inputs; Following; Motion		GT6K	6.0
Syntax	<a_><!>TRGFNcbb		GT6K	6.0
Units	c = trigger input letter 1 <sup>st</sup> b = bit to select Conditional GO (GOWHEN) function 2 <sup>nd</sup> b = bit to select Start New Master Cycle (FMCNEW) function		GV6K	6.0
Range	c = A, B, or M (M is master trigger, "TRG-M") b = 0 (disable function), 1 (enable function), or X (leave unchanged)			
Default	c = A; b = 0			
Response	TRGFN *TRGFNA00			
See Also	[ AS ], ERROR, ERRORP, FMCNEW, GOWHEN, INFNC, [ SS ], TAS, TRGLOT, TSS, [ TRIG ], TTRIG			

Use the TRGFN command to assign certain command functions to the trigger inputs on the DRIVE I/O connector:

Input #	Pin #	GT6K & GV6K Function (INFNC default)
1	37	INFNC1-A (General-purpose. If assigned, TRG-A must be on Input #1)
2	38	INFNC2-A (General-purpose. If assigned, TRG-B must be on Input #2)
3	39	INFNC3-A (General-purpose)
4	34	INFNC4-A (General-purpose)
5	35	INFNC5-A (General-purpose. If assigned, MASTER TRIGGER must be on Input #5)

#### NOTE

The input used in this command must first be defined as a *Trigger Interrupt* input with the INFNCi-H command.

- **“Conditional GO”** Function (TRGFNC1x): Suspend execution of the next start-motion command until the specified trigger input goes active. Start-motion commands are:
  - GO (standard command to begin motion)
  - FGADV (begin geared advance – for Following motion)
  - FSHFC (begin continuous shift – for Following motion)
  - FSHFD (begin preset shift – for Following motion)

Axis status bit #26 (reported with TASF, TAS, or AS) is set to one (1) when there is a pending “Conditional GO” condition initiated by a TRGFN command; this bit is cleared when the trigger is activated or when a stop command (S) or a kill command (K) is issued. If you need execution to be triggered by other factors (e.g., input state, master position, encoder position, etc.) use the GOWHEN command.

- **“New Master Cycle”** Function (TRGFNCx1): This is equivalent to executing the FMCNEW command. When the specified trigger input goes active, the drive begins a new Following master cycle. For more information on master cycles, refer to the Following chapter in the *Programmer’s Guide*.

These trigger functions are cleared once the function is complete. To use the trigger to perform another conditional Go or begin a new master cycle, the TRGFN command must be given again.

**TRGFN in Compiled Motion:** When used in a compiled program, a TRGFNC1x (conditional GO function) command will pause the profile in progress (motion continues at constant velocity) until the trigger is

activated to execute the next move profile. When used in a compiled profile, the TRGFN command consumes one segment of compiled memory. When used in a compiled Following profile, the TRGFN command is ignored on the reverse Following profile (i.e., when the master is moving in the opposite direction of that specified in the FOLMAS command).

**Trigger Interrupt Status:** The status of a trigger interrupt event is reported with the TTRIG and TRIG commands.

**Example:** (refer also to the FOLEN examples)

```
TRGFNBx1      ; When trigger B goes active, a new master cycle will begin
TRGFNB1x      ; When trigger B goes active, execute the move
               ; commanded with the GO command.
GO1           ; The move is commanded, but will not execute until
               ; trigger B goes active.
```

---

## TRGLOT      Trigger Interrupt Lockout Time

Type	Input	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TRGLOT<I>	GT6K	6.0
Units	i = time in milliseconds	GV6K	6.0
Range	0-250		
Default	24		
Response	TRGLOT: *TRGLOT24		
See Also	INDEB, INFNC, RE, REG, TIN, TRGFN, [ TRIG ], TTRIG		

---

The TRGLOT command configures the amount of time in which a “trigger interrupt” input (all inputs configured with the INFNCi-H command) is disabled between its initial active transition and its secondary active transition. This allows rapid recognition of a trigger interrupt input, but prevents subsequent bouncing of the input from causing a false position capture, registration move, or TRGFN event. The lockout time affects only those inputs configured as H (trigger interrupt) with the INFNCi-H command during those interrupt actions (registration, position capture, etc.).

The TRGLOT setting overrides the existing INDEB setting for only the inputs that are assigned the “Trigger Interrupt” function with the INFNCi-H command (see INFNC for details).

**Example:**

```
INFNC1-H      ; Assign input #1 as a "trigger interrupt" input
TRGLOT40     ; Set lockout time for all "trigger interrupt" inputs
               ; to be 40 milliseconds
```

---

## [ TRIG ]      Trigger Interrupt Status

Type	Assignment or Comparison	<b>Product</b>	<b>Rev</b>
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	INFNC, [ PCC ], [ PCE ], [ PCME ], [ PCMS ], TAS, TPCC, TPCE, TPCME, TPCMS, TRGFN, TRGLOT, TTRIG		

---

Use the TRIG operator to assign the Trigger Interrupt 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, 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.



<b>TSCF</b>		<b>Transfer Controller Status (full-text report)</b>		<b>Product</b>	<b>Rev</b>
Type	Transfer			GT6K	6.0
Syntax	<a_><!>TSCF			GV6K	6.0
Units	n/a				
Range	n/a				
Default	n/a				
Response	TSCF: (see example below)				
See Also	[ SC ], SCANP, TSC				

The TSCF command provides a full-text report of the current “Controller Status” register. This is an alternative to the binary report (TSC).

Example TSCF response:

```

*TSCF
*Reserved          NO          Reserved          NO
*Reserved          NO          Reserved          NO
*SCANP Running    NO          Reserved          NO
*Reserved          NO          Reserved          NO
*
*Reserved          NO          Reserved          NO
*Reserved          NO          Reserved          NO
*Reserved          NO          Reserved          NO
*Reserved          NO          Reserved          NO
*
*Reserved          NO          Reserved          NO
*Reserved          NO          Reserved          NO
*Reserved          NO          Reserved          NO
*Reserved          NO          Reserved          NO
*
*Reserved          NO          Reserved          NO
*Reserved          NO          Reserved          NO
*Reserved          NO          Reserved          NO
*Reserved          NO          Reserved          NO

```

<b>TSCAN</b>		<b>Transfer Scan Time of PLCP Program</b>		<b>Product</b>	<b>Rev</b>
Type	Transfer; PLC Scan Program			GT6K	6.0
Syntax	<a_><!>TSCAN			GV6K	6.0
Units	Response is in increments of 2 milliseconds				
Range	2 ms - unlimited				
Default	n/a				
Response	TSCAN *TSCAN4 (4 ms corresponds to 2 system updates)				
See Also	EXE, PLCP, SCANP				

The TSCAN command reports the duration it takes the last PLCP program to be scanned completely. A compiled PLCP program is launched into Scan mode using the SCANP command. During each 2 ms system update, the PLCP program is scanned an allotted 0.5 ms window. If the PLCP program requires more than 0.5 ms to be scanned, the program will be paused and then resumed at the next system update. The value reported by the TSCAN command is in multiples of the 2 ms system update period.

**Example:**

```

SCANP PLCP1 ; Start execution of compiled PLCP program PLCP1 in Scan mode
TSCAN      ; Report the duration of the scan program in multiples of the
           ; 2 millisecond system update period. An example response might
           ; be *TSCAN4 (indicates that 2 system update periods, a total of
           ; 4 milliseconds was required to scan the last PLCP program).

```

---

## TSEG Transfer Number of Free Segment Buffers

Type	Compiled Motion; Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TSEG	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TSEG: *TSEG258		
See Also	MEMORY, TDIR, TMEM, [SEG], [SS], TSS, TSSF		

---

The TSEG command returns the number of free segment buffers in compiled memory.

System status bit (see TSSF, TSS, and SS) 29 is set when the compiled memory is 75% full, and bit 30 is set if the compiled memory is 100% full.

---

## TSGSET Transfer Servo Gain Set

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TSGSETi	GT6K	n/a
Units	i = gain set identification number (see SGSET command)	GV6K	6.0
Range	1-3		
Default	n/a		
Response	(reports all gains in the specified set)		(applicable only to servo axes)
See Also	SFB, SGAF, SGENB, SGSET, SGVF, TGAIN		

---

This command allows you to display any of the 3 gain sets that you saved with the SGSET command. Up to 3 gain sets can be saved with the SGSET command. Each gain set contains these gain parameters:

- DIBW (current loop bandwidth)
- DMTLIM (torque/force limit)
- DMVLIM (velocity limit)
- DNOTAD (notch filter A depth)
- DNOTAF (notch filter A frequency)
- DNOTAQ (notch filter A quality factor)
- DNOTBD (notch filter B depth)
- DNOTBF (notch filter B frequency)
- DNOTBQ (notch filter A quality factor)
- DNOTLD (notch lead filter break frequency)
- DNOTLG (notch lag filter break frequency)
- DPBW (position loop bandwidth)
- LDAMP (load damping)
- LJRAT (load-to-rotor inertia ratio or load-to-force mass ratio)
- SGAF (acceleration feedforward gain)
- SGINTE (integrator enable)
- SGIRAT (current damping ratio)
- SGPRAT (position loop ratio)
- SGPSIG (velocity/position bandwidth ratio)
- SGVF (velocity feedforward gain)
- SGVRAT (velocity damping ratio)

**NOTE:** To report the present gain values in effect, use the TGAIN command.

---

## TSKAX Task Axis

Type	Multi-Tasking	<b>Product</b>	<b>Rev</b>
Syntax	i%TSKAX<a>	GT6K	6.0
Units	i = task number a = 0 or 1	GV6K	6.0
Range	For i: 0-10 For <a>: 1 or 0		
Default	a = 1		
Response	TSKAX: *TSKAX1		
See Also	%, TTASK, [ TASK ], TSWAP, [ SWAP ], TSKTRN		

---

The Task Axis command (TSKAX) allows you to specify whether or not a task is associated with motion.

The default condition in multi-tasking (TSKAX1) is that each task is associated with motion. This means, for example, that when the load hits an end-of-travel limit, program execution will be killed within that task, and in all other tasks associated with motion.



Use the TSKAX0 command to eliminate altogether a task's association with motion. This would be appropriate for a task that is not involved in motion control, but may control I/O or start other tasks.

**Example:**

```
DEF main          ; Begin definition of program called "main"
1%TSKAX1         ; Associate Task1 with motion
2%TSKAX0         ; Make Task2 independent of motion
1%move1         ; Execute stored program "move1" in Task1
2%inout         ; Execute stored program "inout" in Task2
END
```

---

## TSKTRN      Task Turns Before Swapping

Type	Multi-Tasking	<b>Product</b>	<b>Rev</b>
Syntax	i <sub>1</sub> %TSKTRNi <sub>2</sub>	GT6K	6.0
Units	i <sub>1</sub> = task number i <sub>2</sub> = number of turns before task swap	GV6K	6.0
Range	i <sub>1</sub> = 0-10 i <sub>2</sub> = 0-10,000		
Default	i <sub>1</sub> = 0 i <sub>2</sub> = 1		
Response	TSKTRN: *TSKTRN1		
See Also	%, LOCK, TTASK, [ TASK ], TSWAP, [ SWAP ], TSKAX		

---

Use the TSKTRN command to set the relative amount of processing time a task will get. Under default multi-tasking operation, all active tasks have an equal share of processing time; that is, each task executes one "turn" and then "swaps" control to the next active task. (A "turn" is the execution of a command, or a portion of a complex command such as those for math and trig operators.)

For example, if Task2 issued a TSKTRN6 command, while the other tasks stayed at TSKTRN1, Task2 would execute 6 commands (or portions of long commands) before relinquishing control to another task.

The TSKTRN value for a task may be changed at any time, allowing a task to increase its weight for an isolated section of program commands.

---

## TSROFF      Transfer Resolver Offset Angle

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TSROFF	GT6K	n/a
Units	angle offset, in degrees	GV6K	6.0
Range	-180.0 to +180.0		
Default	n/a		(applicable only to servo axes)
Response	TSROFF: *TSROFF-90.0		
See Also	DMODE, SRSET		

---

Use the TSROFF command to ascertain the actual resolver offset angle. SRSET sets the resolver offset angle. When a non-zero value is specified for SRSET, it becomes the new offset angle. When no value is specified, and the drive is in DMODE11 (feedback alignment mode), then a routine is executed to automatically set the resolver angle.

## TSS Transfer System Status

Type	Transfer	Product	Rev
Syntax	<a_><!><%>TSS<.i>	GT6K	6.0
Units	i = system status bit number	GV6K	6.0
Range	1-32		
Default	n/a		
Response	TSS: *TSS1000_1000_0000_0000_0000_0000_0000_0000 TSS.1: *1 (status of Task 0 status bit #1—system is ready)		
See Also	PORT, TAS, TCMDER, TRGFN, TSSF, TSTAT, [ TRIG ], TTRIG		

The TSS command provides information on the 32 system status bits. The TSS status command reports a binary bit report. If you would like to see a more descriptive text-based report, use the TSSF command description.

Response for TSS (b can equal  $\emptyset$ , 1, X, or x): \*TSS<sup>^</sup>bbbb\_bbbbb\_bbbbb\_bbbbb\_bbbbb\_bbbbb\_bbbbb<sup>^</sup>  
Bit #1
Bit #32

### MULTI-TASKING

If you are using multi-tasking, be aware that each task has its own system status register. Therefore, to check a specific task's system status, you must prefix the TSS command with the task identifier (e.g., 2%TSS to check system status for Task 2). If no task identifier is given, the TSS response is for the task supervisor (Task 0).

BIT (Left to Right)	Function (1 = yes, $\emptyset$ = no)	BIT (Left to Right)	Function (1 = yes, $\emptyset$ = no)
1	System Ready (powered up and ready to receive commands)	17	Loading Thumbwheel Data ([TW])
2	Reserved	18	External Program Select Mode is enabled (INSELP1)
3	Executing a Program (see EXE)	19	Dwell in Progress (T command)
4	Immediate Command (set if last command was immediate)	20	Waiting for RP240 Data—[DREAD] or [DREADF]
5	In ASCII Mode	21	RP240 Connected— current PORT setting only
6	In Echo Mode (ECHO1)— current PORT setting only	22	Non-volatile Memory Error
7	Defining a Program	23	GV6K: Servo data gathering transmission in progress; GT6K: Reserved
8	In Trace Mode (TRACE1)	24	Suspended on Swap (see LOCK command)
9	In Step Mode (STEP1)	25	Reserved
10	In Translation Mode (must use fast status area to see)	26	Suspended on COM1 (see LOCK command)
11	Command Error Occurred (bit is cleared when TCMDER is issued)	27	Suspended on COM2 (see LOCK command)
12	Break Point Active (BP)	28	Program Pending (see EXE command)
13	Pause is Active (PS command or pause input, INFNCi-E)	29	Compiled memory is 75% full
14	Wait is in progress (WAIT)	30	Compiled memory is 100% full
15	Monitoring On Condition (ONCOND)	31 *	Compile operation failed (PCOMP)
16	Waiting for Data (READ)	32	EXE Failed (see EXE command)

- \* Bit #31: failed PCOMP compile is cleared on power up, RESET, or after successful compile. Possible causes include:
  - Errors in profile design (e.g., change direction while at non-zero velocity; distance & velocity equate to < 1 count per system update; preset move profile ends in non-zero velocity)
  - Profile will cause a Following error (see TFSF, TFS, or FS command descriptions)
  - Out of memory (see TSS bit #30)
  - Axis already in motion at the time of the PCOMP command
  - Loop programming errors (e.g., no matching PLOOP or PLN; more than 4 embedded PLOOP/PLN loops)
  - PLCP program contains invalid commands.

## TSSF

### Transfer System Status (full-text report)

Type	Transfer	Product	Rev
Syntax	<a_><!><%>TSSF	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TSSF: (see example below)		
See Also	PORT, [ SS ], TAS, TCMDER, TRGFN, TSS, TSTAT		

The TSSF command returns a text-based status report. This is an alternative to the binary report (TSS).

#### MULTI-TASKING

If you are using multi-tasking, be aware that each task has its own system status register. Therefore, to check a specific task's system status, you must prefix the TSSF command with the task identifier (e.g., 2%TSSF to check system status for Task 2). If no task identifier is given, the TSSF response is for the task supervisor (Task 0).

Example TSSF response:

```
*TSSF
*System Ready          YES      Thumbwhl Data Load NO
*RESERVED              NO       Ext Prog Sel Mode  NO
*Program Executing    NO       Time Command       NO
*Immediate Comm Last  NO       Waiting RP240 Data NO
*
*ASCII Mode           YES      RP240 Connected    NO
*Echo Mode            YES      Memory Error        NO
*Defining a Program   NO       Servo Data Transfer NO (RESERVED for GT6K)
*Trace Mode           NO       Suspended on Swap  NO
*
*Step Mode            NO       RESERVED            NO
*FS Translate Mode    NO       Suspended on COM1  NO
*Command Error        NO       Suspended on COM2  NO
*Break Point Active   NO       Program Pending     NO
*
*Pause Active         NO       Comp Mem Near Full  NO
*Wait Active          NO       Compiled Mem Full   NO
*Checking On Conds    NO       Compile Error       NO
*Waiting for Data     NO       EXE Failed          NO
```

---

## TSTAT            Transfer Statistics

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TSTAT	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TSTAT:    (See below)		
See Also	NTADDR, TAS, TDIR, TER, TFB, TIN, TIO, TOUT, TPC, TPE, TREV, TSWAP, TSS, TTIM, TUS, TVEL		

---

The following is an example (**NOTE:** The response for each Gem6K Series product will vary slightly.):

```
*GEM6K Revision: 92-XXXXXX-01-5.3  GEM6K GV6K-U6E D0.11 F1.62
*Ethernet address: xxxxxxxxxx;  IP address: 192.168.10.30
*Power-up program assignment (STARTP):
*ENABLE input OK: Yes
*Drive status (DRIVE): 0
*Encoder resolution (ERES): 4000
*Hard Limit enable: LH3
*Soft Limit enable: LS0
*Current Motion Attributes:
*  Scaling enabled (SCALE1): 0
*  Acceleration scaler (SCLA): 4000
*  Distance scaler: 1
*  Velocity scaler: 4000
*  Commanded position (TPC): +0
*  A10.0000
*  AA10.0000
*  AD10.0000
*  ADA10.0000
*  V1.0000
*  D+4000
*I/O Status:
*  Onboard limit inputs:
*    Hardware state (TLIM): 110
*  Expansion I/O bricks: See TIO response
*Axis Status (see TASF for full text report):
*  Axis 1 (1TAS): 0010_0000_0000_1000_0000_0001_0000_0000
*System Status (This is Task 0 if using multi-tasking):
*5 Programs Defined; Scale Enable 0
*System Status: 1000_1100_0000_0000_0000_0100_0000_0000
```

---

## TSTLT            Transfer Settling Time

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TSTLT	GT6K	n/a
Units	Reported value represents milliseconds	GV6K	6.0
Range	n/a		
Default	n/a		(applicable only to servo axes)
Response	TSTLT:    *TSTLT502		
See Also	STRGTD, STRGTE, STRGTT, STRGTV		

---

TSTLT 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 *Programmer's Guide*.

---

## TSWAP      Transfer Current Active Tasks

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<!>TSWAP	GT6K	6.0
Units	n/a	GV6K	6.0
Range	Binary response status of tasks (0 = inactive, 1 = active). 10-bit pattern represents tasks 1-10 from left to right.		
Default	n/a		
Response	TSWAP:      *TSWAP1001_0000_00 (tasks 1 and 4 are active) TSWAP.3:    *0 (task 3 is inactive)		
See Also	%, [ SS ], [ TASK ],, TSS		

---

The TSWAP command reports a binary bit pattern indicating the tasks that are currently active. Note that TSWAP only indicates if a task is active; to ascertain exactly what activity the task has at a given time, use the system status (SS or TSS commands).

TSWAP's binary 10-bit pattern represents tasks 1-10, from left to right. A "1" indicates that the task is active, and a "0" indicates that the task is inactive. To check the status of only one task, you may use the bit select (.) operator. For example, TSWAP.3 checks the status of Task 3 only.

The "Task Supervisor", represented by task Ø, is always active and is therefore not included in the SWAP and TSWAP status.

---

## TTASK      Transfer Task Number

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<!>TTASK	GT6K	6.0
Units	Reported value is the number of the controlling task.	GV6K	6.0
Range	0-10		
Default	n/a		
Response	TTASK:    *TTASK2 (Task 2 executed the TTASK command)		
See Also	%, [ TASK ]		

---

Use the TTASK command to display the task number of the task which executed the command. This could be used for diagnostic purposes, as a way to indicate which task is executing a given section of program.

---

## TTIM      Transfer Timer

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!><%>TTIM	GT6K	6.0
Units	Reported value represents milliseconds	GV6K	6.0
Range	Maximum count is 999,999,999 (approx. 11 days, 13 hours)		
Default	n/a		
Response	TTIM:      *TTIM64000		
See Also	T, [ TIM ], TIMINT, TIMST, TIMSTP		

---

The 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.

**Multi-Tasking:** Each task has its own timer.

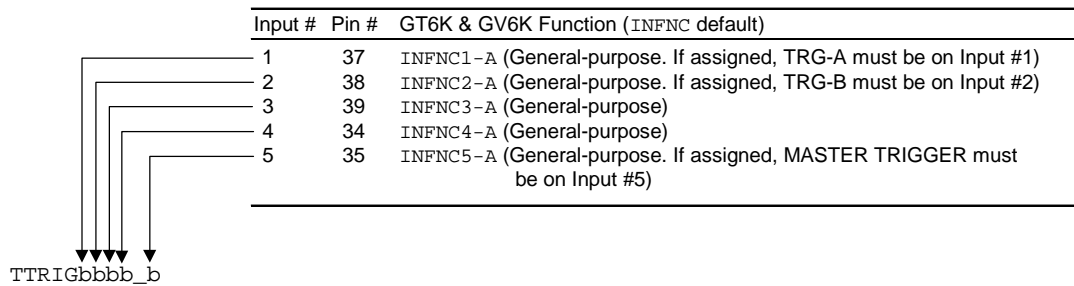
## TTRIG Transfer Trigger Interrupt Status

Type	Transfer, Inputs	Product	Rev
Syntax	<a><!>TTRIG	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TTRIG *TTRIG0000_0		
See Also	INFNC, [ PCC ], [ PCE ], [ PCME ], [ PCMS ], TAS, TPCC, TPCE, TPCME, TPCMS, TRGFN, TRGLOT, [ TRIG ]		

Use the TTRIG command to check whether a “trigger interrupt” input has been activated to capture a position, initiate a registration move, or execute a TRGFN function. “Trigger Interrupt” inputs are onboard inputs that have been assigned the trigger interrupt function with the INFNCi-H command.

Each TTRIG bit is cleared when the captured position value is read with the PCC, PCE, PCME, PCMS, TPCC, TPCE, TPCME, or TPCMS commands, but the position information is still available from the respective register until it is overwritten by a subsequent position capture by the same trigger input.

TTRIG bit assignments for inputs on the DRIVE I/O connector are shown below (bits are numbered 1-5 from left to right). A bit that is set (“1”) indicates the trigger interrupt has occurred, a “0” indicates no trigger interrupt.



### Example

```
COMEXC1 ; Continuous command execution
INFNC1-H ; Define input 1 as trigger A
REGA10000 ; Registration move on trigger A event
WAIT(TRIG.1=B1) ; Wait for trigger A event to occur
WRITE"TRIGGER A OCCURRED" ; Display message
TTRIG ; Get report back (display to monitor)
; response should be: *TTRIG1000_0
```

## TTRQ Transfer Commanded Torque/Force

Type	Transfer	Product	Rev
Syntax	<a_><!>TTRQ	GT6K	n/a
Units	percent of the DMTSCL value	GV6K	6.0
Range	-100.00% to +100.00% : ±0.01		
Default	n/a		(applicable only to servo axes)
Response	TTRQ: *TTRQ50.00		
See Also	DMODE, DMONAV, DMONBV, DMTLIM, DMTSCL, TTRQ		

In Autorun mode (DMODE13) and Torque/Force Tuning mode (DMODE15), TTRQ = 0. In all other DMODE modes, TTRQ reports the actual internal torque/force setpoint as a percentage of DMTSCL. Several commands may limit the maximum value of TTRQ. These commands include DMTLIM, DMVLIM, DMTIC, DMTIP, TDICNT, TDIMAX.

---

## TTRQA Transfer Actual Torque/Force

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TTRQA	GT6K	n/a
Units	percent of full-scale torque/force set by DMTSCL command	GV6K	6.0
Range	-100.00% to +100.00% : ±0.01		
Default	n/a		(applicable only to servo axes)
Response	TTRQA: *TTRQA55		
See Also	DMONAV, DMONBV, DMTKE, DMTLIM, DMTSCL, TTRQ		

---

The TTRQA command reports the calculated torque/force, based on q-axis current and the motor's Ke, as a percentage of the full-scale torque/force set by DMTSCL command.

---

## TUS Transfer User Status

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TUS<.i>	GT6K	6.0
Units	i = user status bit number	GV6K	6.0
Range	1 - 16		
Default	n/a		
Response	TUS: *TUS1111_0000_1111_0000 TUS.4: *1 (user status bit 4 is reported)		
See Also	INDUSE, INDUST, [ US ]		

---

The 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, or an input.

### Example:

```
INDUSE1           ; Enable user status
INDUST1-5A        ; User status bit 1 defined as axis status bit 5
INDUST2-3A        ; User status bit 2 defined as axis status bit 3
3INDUST3-5J       ; User status bit 3 defined as input 5 on I/O brick 3
2%INDUST16-2I     ; User status bit 16 defined as system status bit 2 for task 2
TUS               ; Return the state of the user status word
```

---

## TVE Transfer Velocity Error

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TVE	GT6K	n/a
Units	Revs/sec (linear motors: see DMEPIT for linear/rotary conversion)	GV6K	6.0
Range	-200 to 200		(applicable only to servo axes)
Default	n/a		
Response	TVE: *TVE3		
See Also	DMEPIT, DMONAV, DMONBV, DMVLIM, SMVER, TVEL, TVELA, V		

---

The TVE command reports velocity error in revs/sec (rotary) or meters/sec (linear). The velocity error is the difference between the commanded velocity (TVEL) and estimated actual velocity (TVELA).

The maximum allowable velocity error limit is established with the SMVER command.

---

## TVEL Transfer Current Commanded Velocity

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TVEL	GT6K	6.0
Units	Reported value is in units/sec (scalable by SCLV) (linear motors: see DMEPIT for linear/rotary conversion)	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TVEL: *TVEL23.3450		
See Also	DMEPIT, DMONAV, DMONBV, DMVLIM, ERES, SCALE, SCLV, TVELA, V, [ VEL ]		

---

The TVEL command reports the internal commanded velocity. It is not the programmed velocity (V), and is limited by DMVLIM. The sign determines the direction of motion. If scaling is enabled (SCALE1), the TVEL value is scaled by the velocity scaling factor (SCLV).

**Stepper Axes:** If scaling is disabled (SCALE0), the value is measured in revolutions/sec (actual velocity in commanded counts/sec divided by the drive resolution DRES value).

**Servo Axes:** If scaling is disabled (SCALE0), the value is measured in encoder revs/sec or resolver revs/sec.

---

## TVELA Transfer Current Actual Velocity

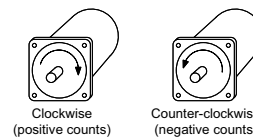
Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TVELA	GT6K	6.0
Units	Reported value is in units/sec (linear motors: see DMEPIT for linear/rotary conversion)	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TVELA: *TVELA+1.55		
See Also	DMEPIT, DMONAV, DMONBV, DMVLIM, SCALE, SCLV, SFB, TVEL, V, [ VEL ], [ VELA ]		

---

For servos, the TVELA command reports the velocity as derived from the feedback device. The sign determines the direction of motion.

### Rotary Motors:

Positive values represent clockwise motion and negative values represent counter-clockwise motion (assuming you connected the feedback device according to instructions in the *Hardware Installation Guide*).



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.

### Units of Measure:

**GT6K Steppers:** The value reported by steppers is the TVEL value, *not* velocity derived from the feedback device. If scaling is disabled (SCALE0), the value is measured in revolutions/sec (actual velocity in commanded counts/sec divided by the drive resolution DRES value).

**GV6K Servos:** If scaling is enabled (SCALE1), the velocity value will be scaled by the velocity scaling factor (SCLV). If scaling is not enabled (SCALE0), the value returned will be in encoder revs/sec or resolver revs/sec.

### Example:

```
TVELA          ; Reports the current actual velocity; since no motion is
               ; commanded, the servoing velocities are reported.
               ; Example response is: *TVELA+0.0097
```



---

## TVMAS      Transfer Current Master Velocity

Type	Following and Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TVMAS	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	TVMAS      *TVMAS+0		
See Also	FFILT, FOLMAS, SCALE, SCLMAS, V, [ VMAS ]		

---

The TVMAS command transfers the current velocity of the master. The master must be assigned first (FOLMAS command) before this command will be useful.

The precision of the reported TVMAS value is dependent upon the FFILT filter value (details are provided in the “Master Position Filtering” section in the Following chapter of the *Programmer's Guide*).

If scaling is enabled (SCALE1), the value returned is scaled by the master scaling factor (SCLMAS). If scaling is disabled (SCALE0), the value returned is in counts/sec.

---

## [ TW ]      Thumbwheel Assignment

Type	Assignment or Comparison	<b>Product</b>	<b>Rev</b>
Syntax	TWi (See below for examples)	GT6K	6.0
Units	i = sets used by INPLC, INSTW, OUTPLC and OUTTW	GV6K	6.0
Range	1-8		
Default	n/a		
Response	n/a		
See Also	INPLC, INSTW, OUTPLC, OUTTW, [ SS ], TSS		

---

The 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. Rule of Thumb for command value substitutions: If the command syntax shows that the command field requires a real number (denoted by <r>) or an integer value (denoted by <i>), you can use the TW substitution (e.g., V(TW)).

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., A(TW2)). However, the TW command cannot be used in an expression such as VAR4=1 + TW2 or IF(TW2<8).

For more information on interfacing thumbwheels, refer to the *Programmer's Guide*

### Example:

```
INSTW2,1-4,5      ; Set INSTW set 2 as BCD digits on onboard inputs 1-4, with  
                  ; input 5 as the sign bit  
OUTTW2,1-3,4,50   ; Set OUTTW set 2 as output strobes on onboard outputs 1-3,  
                  ; with output 4 as the output enable bit, and strobe time  
                  ; of 50 milliseconds  
A(TW2)            ; Read data into acceleration using INSTW set 2 and  
                  ; OUTTW set 2 as the data configuration
```

---

## UNTIL( )      Until Part of Repeat Statement

Type	Program Flow Control	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>UNTIL(expression)	GT6K	6.0
Units	n/a	GV6K	6.0
Range	Up to 80 characters (including parentheses)		
Default	n/a		
Response	n/a		
See Also	JUMP, REPEAT		

---

The 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, AS, D, DPTR, ER, IN, INO, MOV, OUT, PC, PCC, PCE, PCMS, PE, PER, SS, TIM, US, V, VEL, etc.) can be used within the UNTIL expression.

### Example:

```
REPEAT          ; Beginning of REPEAT ... UNTIL( ) loop
G01             ; Initiate motion
IF(IN=b1X0)    ; IF condition: if onboard input 1 = 1, input 3 = 0
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 encoder
NIF            ; End IF statement
UNTIL(VAR1=12) ; Repeat loop until variable 1 = 12
```

---

## [ US ]      User Status

Type	Assignment or Comparison	<b>Product</b>	<b>Rev</b>
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	INDUSE, INDUST, TUS		

---

The User Status (US) operator 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, 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:** VARn=US where “n” is the binary variable number,  
or US can be used in an expression such as IF(US=b1101), 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, or an input.

You can use the bit select (.) operator to assign only one bit of the user status value to a binary variable, instead of all 16. For example, VARB1=US.12 assigns user status bit 12 to binary variable 1.

**Example:**

```

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=b111011X11) ; 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=h7F00)  ; 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

```

---

<b>V</b>	<b>Velocity</b>		
Type	Motion	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>V<r>	GT6K	6.0
Units	r = units/sec (linear motors: see DMEPIT for linear/rotary conversion)	GV6K	6.0
Range	Stepper Axes:0.0000-60.0000 (after conversion to revs/sec) Servo Axes: 0.0000-200.0000 (after conversion to revs/sec; if ERES > 10,000, maximum = 2,000,000/ERES)		
Default	1.0000		
Response	V: *v1.0000		
See Also	DMEPIT, GO, MC, SCALE, SCLV, TSTAT, TVEL, TVELA, [ V ], [ VEL ], [ VELA ], VF		

---

The Velocity (v) command defines the speed at which the motor will run when given a GO command. The motor will attempt to accelerate at a predefined acceleration rate (A), before reaching the velocity specified (v).

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.

The v command value may be used in variable (VARI) assignments, and in IF and WAIT conditional statements. In addition, VARI variables may be substituted for the v command value.

**UNITS OF MEASURE and SCALING:** refer to page 16.

**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:**

```

SCALE1           ; Enable scaling
SCLA25000        ; Set the acceleration scaling factor to
                 ; 25000 counts/unit/unit
SCLV25000        ; Set the velocity scaling factor to 25000 counts/unit
SCLD1           ; Set the distance scaling factor to 1 count/unit
DEL proge        ; Delete program called proge
DEF proge        ; Begin definition of program called proge
MA0             ; Incremental positioning mode
MCO             ; Preset positioning mode
A10             ; Set the acceleration to 10 units/sec/sec
V1             ; Set the velocity to 1 units/sec
D100000         ; Set the distance to 100000 units
GO1             ; Initiate motion
END             ; End definition of proge

```

---

**[ V ] Velocity (Programmed) Assignment**

Type	Assignment or Comparison	Product	Rev
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	GO, SCALE, SCLV, V, [ VEL ]		

---

The velocity assignment operator (v) 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=v where “n” is the variable number, or v can be used in an expression such as IF (V<25)

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 or VELA command.

**UNITS OF MEASURE and SCALING:** refer to page 16.

**Example:**

```

IF (V<25)        ; If the programmed velocity is less than 25
                 ; units/sec, then do the statements between the IF and NIF
VAR1=V*2         ; Variable 1 = programmed velocity times 2
V(VAR1)          ; Set the velocity to the value of variable 1
NIF              ; End the IF statement

```

---

**VAR Numeric Variable Assignment**

Type	Variable	Product	Rev
Syntax	<a_><!>VAR<i><=r>	GT6K	6.0
Units	i = variable number r = number or expression	GV6K	6.0
Range	i = 1-225 r = ±999,999,999.99999999		
Default	n/a		
Response	VAR1: *VAR1=+0.0		
See Also	DVAR, VARB, VARCLR, VARI, VCVT, 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).

All variables (numeric [VAR], integer [VARI], binary [VARB], and string [VARS]) are automatically stored in battery-backed RAM.

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) is legal. Indirect variable assignments are also legal; (e.g., VAR(VAR1)=5 or VAR(VAR2)=VAR(VAR4)).

**Rule of Thumb for command value substitutions:** If the command syntax shows that the command field requires a real number (denoted by <r>) or an integer value (denoted by <i>), you can use the VAR substitution.

**Example:**

```
VAR1=2*PI           ; Set Variable 1 to 2 pi
D(VAR2)             ; Set the distance value equal to variable 2
```

**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:**

```
VAR51 = 1           ; Set Variable 51 to 1
REPEAT              ; Begin repeat/until loop
VAR(VAR51) = 0      ; Clear variables (e.g., if VAR51 = 8,
                    ; then VAR(VAR51)=0 is equivalent to VAR8=0)
VAR51 = VAR51 + 1   ; Increment counter
UNTIL (VAR51 = 51) ; End repeat/until loop
```

---

## VARB Binary Variable Assignment

Type	Variable	Product	Rev
Syntax	<a_><!>VARB<i><=bb...bbb> (32 bits) bit select syntax: VARB<i>.<bit #>-<b>	GT6K	6.0
Units	i = variable number	GV6K	6.0
Range	i = 1-125 b = 0, 1, X, or x		
Default	n/a		
Response	VARB1: *VARB1=XXXX_XXXX_XXXX_XXXX_XXXX_XXXX_XXXX_XXXX		
See Also	DVARB, PLCP, VAR, VARI, VARCLR, 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.

All variables (numeric [VAR], integer [VARI], binary [VARB], and string [VARS]) are automatically stored in battery-backed RAM.

**Example:** VARB1=b111100001111XXXX11110000xxxx1111  
Notice that the letter “b” is required. The “b” signifies binary: 1’s, 0’s, and X’s only.

**Example:** VARB1.14-1  
This is a shortcut to change only the state of VARB1 bit #14 to “1”.

**Example:** VARB1=h7F4356A3  
Notice that the letter “h” is required. The “h” signifies hexadecimal: 0-9, A-F only.

Binary variables are also used in conjunction with bitwise operators (&, |, ^, and ~).

**Example:** VARB1=VARB2 | VARB3 & b1111000011001

The expression must be less than 80 characters in length, including the (VARB1=b or VARB1=h) part of the expression.

**NOTE:** When assigning a binary variable, any bit set to “x” remains “x” until set to “1” or “0”. Any bit that is unspecified is set to “x”.

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 more than 4 bits of binary information; therefore, the command OUT(VARB1) is legal.

**Rule of Thumb for command value substitutions:** If the command syntax shows that the command field requires a binary value (denoted by <b>), you can use the VARB substitution.

**Example:**

```
VARB1=b1110 & hA      ; Binary variable 1 is set to binary 1110 bitwise
                      ; "AND"ed with hexadecimal A
VARB1=IN.7            ; State of onboard input bit 7 assigned to binary variable 1
OUT(VARB2)            ; State of all onboard outputs assigned by binary variable 2
```

---

## VARCLR Variable Clear

Type	Variable	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>VARCLR	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	DVAR, DVARI, DVARB, VAR, VARB, VARI, VARS		

---

VARCLR resets all numeric variables (VAR), integer variables (VARI), binary variables (VARB), and string variables (VARS) to their factory default values:

- Numeric (VAR) variables are set to 0.0
- Integer (VARI) variables are set to 0
- Binary (VARB) variables are set to bxxxx\_xxxx\_xxxx\_xxxx\_xxxx\_xxxx\_xxxx
- String (VARS) variables are set to " "

---

## VARI Integer Variable Assignment

Type	Variable	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>VARI<i><=>i>	GT6K	6.0
Units	1 <sup>st</sup> i = variable number 2 <sup>nd</sup> i = integer number	GV6K	6.0
Range	1 <sup>st</sup> i = 1-225 2 <sup>nd</sup> i = -2,147,483,647 to +2,147,483,647		
Default	n/a		
Response	VARI1: *VARI1=+Ø		
See Also	DVARI, INVARI, VARB, VAR, VARCLR, VCVT, WRVARI		

---

Integer variables can be used to store integer number values, with a range from -2,147,483,647 to +2,147,483,647. The information is assigned to the variable with the equal sign (e.g., VARI1=32).

All variables (numeric [VAR], binary [VARB], integer [VARI], and string [VARS]) are automatically stored in battery-backed RAM.

Integer variables can be used with mathematical (=, +, -, \*, /) and bitwise operators (&, |, ^, ~). For example, VARI1=(3+4-7\*4/4+3-2/2)\*3. Numeric (VAR) and integer (VARI) variables can be mixed in the

mathematical expressions. The results, if fractional, are truncated. **NOTE:** VARI cannot be used with trigonometric operators (ATAN, COS, PI, SIN, TAN) and square root (SQRT).

Each variable expression must be less than 80 characters in length, including the VARI= 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., VARI1=READ1). Setting an integer variable to a real number results in a truncation.

All integer 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(VARI1) is legal. Indirect variable assignments are also legal; (e.g., VARI(VARI1)=5 or VARI(VARI2)=VARI(VARI4)).

Integer variables should be used whenever possible to allow faster math operation than numeric variables (VAR).

Mapping inputs to a VARI variable: It is possible to map the binary value of a group of inputs to an integer variable. For details, refer o the INVARI command.

**Rule of Thumb for command value substitutions:** If the command syntax shows that the command field requires a real number (denoted by <f>) or an integer value (denoted by <i>), you can use the VARI substitution.

**Example:**

```
VARI1=2*3           ; Set Variable 1 to 6
D(VARI2)           ; Set the distance value equal to integer variable 2
```

**Indirect Variables:** Integer variables can be used indirectly. Only one level of indirection is possible (e.g., VARI(VARI(VARIn)) is not a legal command). The example below shows how indirect variables are used to clear 50 variables (from 1 to 50).

**Example:**

```
VARI51 = 1           ; Set Integer Variable 51 to 1
REPEAT              ; Begin repeat/until loop
VARI(VARI51) = 0    ; Clear variables (e.g., if VARI51 = 8, then
                    ; VARI(VARI51)=0 is equivalent to VARI8=0)
VARI51 = VARI51 + 1 ; Increment counter
UNTIL (VARI51 = 51)
```

<b>VARS</b>		<b>String Variable Assignment</b>	
Type	Variable	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>VARS<i><="message">	GT6K	6.0
Units	i = variable number message = text string	GV6K	6.0
Range	i = 1-25 Message = up to 50 characters		
Default	n/a		
Response	VARS1: *VARS1="HI JOHN"		
See Also	' , [ \ ], EOT, [ READ ], VAR, VARB, VARCLR, VARI, WRITE, WRVARS		

String variables can be assigned a character string up to 50 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.

All variables (numeric [VAR], integer [VARI], binary [VARB], and string [VARS]) are automatically stored in battery-backed RAM.

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 VAR1="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

You can copy one VARS to another VARS. VARSn=VARSm may be used, as well as variable substitution for "n" or "m".

**Example:**

```
VAR1="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.
; Example of data entry is to type '!10.0', which
; will assign numeric variable 2 the value 10.00
```

---

## VCVT Variable Type Conversion

Type	Operator (Mathematical)	<b>Product</b>	<b>Rev</b>
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	VAR, VARB, VARI		

---

Using VCVT operator, you can convert numeric (VAR or VARI) values to binary (VARB) 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 (∅).

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.

You can also convert real (VAR) values to integer (VARI) values (real values are truncated in the process).

**NOTE:** Numeric variables (VAR) have insufficient range to convert a full 32-bit binary variable (VARB). For example, executing the VARB1=h00000004 command and then the VAR1=VCVT(VARB1) command yields an INVALID DATA error.



#### Numeric-to-Binary Conversion:

```
VAR1=-5          ; Set numeric variable value = -5
VARB1=VCVT(VAR1) ; Convert the numeric value to a binary value and
                 ; store in VARB1
VARB1           ; Display value of VARB1. The response should be:
                 ; *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           ; Display value of VARB1. The response should be:
                 ; *VARB1=1001_1000_0000_0000_0000_0000_0000_0000
```

#### Binary-to-Numeric Conversion:

```
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 (Commanded) Assignment

Type	Assignment or Comparison	Product	Rev
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	SCALE, SCLV, SFB, TVEL, TVELA, V, [ V ], VELA		

---

Use the VEL operator to compare the current *commanded* velocity to another value or variable, or to assign the current commanded velocity to a variable. The velocity value used in any comparison, or in any assignment statement, is the current commanded velocity value, not the *programmed* velocity (V) or the actual velocity as measured from the feedback device (VELA).

**Syntax:** VARn=VEL where “n” is the variable number, or VEL can be used in an expression such as IF(VEL>4).

The VEL value represents the current commanded velocity. It is not the programmed velocity (V). If scaling is enabled (SCALE1), the VEL value is scaled by the velocity scaling factor (SCLV).

**GT6K Steppers:** If scaling is disabled (SCALE0), the value is measured in revolutions/sec (actual velocity in commanded counts/sec divided by the drive resolution DRES value).

**GV6K Servos:** If scaling is disabled (SCALE0), the value is measured in encoder revs/sec or resolver revs/sec.

**Example:**

```
IF(VEL<25)      ; If the current velocity is less than 25 units/sec,
                 ; then do the statements between the IF and NIF
    VAR1=V*2     ; Variable 1 = programmed velocity times 2
NIF             ; End the IF statement
```

---

## [ VELA ]      Velocity (Actual) Assignment

Type	Assignment or Comparison	<b>Product</b>	<b>Rev</b>
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	ERES, SCLV, TVEL, TVELA, [V], V, [VEL]		

---

The VELA operator is used to compare the current *actual* velocity (as derived from the feedback device) to another value or variable, or to assign the current velocity to a variable. If the programmed velocity information is required, refer to the [V] operator; if the current commanded velocity information is required, refer to the [VEL] operator.

The sign determines the direction of motion. You can use the VELA operator 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.

**Syntax:**    VARn=VELA where “n” is the variable number, or VELA can be used in an expression such as IF(VELA>4).

### Units of Measure:

**GT6K Steppers:** The velocity is always revs/sec (actual velocity in counts/sec multiplied by the ERES value if in ENCCNT1 mode, or multiplied by DRES if in ENCCNT0 mode).

**GV6K Servos:** If scaling is enabled (SCALE1), the velocity value will be scaled by the velocity scaling factor (SCLV). If scaling is not enabled (SCALE0), the value returned will be in encoder revs/sec or resolver revs/sec.

### Example:

```
IF(VELA<25)            ; If the current velocity is less than 25 units/sec,  
                      ; then do the statements between IF and NIF  
    VAR1=V*2            ; Variable 1 = programmed velocity times 2  
NIF                    ; End the IF statement
```

---

## VF                      Final Velocity

Type	Compiled Motion	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>VF<r>	GT6K	6.0
Units	n/a	GV6K	6.0
Range	0 (non-zero values result in error message)		
Default	0		
Response	n/a		
See Also	FOLRN, FOLRD, FOLMAS, FOLMD, GOBUF, SCLD, FOLEN, V		

---

The VF command designates that the motor will move the load the programmed distance in a preset GOBUF segment, completing the move at a final speed of zero. VF applies only to the next (subsequent) GOBUF, which marks an intermediate “end of move” within a profile. VF is used only in conjunction with the GOBUF command. Normal preset GO moves always finish with zero velocity.

The VF command remains in effect for the affected axis until a GOBUF is executed on that axis, or until you issue a RESET command.

Any non-zero value that is entered for VF will result in an immediate error message.
--

---

## [ VMAS ] Current Master Velocity

Type	Following; Assignment or Comparison	Product	Rev
Syntax	See below	GT6K	6.0
Units	n/a	GV6K	6.0
Range	n/a		
Default	n/a		
Response	n/a		
See Also	FFILT, FMCNEW, FMCP, FOLMAS, FOLMD, SCALE, SCLMAS, TVMAS		

---

The VMAS command is used to assign the master velocity value to a variable, or to make a comparison against another value. The master must be assigned first (FOLMAS command) before this command will be useful.

**Syntax:** VARn=VMAS where “n” is the variable number, or VMAS can be used in an expression such as IF (VMAS>1Ø).

The precision of the VMAS value is dependent upon the FFILT filter value.

If scaling is enabled (SCALE1), the velocity value is scaled by the master scaling factor (SCLMAS). If scaling is disabled (SCALEØ), the velocity value is in counts/sec.

**Example:**

```
IF (VMAS>4.3)      ; If the master is traveling at more than
                   ; 4.3 user units/sec then do the IF statement
    OUT.4=b1       ; Set onboard output #4 to 1
NIF                ; End of IF statement
VAR14=VMAS        ; Set VAR14 to master velocity
```

---

## WAIT( ) Wait for a Specific Condition

Type	Program Flow Control	Product	Rev
Syntax	<a_><!>WAIT(expression)	GT6K	6.0
Units	n/a	GV6K	6.0
Range	Up to 80 characters (including parentheses)		
Default	n/a		
Response	n/a		
See Also	FMCLen, FMCNEW, FMCP, GOWHEN, IF, NWHILE, REPEAT, [ SS ], T, TSS, UNTIL, WHILE		

---

The WAIT command is used to wait for a specific expression to evaluate true. No commands, except for immediate commands (a command preceded with !), 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. When a wait condition is pending, system status (TSS) bit #14 is set.

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, AS, D, ER, IN, INO, MOV, OUT, PC, PCE, PCMS, PE, PER, SS, TIM, US, V, VEL, etc.) can be used within the WAIT( ) expression.

**Example:**

```

MC1           ; Mode continuous
COMEXC1      ; Enable continuous command mode
GO1          ; Initiate motion
WAIT(IN=b1)  ; Wait for input 1 to be active
S1           ; Stop motion
WAIT(MOV=b0) ; Wait for motion complete
COMEXC0      ; Disable continuous command execution mode

```

---

## WHILE( ) WHILE Statement

Type	Program Flow Control; Conditional Branching	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>WHILE(expression)	GT6K	6.0
Units	n/a	GV6K	6.0
Range	Up to 80 characters (including parentheses)		
Default	n/a		
Response	n/a		
See Also	IF, JUMP, NWHILE, REPEAT, UNTIL, WAIT		

---

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.

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, AS, D, ER, IN, INO, MOV, OUT, PC, PCE, PCMS, PE, PER, SS, TIM, US, V, VEL, etc.) can be used within the WHILE( ) expression.

**Example:**

```

WHILE(IN=b1X0) ; While onboard input 1 = 1, input 3 = 0,
                ; 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
TPC           ; Transfer commanded position
NWHILE        ; End WHILE statement

```

---

## WRITE Write a Message

Type	Communication Interface	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>WRITE"<message>"	GT6K	6.0
Units	n/a	GV6K	6.0
Range	Up to 69 characters (may not use ", ; or :)		
Default	Messages are written to COM1 by default NOTE: "COM1" is the "RS-232/485" or "ETHERNET" connector. "COM2" is the "RS-232" connector.		
Response	WRITE"message": message		
See Also	[ \ ], EOT, PORT, [ READ ], VARS, WRVAR, WRVARB, WRVARS		

---

The WRITE command provides an efficient way of transmitting message strings to the Ethernet port and the RS-232 or RS-485 ports. 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
:	Colon	58
;	Semi-colon	59
\	Backslash	92

---

**NOTE:** All text strings are transmitted in UPPER CASE. If you must transmit lower-case strings, use the ASCII decimal equivalent for each character (see ASCII table in [Appendix B](#)). For example, WRITE"\106\117\110\107" will transmit the text string "junk" in lower-case letters.

**NOTE:** Upon power up or after a reset, by default the WRITE command writes messages to COM1. See the PORT command for instructions on writing messages to COM2. If you want to automatically write messages to COM2 after power up or a reset, consider putting the PORT command in your STARTP program. See the STARTP command for more information.

**Example:**

```
WRITE"It's a wonderful life!" ; Send the message "IT'S A WONDERFUL LIFE!"
```

---

## WRVAR Write a Numeric Variable

Type	Communication Interface	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>WRVAR<i>	GT6K	6.0
Units	i = variable number	GV6K	6.0
Range	i = 1-225		
Default	n/a		
Response	WRVAR1: +0.0		
See Also	EOT, [ READ ], VAR, WRITE, WRVARB, WRVARI, WRVARS		

---

Use the WRVAR command to transfer a specific numeric variable (VAR) to the Ethernet port and the RS-232C or RS-485 ports. Only the value and the EOT command characters are transmitted.

### Example:

```
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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>WRVARB<i>	GT6K	6.0
Units	i = variable number	GV6K	6.0
Range	i = 1-125		
Default	n/a		
Response	WRVARB1: XXXX_XXXX_XXXX_XXXX_XXXX_XXXX_XXXX_XXXX		
See Also	EOT, [ READ ], VARB, WRITE, WRVAR, WRVARI, WRVARS		

---

Use the WRVARB command to transfer a specific binary variable (VARB) to the Ethernet port and the RS-232C or RS-485 ports. Only the binary value and the EOT command characters are transmitted.

### Example:

```
VARB1=b1101      ; Set binary variable 1 to 1101
WRVARB1          ; Transmit binary variable 1
                  ; (value transmitted =1101_XXXX_XXXX_XXXX_XXXX_XXXX_XXXX_XXXX)
```

---

## WRVARI Write an Integer Variable

Type	Communication Interface	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>WRVARI<i><=<i>	GT6K	6.0
Units	1 <sup>st</sup> i = variable number 2 <sup>nd</sup> i = integer number	GV6K	6.0
Range	1 <sup>st</sup> i = 1-225 2 <sup>nd</sup> i = -2,147,483,647 to +2,147,483,647		
Default	n/a		
Response	WRVARI1: +0		
See Also	EOT, [ READ ], VARI, WRITE, WRVAR, WRVARB, WRVARS		

---

Use the WRVARI command to transfer a specific integer variable (VARI) to the Ethernet port and the RS-232C or RS-485 ports. Only the integer value and the EOT command characters are transmitted.

Integer variables should be used whenever possible to allow faster math operation than numeric variables (VAR).

### Example:

```
VAR11=100        ; Set integer variable 1 equal to 100
WRVARI1          ; Transmit integer variable 1 (the value +100 is transmitted)
```

---

## WRVARS Write a String Variable

Type	Communication Interface	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>WRVARS<i>	GT6K	6.0
Units	i = variable number	GV6K	6.0
Range	i = 1-25		
Default	n/a		
Response	WRVARS1: No response until a string is placed in VARS1		
See Also	EOT, [ READ ], VARS, WRITE, WRVAR , WRVARB, WRVARI		

---

Use the WRVARS command to transfer a specific string variable (VARS) to the Ethernet port and the RS-232C or RS-485 ports. Only the string and the EOT command characters are transmitted.

**NOTE:** All text strings are transmitted in UPPER CASE. If you must transmit lower-case strings, use the ASCII decimal equivalent for each character (see ASCII table in [Appendix B](#)). For example, if you entered VARS1="\106\117\110\107", the WRVARS1 command will transmit the text string "junk" in lower-case letters.

**Example:**

```
VAR1="John L" ; Set string variable 1 = "John L"
WRVARS1      ; Transmit string variable 1 (string "JOHN L" is transmitted)
```

---

## XONOFF Enable/Disable XON / XOFF

Type	Communication Interface	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>XONOFF<b>	GT6K	6.0
Units	b = enable bit	GV6K	6.0
Range	0 (disable) or 1 (enable)		
Default	1 for COM1, 0 for COM2 (PORT command setting determines which COM port's XONOFF setting is checked)		
Response	XONOFF *XONOFF1		
See Also	], [, BOT, DRPCHK, E, EOT, ERBAD, ERROK, LOCK, PORT		

---

Use the XONOFF command to enable or disable XON/XOFF (ASCII handshaking).

XONOFF1 enables XON/XOFF, which allows the Gem6K to recognize ASCII handshaking control characters. When XON/XOFF is enabled, ASCII 17 or ^Q is a signal to start sending characters; ASCII 19 or ^S is a signal to stop sending characters. XONOFF0 disables XON/XOFF.

The PORT command determines which COM port is affected by the XONOFF command. Each port will track its XON/XOFF values

**RS-485 Multi-drop:** If you are using RS-485 multi-drop, disable XON/XOFF by executing the PORT1 command followed by the XONOFF0 command.

**NOTE:** COM1 is the "RS-232/485" or "ETHERNET" connector; COM2 is the "RS-232" connector.





# Appendix A: Gem6K Command List

(Firmware Revision 6.0)

Command	Description	Command	Description
%	Task Identifier	ANO	Set Analog Output Voltage Value
[ ! ]	Immediate Command Identifier	[ ANO ]	Analog Output Voltage Value [operator]
[ @ ]	Global Command Identifier	[ AS ]	Axis Status [operator]
;	Begin Comment	[ ASX ]	Axis Status, Extended [operator]
\$	Label Declaration	[ ATAN( ) ]	Arc Tangent [operator]
[ # ]	Step Through a Program	BAUD	Baud Rate
'	Enter Interactive Data	BOT	Beginning of Transmission Characters
[ . ]	Bit Select	BP	Set a Program Break Point
[ " ]	Begin and End String	BREAK	Terminate Program Execution
[ \ ]	ASCII Character Designator	C	Continue Command Execution
[ = ]	Assignment or Equivalence	CERRLG	Clear Error Log
[ > ]	Greater Than	COMEXC	Continuous Command Processing Mode
[ >= ]	Greater Than or Equal	COMEXL	Continue Execution on End-of-Travel Limit
[ < ]	Less Than	COMEXR	Continue Motion on Pause/Continue Input
[ <= ]	Less Than or Equal	COMEXS	Continue Execution on Stop
[ <> ]	Not Equal	[ COS( ) ]	Cosine [operator]
[ ( ) ]	Operation Priority Level	D	Distance
[ + ]	Addition	[ D ]	Distance [operator]
[ - ]	Subtraction	DABSD	Enable ABS Damping
[ * ]	Multiplication	DACTDP	Active Damping
[ / ]	Division	[ DAT ]	Data Assignment [operator]
[ & ]	Boolean And	DATA	Data Statement
[   ]	Boolean Inclusive Or	[ DATP ]	Data Program
[ ^ ]	Boolean Exclusive Or	DATPTR	Set Data Pointer
[ ~( ) ]	Boolean Not	DATRST	Reset Data Pointer
[ << ]	Shift from Right to Left (bit 32 to bit 1)	DATSIZ	Data Program Size
[ >> ]	Shift from Left to Right (bit 1 to bit 32)	DATTC	Data Teach
[	Send Response to Both COM Ports	DAUTOS	Auto Current Standby
]	Send Response to Alternate COM Port	DCLEAR	Clear RP240 Display
A	Acceleration	DCLRLR	Clear Latched Status Register Bits
[ A ]	Acceleration [operator]	DDAMPA	Damping During Acceleration/Deceleration
AA	Acceleration, S-curve	DEF	Begin Definition of Program
AD	Deceleration	DEL	Delete Program
[ AD ]	Deceleration [operator]	DELVIS	Electronic Viscosity
ADA	Deceleration, S-curve	DIBW	Current Loop Bandwidth
ADDR	Auto-Address Multiple Serial Units	DIFOLD	Current Foldback Enable
[ AND ]	And [operator]	DIGN	Current Loop Gain
[ ANI ]	Analog Input Voltage [operator]	DJOG	Enable RP240 Jog Mode
ANIEN	Analog Input Enable	[ DKEY ]	Value of RP240 Key
ANIMAS	Assign Analog Input as Master	DLED	Turn RP240 LEDs ON/OFF
ANIRNG	Analog Input Voltage Range	DMEPIT	Motor Electrical Pitch

Command	Description	Command	Description
DMODE	Drive Control Mode	ENCCNT	Encoder Count Reference Enable
DMONAS	Analog Monitor Output A – Scaling	END	End Definition of Program
DMONAV	Analog Monitor Output A – Variable	EOL	End-of-Line Termination Characters
DMONBS	Analog Monitor Output B – Scaling	EOT	End-of-Transmission Characters
DMONBV	Analog Monitor Output B – Variable	[ ER ]	Error Status [operator]
DMTAMB	Motor Ambient Temperature	ERASE	Erase All Programs
DMTD	Motor Damping	ERES	Encoder Resolution
DMTIC	Continuous Current	ERRBAD	Error Prompt Characters
DMTICD	Continuous Current Derating	ERRDEF	Program Definition Prompt Characters
DMTIND	Motor Inductance	ERRLVL	Error Detection Level
DMTIP	Peak Current	ERROK	Good Prompt Characters
DMTJ	Motor Rotor Inertia / Forcer Mass	ERROR	Enable Error Checking
DMTKE	Motor Ke	ERRORL	Error Log Selection
DMTLIM	Torque/Force Limit	ERRORP	Assign an Error Program
DMTLMN	Minimum Motor Inductance	ESDB	Stall Backlash Deadband
DMTLMX	Maximum Motor Inductance	ESK	Kill on Stall
DMTMAX	Maximum Motor Winding Temperature	ESTALL	Enable Encoder Stall Detection
DMTR	Identify (and Load) Motor	EXE	Execute Program from a Compiled Program
DMTRES	Motor Winding Resistance	[ FB ]	Value of Feedback Device [operator]
DMTRWC	Motor Winding Thermal Resistance	FFILT	Following Filter
DMTSCS	Torque/Force Scaling	FGADV	Following Geared Advance
DMTSTT	Motor Static Torque	FLTDSB	Fault on Drive Disable (DRIVE0)
DMTTCM	Motor Thermal Time Constant	FMAXA	Follower Axis Maximum Acceleration
DMTTCW	Motor Winding Time Constant	FMAXV	Follower Axis Maximum Velocity
DMTW	Motor Rated Speed	FMCLEN	Master Cycle Length
DMVLIM	Velocity Limit	FMCNEW	Restart Master Cycle Counting
DNOTAD	Notch Filter A Depth	FMCP	Initial Master Cycle Position
DNOTAF	Notch Filter A Frequency	FOLEN	Enable Following Mode
DNOTAQ	Notch Filter A Quality Factor	FOLK	Following Kill, Limitations
DNOTBD	Notch Filter B Depth	FOLMAS	Assignment of Master to Follower
DNOTBF	Notch Filter B Frequency	FOLMD	Master Distance
DNOTBQ	Notch Filter B Quality Factor	FOLRD	Denominator of Follower-to-Master Ratio
DNOTLD	Notch Lead Filter Break Frequency	FOLRN	Numerator of Follower-to-Master Ratio
DNOTLG	Notch Lag Filter Break Frequency	FOLRNF	Numerator of Final Follower-to-Master Ratio
DPASS	Change RP240 Password	FPPEN	Enable Master Position Prediction
DPBW	Position Loop Bandwidth	[ FS ]	Following Status [operator]
DPCUR	Position RP240 Display Cursor	FSHFC	Continuous Shift
DPHBAL	Phase Balance	FSHFD	Preset Shift
DPHOFA	Phase A Current Offset	FVMACC	Virtual Master Count Acceleration
DPHOFB	Phase B Current Offset	FVMFRQ	Virtual Master Count Frequency
DPOLE	Number of Motor Pole Pairs	GO	Initiate Motion
[ DPTR ]	Data Pointer Location [operator]	GOBUF	Store a Compiled Motion Segment
DPWM	Drive PWM Frequency	GOSUB	Call a Subroutine
[ DREAD ]	Read RP240 Numeric Data [operator]	GOTO	Go To a Program or Label
[ DREADF ]	Read RP240 Function Key [operator]	GOWHEN	Conditional Go
DREADI	RP240 Data Read, Immediate Mode	HALT	Terminate Program Execution
DRES	Drive Resolution	HELP	Technical Support Phone Numbers
DRESET	Drive Reset	HOM	Initiate Homing Operation
DRIVE	Drive Enable/Disable	HOMA	Homing Acceleration
DRPCHK	RP240 COM Port Check	HOMAA	Homing Acceleration, S-curve
DSTALL	Enable/Disable Drive Stall Detection	HOMAD	Homing Deceleration
DSTP	Enable/Disable RP240 Stop Key	HOMADA	Homing Deceleration, S-curve
DVAR	Display Numeric Variable on RP240	HOMBAC	Backup to Home
DVARB	Display Binary Variable on RP240	HOMDF	Homing Final Direction
DVARI	Display Integer Variable on RP240	HOMEDG	Home Reference Edge
DWAVEF	Waveform	HOMV	Homing Velocity
DWRITE	Write Text to RP240	HOMVF	Homing Velocity, Final Approach
E	Enable Serial Communication	HOMZ	Home to Encoder Z Channel
ECHO	Enable Communication Echo	IF ( )	IF Statement
ELSE	Else Condition of IF Statement	IN	Virtual Input Override

Command	Description	Command	Description
[ IN ]	Input Status [operator]	MESND	Master Encoder Step & Direction Mode
INDEB	Input Debounce Time	[ MOV ]	Axis Moving Status [operator]
INDUSE	Enable User Status	NIF	End IF Statement
INDUST	User Status Definition	[ NMCY ]	Master Cycle Number Status [operator]
INEN	Enable Specific Inputs	[ NOT ]	Not [operator]
INFNC	Input Function Assignment	NTADDR	Set IP Address
INLVL	Input Active Level	NTCONN	Network Connect
[ INO ]	Other Inputs (Enable Input) Status [operator]	NTFEN	Ethernet Communication Enable
INPLC	Establish PLC Data Inputs	NTID	Network Sharing Unit ID for Peer-to-Peer
INSELP	Enable Program Selection via Inputs	NTIO	Network I/O Configuration
INSTW	Establish Thumbwheel Data Inputs	NTIP	Network IP Address
INTHW	Enable Checking for Alarm Events	NTMASK	Ethernet Network Mask
INTSW	Force an Alarm Event	NTMPRB	Network Map Binary Variables for Reading
INVARI	Map Inputs to Integer Variable	NTMPRI	Network Map Integer Variables for Reading
JOG	Enable Jog Mode	NTMPWB	Network Map Binary Variable for Writing
JOGA	Jog Acceleration	NTMPWI	Network Map Integer Variable for Writing
JOGAA	Jog Acceleration, S-curve	NTPOLL	Network Polling Rate
JOGAD	Jog Deceleration	NTRATE	Network Sharing Rate for Peer-to-Peer
JOGADA	Jog Deceleration, S-curve	[ NTS ]	Network Status
JOGVH	Jog Velocity, High	NTSELP	Network Select Program Enable
JOGVL	Jog Velocity, Low	NTSFS	Ethernet Send Fast Status Packet
JOY	Enable Joystick Mode	NTWRIT	Network (Ethernet) ASCII Write
JOYA	Joystick Acceleration	NWHILE	End of WHILE Statement
JOYAA	Joystick Acceleration, S-curve	ONCOND	Enable Program Interrupt ("On") Conditions
JOYAD	Joystick Deceleration	ONIN	On an Input Condition GOSUB
JOYADA	Joystick Deceleration, S-curve	ONP	On Condition Program Assignment
JOYAXH	Joystick Analog Channel, High	ONUS	On a User Status Condition GOSUB
JOYCDB	Joystick Center Deadband	ONVARA	On Numeric Variable 1 Condition GOSUB
JOYCTR	Joystick Center	ONVARB	On Numeric Variable 2 Condition GOSUB
JOYEDB	Joystick End Deadband	[ OR ]	Or [operator]
JOYVH	Joystick Velocity, High	ORES	Output Resolution
JOYVL	Joystick Velocity, Low	OUT	Activate Programmable Outputs
JOYZ	Joystick Zero (Center)	[ OUT ]	Programmable Outputs Status [operator]
JUMP	Jump to Program or Label (No Return)	OUTALL	Activate Programmable Outputs, Range
K	Kill Motion	OUTBD	Brake Output Delay
<ctrl>K	Kill Motion	OUTEN	Disable Programmable Outputs
KDRIVE	Disable Drive on Kill	OUTFNC	Programmable Output Function Assignment
KIOEN	Kill on EVM32 Disconnect	OUTLVL	Programmable Output Active Level
L	Loop	OUTPA	Output on Position
LDAMP	Load Damping	OUTPLC	Establish PLC Strobe Outputs
LH	Enable Hardware End-of-Travel Limits	OUTTW	Establish Thumbwheel Strobe Outputs
LHAD	Hardware EOT Limits Deceleration	[ PC ]	Position Commanded [operator]
LHADA	Hardware EOT Limits Decel, S-curve	[ PCC ]	Captured Commanded Position [operator]
[ LIM ]	Hardware EOT & Home Limit Inputs, Status	[ PCE ]	Position of Captured Encoder [operator]
LIMEN	Limit Input Enable	[ PCME ]	Position of Captured Master Encoder [operator]
LIMFNC	Limit Input Function Assignment	[ PCMS ]	Position of Captured Master Cycle [operator]
LIMLVL	Hardware EOT & Home Inputs, Active Level	PCOMP	Compile a Profile or Program
LJRAT	Load-to-Rotor/Forcer Inertia/Mass Ratio	[ PE ]	Position of Encoder [operator]
LN	End of Loop	[ PER ]	Position Error [operator]
LOCK	Lock Resource to a Task	PESET	Set Encoder Absolute Position (steppers)
LS	Enable Software End-of-Travel Limits	PEXE	Execute a Compiled Program
LSAD	Software EOT Limits, Deceleration	[ PI ]	Pi ( $\pi$ ) [operator]
LSADA	Software EOT Limits Decel, S-curve	PLCP	Compiled PLC Program
LSNEG	Negative-Direction Software EOT Limit	PLN	End of Loop, Compiled Motion
LSPOS	Positive-Direction Software EOT Limit	PLOOP	Start of Loop, Compiled Motion
LX	Terminate Loop	[ PMAS ]	Current Master Cycle Position [operator]
MA	Enable Absolute/Incremental Positioning	[ PME ]	Position of Master Encoder [operator]
MC	Enable Continuous/Preset Positioning	PMECLR	Clear Master Encoder Absolute Position
MEMORY	Partition Product Memory	PMESET	Set Master Encoder Absolute Position
MEPOL	Master Encoder Polarity	PORT	Designate Destination COM Port

Command	Description	Command	Description
POUTA	Compiled Output (Compiled Motion)	TASF	Transfer Axis Status (full-text report)
PRUN	Run a Compiled Profile	[ TASK ]	Task Number Assignment [operator]
PS	Pause Program Execution	TASX	Transfer Axis Status, Extended
PSET	Establish Absolute Position Reference	TASXF	Transfer Axis Status, Extended (full-text)
[ PSHF ]	Net Position Shift Status [operator]	TCMDER	Transfer Command Error
[ PSLV ]	Commanded Follower Position [operator]	TCS	Transfer Configuration Status
PUCOMP	Un-Compile a Compiled Profile	TDHRS	Transfer Operating Hours
RADIAN	Specify Units in Radians or Degrees	TDICNT	Transfer Continuous Current Rating
RE	Enable Registration	TDIMAX	Transfer Maximum Current Rating
[ READ ]	Read a Value	TDIR	Transfer Program Directory
REG	Registration Distance	TDPTR	Transfer Data Pointer Status
REGLD	Registration Lockout Distance	TDTEMP	Transfer Drive Temperature
REGSS	Registration Single-Shot	TDVBUS	Transfer Bus Voltage
REPEAT	REPEAT Statement	TER	Transfer Error Status
RESET	Reset the Gem6K Drive	TERF	Transfer Error Status (full-text report)
RFS	Return to Factory Settings	TERRLG	Transfer Error Log
RUN	Begin Executing a Program	TEX	Transfer Program Execution Status
S	Stop Motion	TFB	Transfer Position of Feedback Devices
[ SC ]	Controller Status [operator]	TFS	Transfer Following Status
SCALE	Enable Scaling Factors	TFSF	Transfer Following Status (full-test report)
[ SCAN ]	PLC Scan Runtime [operator]	TGAIN	Transfer Servo Gains
SCANP	Scan a Compiled PLC Program	THALL	Transfer Hall Sensor Values
SCLA	Acceleration Scale Factor	[ TIM ]	Current Timer Value [operator]
SCLD	Distance Scale Factor	TIMINT	Timer Value to Cause an Alarm Event
SCLMAS	Master Axis Scale Factor	TIMST	Start Timer
SCLV	Velocity Scale Factor	TIMSTP	Stop Timer
[ SEG ]	Number of Free Segment Buffers [operator]	TIN	Transfer Programmable Input Status
SFB	Select Servo Feedback Source	TINO	Transfer Other Input Status
SGAF	Gain – Acceleration Feedforward	TINOF	Transfer Other Input Status (full-text report)
SGENB	Enable a Servo Gain Set	TIO	Transfer Expansion I/O Status
SGINTE	Integrator Enable	TLABEL	Transfer Labels
SGIRAT	Current Damping Ratio	TLIM	Transfer Hardware Limit Status
SGPRAT	Position Damping Ratio	TMEM	Transfer Memory Usage
SGPSIG	Velocity/Position Bandwidth Ratio	TMTEMP	Transfer Motor Temperature
SGSET	Save a Servo Gain Set	TNMCY	Transfer Master Cycle Number
SGVF	Gain – Velocity Feedforward	TNT	Transfer Ethernet Status
SGVRAT	Velocity Damping Ratio	TNTMAC	Transfer Ethernet Address
SHALL	Hall Sensor Inversion	TNTS	Transfer Network Status
[ SIN ( ) ]	Sine [operator]	TNTSF	Transfer Network Status (full-text report)
SINAMP	Virtual Master Sine Wave Amplitude	TOUT	Transfer Programmable Output Status
SINANG	Virtual Master Sine Wave Angle	TPC	Transfer Commanded Position
SINGO	Virtual Master - Start Internal Sine Wave	TPCC	Transfer Captured Commanded Position
SMPER	Maximum Allowable Position Error	TPCE	Transfer Position of Captured Encoder
SMVER	Maximum Allowable Velocity Error	TPCME	Transfer Position of Captured Master Encoder
[ SQRT ]	Square Root [operator]	TPCMS	Transfer Position of Captured Master Cycle
SRSET	Resolver Offset Angle	TPE	Transfer Position of Encoder
[ SS ]	System Status [operator]	TPER	Transfer Position Error
STARTP	Start-up Program	TPMAS	Transfer Position of Master (current cycle)
STEP	Enable Single Step Mode	TPME	Transfer Position of Master Encoder
STRGTD	Target Zone Distance	TPRA	Transfer Absolute Resolver Position
STRGTE	Enable Target Zone Mode	TPROG	Transfer Program Contents
STRGTT	Target Zone Timeout Period	TPSHF	Transfer Net Position Shift
STRGTV	Target Zone Velocity	TPSLV	Transfer Commanded Position of Follower
[ SWAP ]	Task Swap Assignment [operator]	TRACE	Enable Program Trace Mode
T	Time Delay	TRACEP	Enable Program Flow Mode
TACC	Transfer Commanded Acceleration	TRANS	Enable Translation Mode
[ TAN ( ) ]	Tangent [operator]	TREV	Transfer Revision Level
TANI	Transfer ANI Analog Input Voltage	TRGFN	Trigger Functions
TANO	Transfer Analog Output Voltage	TRGLOT	Trigger Interrupt Lockout Time
TAS	Transfer Axis Status	[ TRIG ]	Trigger Interrupt Status [operator]

Command	Description
TSC	Transfer Controller Status
TSCAN	Transfer Scan Time of PLC Program
TSCF	Transfer Controller Status (full-text report)
TSEG	Transfer Number of Free Segment Buffers
TSGSET	Transfer Servo Gain Set
TSKAX	Task Axis Association for Multi-Tasking
TSKTRN	Task Turns Before Swapping
TSROFF	Transfer Resolver Offset Angle
TSS	Transfer System Status
TSSF	Transfer System Status (full-text report)
TSTAT	Transfer Drive Statistics
TSTLT	Transfer Settling Time
TSWAP	Transfer Currently Active Tasks
TTASK	Transfer Task Number
TTIM	Transfer Timer Value
TTRIG	Transfer Trigger Interrupt Status
TTRQ	Transfer Commanded Torque/Force
TTRQA	Transfer Actual Torque/Force
TUS	Transfer User Status
TVE	Transfer Velocity Error
TVEL	Transfer Current Commanded Velocity
TVELA	Transfer Current Actual Velocity
TVMAS	Transfer Current Master Velocity
[ TW ]	Thumbwheel Assignment [operator]
UNTIL ( )	Until Part of REPEAT Statement
[ US ]	User Status [operator]
V	Velocity
[ V ]	Velocity [operator]
VAR	Numeric Variable Assignment
VARB	Binary Variable Assignment
VARCLR	Clear All Variables
VARI	Integer Variable Assignment
VARS	String Variable Assignment
VARSHI	Shared Input Variable for Peer-to-Peer
VARSHO	Shared Output Variable for Peer-to-Peer
VCVT ( )	Variable Type Conversion
[ VEL ]	Commanded Velocity Assignment [operator]
[ VELA ]	Actual Velocity Assignment [operator]
VF	Final Velocity
[ VMAS ]	Velocity of Master [operator]
WAIT ( )	Wait for a Specific Condition
WHILE ( )	WHILE Statement
WRITE	Write a Message
WRVAR	Write a Numeric Variable
WRVARB	Write a Binary Variable
WRVARI	Write a Integer Variable
WRVARS	Write a String Variable
XONOFF	Enable XON/XOFF ASCII Handshaking
[ \ANI ]	Network Analog Input Voltage Status
\ANO	Network Analog Output
[ \ANO ]	Network Analog Output Status
[ \IN ]	Network Digital Input Status
\OUT	Network Digital Output
[ \OUT ]	Network Digital Output Status
\TANI	Transfer Network Analog Input Status
\TANO	Transfer Network Analog Output Status
\TIN	Transfer Network Digital Input Status
\TIO	Transfer Ethernet I/O Status
\TOUT	Transfer Network Digital Output Status

Command	Description
---------	-------------



# Appendix B: ASCII Table

DEC	HEX	CHAR
0	00	NUL
1	01	SOH
2	02	STX
3	03	EXT
4	04	EOT
5	05	ENQ
6	06	ACK
7	07	BEL
8	08	BS
9	09	HT
10	0A	LF
11	0B	VT
12	0C	FF
13	0D	CR
14	0E	SO
15	0F	S1
16	10	DLE
17	11	XON
18	12	DC2
19	13	XOFF
20	14	DC4
21	15	NAK
22	16	SYN
23	17	ETB
24	18	CAN
25	19	EM
26	1A	SUB
27	1B	ESC
28	1C	FS
29	1D	GS
30	1E	RSt
31	1F	US
32	20	SPACE
33	21	!
34	22	"
35	23	#
36	24	\$
37	25	%
38	26	&
39	27	`
40	28	(
41	29	)

DEC	HEX	CHAR
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	Ø
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S

DEC	HEX	CHAR
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_
96	60	'
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
100	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	
125	7D	}

DEC	HEX	CHAR
126	7E	~
127	7F	DEL
128	80	Ç
129	81	ü
130	82	é
131	83	â
132	84	ä
133	85	à
134	86	å
135	87	ç
136	88	ê
137	89	ë
138	8A	è
139	8B	ï
140	8C	î
141	8D	ì
142	8E	Ä
143	8F	Å
144	90	É
145	91	æ
146	92	Æ
147	93	ô
148	94	î
149	95	ò
150	96	û
151	97	ù
152	98	ÿ
153	99	Ö
154	9A	Ü
155	9B	ç
156	9C	£
157	9D	¥
158	9E	Pt
159	9F	f
160	A0	á
161	A1	í
162	A2	ó
163	A3	ú
164	A4	ñ
165	A5	Ñ
166	A6	ª
167	A7	º
168	A8	¿
169	A9	┌
170	AA	¬
171	AB	½

DEC	HEX	CHAR
172	AC	¼
173	AD	ı
174	AE	«
175	AF	»
176	B0	█
177	B1	█
178	B2	█
179	B3	ı
180	B4	┆
181	B5	≡
182	B6	≡
183	B7	┐
184	B8	≡
185	B9	≡
186	BA	
187	BB	┐
188	BC	┘
189	BD	┌
190	BE	┘
191	BF	┐
192	C0	└
193	C1	┘
194	C2	T
195	C3	┆
196	C4	-
197	C5	†
198	C6	‡
199	C7	‡
200	C8	┌
201	C9	┐
202	CA	≡
203	CB	≡
204	CC	≡
205	CD	=
206	CE	≡
207	CF	≡
208	D0	≡
209	D1	≡
210	D2	≡
211	D3	└

DEC	HEX	CHAR
212	D4	┌
213	D5	┐
214	D6	┐
215	D7	┆
216	D8	≡
217	D9	┐
218	DA	┌
219	DB	█
220	DC	█
221	DD	█
222	DE	█
223	DF	█
224	E0	α
225	E1	β
226	E2	Γ
227	E3	π
228	E4	Σ
229	E5	σ
230	E6	∞
231	E7	τ
232	E8	Φ
233	E9	θ
234	EA	Ω
235	EB	δ
236	EC	∞
237	ED	∅
238	EE	∈
239	EF	∩
240	F0	≡
241	F1	±
242	F2	≥
243	F3	≤
244	F4	ı
245	F5	
246	F6	÷
247	F7	≈
248	F8	°
249	F9	•
250	FA	•
251	FB	√
252	FC	η
253	FD	2
254	FE	•
255	FF	



# Index

---

## Operator Symbols

- 37  
' 31  
! 30  
! 28  
" 32  
# 30  
 $\pi$  (pi) 253  
\$ 29  
& 38  
( ) 36  
\* 37  
. (bit select operator) 31  
/ 38  
: 29  
@ 29  
\ 32  
^ 40  
| 39  
~() 41  
+ 37  
< 35  
<< 41  
<= 35  
> 36  
= 33  
> 34  
>= 34  
>> 42

---

## A

ABS damping 70  
absolute position  
  absolute positioning mode (MA1) 219  
  effect on distance 68  
  establishing 251, 261  
  effect on position report 138, 244, 245, 246, 247, 248, 250, 256, 318, 331, 332, 333, 334, 335, 337  
  master encoder  
    clear 257  
    establishing 257  
  resolver 337  
  zeroed after homing 164  
acceleration 45  
  assignment of 46  
  change on-the-fly 63, 220  
  commanded (TACC) 300  
  feedforward gain 285  
  jerk calculations 21

  maximum, follower axis 140  
  scaling 23  
  scaling factor (SCLA) 275, 279  
  s-curve profiling 19, 46  
    homing 167  
    jogging 190  
    joystick 196  
access 181, 213  
active damping 71  
actual feedback device position See position  
addition (+) 37  
address  
  Ethernet 330  
  IP 227  
address, auto-addressing units in a chain 49  
advance, geared (Following) 139  
alarm event  
  enable checking (INTHW) 186  
  force a condition (INTSW) 187  
  trigger with an input 180, 212  
  trigger with timer value 322  
analog input  
  ANI option See ANI  
  joystick 197, 198, 199, 200, 201  
  voltage range 199  
  voltage range selection 52  
analog monitor output A  
  scaling 86  
  variable 87  
analog monitor output B  
  scaling 88  
  variable 89  
AND (logical operator) 50  
ANI  
  as Following master (ANIMAS) 52  
  check input voltage 301  
  enable (ANIEN) 51  
  override (ANIEN) 51  
  voltage  
    status 51, 301, 325  
  voltage range section (ANIRNG) 52  
application examples  
  continuous phase shift 153  
  Following 143  
  GOWHEN 160  
  preset phase shift 155  
  scaling setup 24  
arc tangent 58, 264  
ASCII character designator (\) 32  
ASCII Table 379  
assignment of axis to tasks 346  
assignment of master and follower 146  
auto current standby 77  
autorun 86

axis assigned to task (TSKAX) 346  
axis moving status 224, 303  
axis scaling 22  
axis status 55, 302  
axis status, extended 56, 305, 306, 308

---

## B

backup to home (HOMBAC) 168, 169, 171  
baud rate, establish 58  
BCD program select input 178, 185, 211  
begin and end string (" ) 32  
begin comments (;) 29  
begin executing a program (RUN) 272  
begin program definition (DEF) 78  
beginning of transmission characters (BOT) 59  
binary value identifier (b) 5  
binary variable (VARB) 360  
  clearing 360  
  display of bits 41  
  display on RP240 116  
  writing 370  
bit select operator (.) 5, 31  
bitwise AND (&) 39  
bitwise exclusive OR (^) 40  
bitwise NOT (~) 41  
bitwise OR (|) 39  
Boolean And (&) 38  
Boolean Exclusive Or (^) 40  
Boolean Inclusive Or (|) 39  
Boolean Not (~) 41  
brake output delay (OUTBD) 238  
branching  
  branch to error program 129  
  ELSE 119  
  error program 132  
  GOSUB 158  
  GOTO 159  
  IF 172  
  JUMP 202  
  NIF 225  
  NWHILE 228  
  REPEAT 271  
  UNTIL 356  
  WHILE 368  
BREAK 60, 159  
break point (BP) 59  
buffered commands  
  looping (begin - L) 205  
  looping (end - LN) 215  
  looping, compiled 254, 255  
bus voltage, report 313

---

## C

- call a subroutine (GOSUB) 158
- carriage return
  - command delimiter 5
  - transmission character 121
- case sensitivity 5
- characters
  - command delimiters 5
  - comment delimiter 5, 29
  - field separators 5
  - limit per line 5
  - neutral (spaces) 5
- clear display (DCLEAR) 77
- clear error condition 131
- clear variables (VARCLR) 360
- clearing latched register bits 77
- COM port
  - enable/disable (E) 118
  - function, setup 114
  - lock to a task 215
  - selection (PORT) 258
- COMM6SRVR 7
- commanded acceleration (TACC) 300
- commanded acceleration, feedforward gain 285
- commanded position 251, 336
  - capture 180, 245, 332
  - comparison or assignment 244
  - display 331
  - follower
    - assignment/comparison 263
    - transfer 339
- commands
  - buffered 28
    - looping 215
    - looping, compiled 254, 255
  - command buffer execution
    - after end-of-travel limit (COMEXL) 65
    - after pause/continue input (COMEXR) 65
    - after stop (COMEXS) 67
    - continuous (COMEXC) 63
  - command description format 3
  - command field symbols 4
  - command list 373
  - command value substitutions 6
  - command-to-product compatibility 3
  - default settings 3
  - delimiters 5
  - immediate 28
  - syntax 3
- comment delimiter 5, 29
- communication interface
  - addressing units in a chain 49
  - baud rate 58
  - BOT characters 59
  - COM port selection (PORT) 258
  - controlling multiple COM ports 258
    - send response to alternate port 43
    - send response to both ports 42
  - echo enable 118
  - enable communication (E) 118
  - end-of-transmission characters 121
  - error prompt definition 126
  - Ethernet

- enable communication 227
  - Ethernet address 330
  - IP address 227
  - network mask 228
- good prompt definition 129
- lock COM port 215
- program definition prompt 126
- RP240 check 114
- XON/XOFF, enable & disable 371
- communications server 186, 187
- compiled motion See PLC program
- compiling (PCOMP) 249
  - execute from PLC program (PEXE) 252
  - failed PCOMP 295, 348
  - final velocity 365
  - GOBUF segments 157
  - looping 254, 255
  - memory status 295, 348
  - outputs (POUTA) 258
  - run the profile (PRUN) 260
  - status of program storage 312
  - status, free segments 284, 346
  - uncompile the profile (PUCOMP) 264
- compiling a profile or program (PCOMP) 249
- conditional branching
  - ELSE 119
  - IF 172
  - NIF 225
  - NWHILE 228
  - REPEAT 271
  - UNTIL 356
  - WHILE 368
- conditional go 160
- configuration
  - using a setup program 296
  - wizard in Motion Planner 8
- configuration error, motor 97
  - status 310
- configuration wizard
  - Motion Planner 8
- continue (!C) 59, 61, 67, 205
- continuous command execution mode (COMEXC) 63
- continuous current derating, motor 91
- continuous current, motor 91
- continuous positioning mode (MC1) 220
- continuous shift See shift, continuous
- control characters 369
- controlling multiple serial ports
  - select target port (PORT) 258
  - send response to alternate port 43
  - send response to both ports 42
- cosine 68, 264
- current damping ratio 287
- current foldback, enable/disable 81
- current loop bandwidth 80
- current loop gain 82
- current offset, phase A 108
- current offset, phase B 108
- current reduction at rest 77

---

## D

- damping

- servo
  - current 287
  - load 205
  - motor 90
  - position 287
  - velocity 289
- stepper
  - ABS 70
  - active damping 71
  - during acceleration 78
  - electronic viscosity 80
- data
  - assignment (DAT) 71
  - fields, in command syntax 4
  - program (DATP) 71, 72
  - program size (DATSIZ) 74
  - read from the RP240 111
  - statement (DATA) 72
    - memory required 222
  - storage 72, 73, 74, 313
  - teach 76
  - transfer 242, 243
- deadband
  - joystick center 198
  - joystick end 199
  - stall 135
- debounce time for programmable inputs 175
- debounce time for trigger interrupt inputs 343
- debugging tools See Programmer's Guide
  - axis status (extended) report 305
  - axis status report 302
  - break point 59
  - break, manual 60
  - configuration status report 310
  - ENABLE status 325
  - error log 317
  - error messages 14
  - HALT 163
  - identify bad command 18, 309
  - single-step mode 30, 296
  - technical support 164
  - trace mode 339, 340
  - translation mode 340
- deceleration 47
  - assignment/comparison 48
  - change on-the-fly 63
  - limits
    - hard 208
    - soft 217
  - scaling 23
  - s-curve profiling 19, 49
    - hard limits 208
    - homing 168
    - jogging 191
    - joystick 197
    - soft limits 217
- default command settings 3
  - restore (RFS) 272
- define
  - program/subroutine (DEF) 78
  - user status 176
- degrees, unit of measure 264
- delay time (T command) 300
- delay, brake output (OUTBD) 238
- delete a program/subroutine (DEL) 79
- delimiter, comment 5, 29
- delimiters, command 5
- direction
  - "positive" vs "negative" 69

- change with D command 68
- disable drive 114
- disable drive on kill 204
- disable hall sensor error checking (DHALL) 81
- display
  - messages 32
  - RP240 See RP240
- distance 68
  - assignment 69
  - change on-the-fly 63, 69
  - fractional step truncation 24, 281, 282
  - master See master, distance
  - registration 268
    - lock-out 269
  - scaling factor (SCLD) 275
  - target zone 297
- division 38
- download/upload files
  - using Motion Planner 8
- DRESET 113
- drive
  - active damping 71
  - auto current standby 77
  - bus voltage, status of 313
  - configuration
    - disable drive on kill 204
    - resolution (DRES) 113
  - configuration procedure
    - using Motion Planner 8
  - disable 114
  - disable drive on kill 204
  - enable 114
  - fault
    - if DRIVE0 or enable input
      - open 139
    - status 56, 306, 308
  - operating modes (DMODE) 86
  - shutdown 114, 203, 204
  - stall detect
    - enable (DSTALL) 115
  - temperature fault 307
  - temperature, status report 313

---

## E

- echo, communication 118
- electrical pitch 84
- electronic viscosity 80
- ELSE 119, 172, 225
- enable input
  - error status 314
- enable input status 123, 183, 325
  - error checking 129
  - error program 131
  - error status 122
- enable or disable Following 143
  - status 151, 319
- enabling the drives 114
- encoder
  - capture/counter enable (steppers) 120
  - input resolution (ERES) 124
  - master
    - polarity 224
  - position
    - assignment/comparison 256, 337
    - capture 247, 334
    - status 337
  - step & direction input 224
  - output resolution (ORES) 124
- position
  - assignment/comparison 138, 250
  - capture 180, 246, 333
  - error 251
  - status 318
- position report 335
- resolution (ERES) 124
  - selected with SFB 284
- Z-channel homing 164, 171
- end of line terminating characters (EOL) 120
- end of loop (LN) 205, 215
- end of loop, compiled (PLN) 254
- end of transmission characters (EOT) 121
- end program/subroutine/profile definition (END) 120
- end-of-travel limits
  - active level 183, 214
  - deceleration 208
    - s-curve 208
  - effect on command buffer 65
  - effect on homing 164, 169
  - enable/disable 207
  - error status 314
  - function assignment (INFNC) 181
  - function assignment (LIMFNC) 211
  - simulate activation 210
  - soft limit
    - decel 216
    - deceleration (LSAD) 217
    - deceleration, s-curve 217
    - range, negative direction 218
    - range, positive direction 218
  - status 209, 303, 327
- enter interactive data (') 31
- epitch 84
- erase all programs (ERASE) 124
- error
  - clearing 131
  - error checking enable (ERROR) 129
  - error detection level (ERRLVL) 127
  - error log
    - clear (CERRLG) 62
    - display contents (TERRLG) 317
  - error log (ERRORL) 130
  - motor configuration 310
  - program assignment (ERRORP) 131
  - prompt (ERRBAD) 126
  - responses 14
  - status 122, 314, 316
- Ethernet
  - enable communication (INFEN) 227
  - Ethernet address 330
  - IP address 227
  - network mask 228
- events, alarm 186
- exclusive or (^) 40
- execute program from PLC program 137
- expansion I/O, status 325
- extended axis status 56, 306, 308

---

## F

- factory default settings 113, 271
- factory default settings, restore 272
- fast status register 187
- faults See drive, fault. See error
  - axis (extended) status 307
  - axis status 304
  - cause drive shut down 114, 304, 307
  - configuration 310
  - fault on drive disable 139
  - faults activate fault output and relay output 114, 304, 307
  - motor configuration error 97, 310
- feedback source selection 284
- field separator 5
- filter, master position 138
- final velocity
  - homing 171
- final velocity (compiled motion) 365
- firmware update
  - using Motion Planner 10
- foldback mode 77
- follower
  - acceleration, max. 140
  - commanded position
    - assignment/comparison 263
    - transfer 339
  - conditional go 160
  - definition of 146
  - master assignment 146
  - phase shift See shift ratio to master 149
    - change on the fly 149
    - final (compiled motion) 150
    - status 151, 318
  - velocity, max. 140
- Following
  - application examples 143
  - conditions for killing a profile 145
  - distance scaling 24
  - enable or disable 143
    - status 151, 319
  - geared advance 139
  - status 151, 318, 320
  - step & direction input (MESND) 224
- force
  - actual, status 352
  - commanded, status 352
  - limit 94
    - status 78, 307
  - scaling 99
- force an alarm event (INTSW) 187
- forcer
  - inertia/mass ratio 214
  - mass 93
- fractional step truncation 281, 282

---

## G

- gains
  - acceleration feedforward 285
  - current damping ratio (SGIRAT) 287
  - current loop bandwidth (DIBW) 80
  - gain set

- displaying 321
- gain set, display 346
- gain set, enabling 285
- gain set, saving 288
- integrator enable (SGINTE) 286
- load damping (LDAMP) 205
- load-to-force mass ratio (LJRAT) 214
- load-to-rotor inertia ratio (LJRAT) 214
- notch filter A depth (DNOTAD) 102
- notch filter A frequency (DNOTAF) 103
- notch filter A quality factor (DNOTAQ) 104
- notch filter B depth (DNOTBD) 104
- notch filter B frequency (DNOTBF) 104
- notch filter B quality factor (DNOTBQ) 105
- notch lag filter break freq (DNOTLG) 106
- notch lead filter break freq (DNOTLD) 105
- position damping ratio (SGPRAT) 287
- position loop bandwidth (DPBW) 106
- torque/force limit (DMTLIM) 94
- velocity damping ratio (SGVRAT) 289
- velocity feedforward 288
- velocity limit (DMVLIM) 101
- velocity/position bandwidth ratio (SGPSIG) 287
- geared advance 139
- global command identifier (@) 5, 29
- GO 156
  - compiled (GOBUF) 157
- GOBUF, compiled motion segments 157
- good prompt (ERROK) 129
- gosub
  - branch to error program 129
  - on input condition 230
  - on user status condition 231
  - on VAR1 condition 232
  - on VAR2 condition 232
- GOSUB 158
- goto 159
  - branch to error program 129
- GOWHEN 160
  - error condition 123
    - check 130, 131
    - via trigger input 342
- greater than (>) 34
- greater than or equal (>=) 34

---

## H

- hall sensor
  - check sensor values (THALL) 321
  - inversion (SHALL) 289
- halt
  - stop error program 131
- HALT 163
- hard limit
  - active level (LIMLVL) 214

- deceleration (LHAD) 208
  - enable (LH) 207
  - s-curve deceleration 208
  - status 209, 327
- HELP 164
- hexadecimal value identifier (h) 5
- homing
  - acceleration 166
  - backup enable 168
  - deceleration 167
  - final direction 169
  - home input
    - active level 214
    - function assignment (INFNC) 181
    - function assignment (LIMFNC) 213
    - reference edge 169
    - status 209, 327
  - initiate (HOM) 164
  - s-curve accel/decel 167, 168
  - to encoder Z-channel 171
  - velocity
    - final 171
    - starting 170
  - zero absolute position 164

---

## I

- IF 50, 172, 225
  - ELSE 119
- immediate commands 28
- immediate data read from RP240 111
- immediate stop 67, 274
- in position 189
- inclusive or (!) 39
- incremental positioning mode (MAØ) 219
  - effect on distance 68
- indirect variables 359
- inductance, motor 92
  - maximum 95
  - minimum 95
- initial master cycle position 142
- inputs 11
  - analog See ANI
  - joystick See joystick
  - joystick (INFNC functions) 180
  - joystick (LIMFNC functions) 212
  - limits
    - end-of-travel See end-of-travel
    - home See homing, home input
  - programmable
    - active level (INLVL) 183
    - alarm event 180, 212
    - bit pattern 11
    - configuration 325
    - debounce time 175
    - enable (INEN) 177
    - function assignments (INFNC) 178
    - function assignments (LIMFNC) 211
    - simulate activation of 210
- jog
  - negative direction 180, 212
  - positive direction 180, 212
  - speed select 180, 212
- kill 180, 212
  - pause/continue 180, 212

- effect on command buffer 65
  - position capture 180
  - program select 181, 213
  - program select, BCD 178, 211
  - registration 265
  - status 174, 324
  - stop 67, 180, 212, 274
  - strobe time 185
  - thumbwheel 186, 355
  - trigger interrupt 180, 245, 246, 247, 248, 332, 333, 334
  - lockout time 343
  - user fault 180, 212
- trigger See trigger inputs
- virtual input override 173
- installation
  - Motion Planner 7
- integer variable (VARI) 361
  - display on RP240 117
  - writing (WRVARI) 370
- integrator, enable 286
- interactive data (') 31
- interrupt, program 229, 230, 232
- IP address 227

---

## J

- jerk calculations* 21
- jerk, reducing* 19
- jog*
  - acceleration* 190
    - s-curve* 190
  - deceleration* 191
    - s-curve* 191
  - input*
    - negative direction* 180, 212
    - positive direction* 180, 212
    - speed select* 180, 212
    - mode enable (JOG)* 189
    - using RP240* 83
    - velocity*
      - high* 192
      - low* 192
- joystick*
  - acceleration (JOYA)* 195
  - analog channel high (JOYAXH)* 197
  - center (JOYCTR)* 198
  - center deadband (JOYCDB)* 198
  - deceleration (JOYAD)* 196
  - enable operation (JOY)* 193
  - end deadband (JOYEDB)* 199
  - input functions (INFNC)* 180
  - input functions (LIMFNC)* 212
  - s-curve accel/decel* 196, 197
  - velocity high (JOYVH)* 199
  - velocity low (JOYVL)* 200
  - voltage range* 199
  - zero (JOYZ)* 201
- jump* 160
- JUMP* 202

---

## K

- kill
  - conditions that will kill a
    - Following move 145
    - disable drive 204

immediate (!K) 185, 220  
input 180, 212  
on stall (ESK) 135  
Kill 203

---

## L

label  
  declaration (\$) 29  
  transfer 326  
lead/lag filters  
  lag 106  
  lead 105  
LEDs, RP240 84  
left-to-right math 5  
length, master cycle 141  
less than (<) 35  
less than or equal (<=) 35  
limits  
  activate output 239  
  end-of-travel See end-of-travel.  
  See end-of-travel limits  
  function assignments (LIMFNC)  
  211  
  simulate activation of 210  
  home See homing, home input  
  status 209, 327  
line feed  
  command delimiter 5  
  transmission character 121  
linear interpolation  
  distance scaling 24, 280  
linear motor pitch 84  
load, damping 205  
load-to-forcer mass 214  
load-to-rotor inertia 214  
lock COM port to a task 215  
lock-out distance, registration 269  
logical operators  
  AND 50  
  NOT 226  
  OR 233  
loops  
  compiled 255  
  end of loop 215  
  compiled 254  
  nested 205  
  terminate 219

---

## M

mass, forcer 93  
master  
  assign an ANI input (ANIMAS)  
  52  
  definition of 146  
  status 151, 319  
  direction, status of 151, 319  
  distance  
  fractional step truncation 282  
  programming (FOLMD) 148  
  encoder  
  position 256, 337  
  captured 247, 334  
  clear 257  
  set 257  
  follower assignment 146  
  master cycle  
  counting

  restart 141  
  length 141  
  master cycle  
  number  
  assignment/comparison 226  
  transfer 328  
  position  
  assignment/comparison  
  (PMAS) 256  
  capture (PCMS) 248  
  capture (TPCMS) 334  
  initial 142  
  rollover 256, 336  
  transfer (TPMAS) 336  
  status 151, 319  
  master cycle position capture 248,  
  334  
  master position filtering 138  
  status 151, 319  
  master position prediction 150  
  status 151, 319  
  moving, status of 151, 319  
  ratio to follower 148  
  change on the fly 149  
  status 151, 318  
  scaling 24, 281  
  velocity 355, 366  
mathematical operators  
  () 36  
  \* 37  
  / 38  
  + 37  
  = 33  
  SQRT 294  
maximum allowable position error  
  293  
  axis status 303  
  error status 314  
maximum allowable velocity error  
  axis status 307  
  defining 293  
maximum follower acceleration 140  
maximum follower velocity 140  
memory  
  after a drive reset (DRESET) 113  
  after a reset 271  
  allocation 222  
  data statement (teach mode) 222  
  labels 29  
  locking 181, 213  
  return to factory settings 272  
  status, usage 327  
messages  
  display on RP240 (DWRITE) 117  
  error 14  
  sending 32  
motion parameters 156  
Motion Planner 7  
  drive configuration 8  
  drive operating system update 10  
  installation 7  
  quick tour 8  
motion, compiled See compiled  
motion  
motor  
  ambient temperature 90  
  auto-configure 8, 97  
  configuration error 97  
  continuous current 91  
  continuous current derating 91  
  current standby mode 77  
  damping 90  
  inductance 92

  maximum 95  
  minimum 95  
  linear motor pitch 84  
  peak current 92  
  pole pairs 108  
  rated speed 101  
  rotor inertia 93  
  selection recorded with DMTR 97  
  static torque 99  
  temperature fault 307  
  temperature report 328  
  thermal time constant 100  
  voltage constant (Ke) 93  
  winding resistance 98  
  winding temperature, max. 96  
  winding thermal resistance 98  
  winding time constant 100  
motor data table, updates 97  
move completion criteria 297  
moving/not moving status 224, 303  
multi-line response 121  
multiplication 37  
multi-tasking  
  assign axis to tasks (TSKAX) 346  
  control task swapping 347  
  identify the controlling task 306,  
  351  
  lock resource to a task 215  
  status 350  
  task identifier (%) 5, 27  
  tasks active (status) 299, 351

---

## N

nested loops 205  
network mask 228  
neutral characters 5  
NIF 172, 225  
  ELSE 119  
not equal (<>) 36  
not, bitwise operator (~) 41  
not, logical operator (NOT) 226  
notch filter A  
  depth 102  
  frequency 103  
  quality factor 104  
notch filter B  
  depth 104  
  frequency 104  
  quality factor 105  
notch lag filter break frequency 106  
notch lead filter break frequency 105  
number, master cycle 226, 328  
numeric variable 370  
  clearing 360  
  display on RP240 116  
NWHILE 228, 368

---

## O

offset  
  position 251, 261  
  master encoder 257  
  resolver offset angle 294  
  status 347  
on conditions (program interrupts)  
  229, 230, 231, 232  
one-shot registration 269  
on-the-fly D changes 63, 69

- on-the-fly FOLRD & FOLRN changes 63, 149
- on-the-fly MA & MC changes 220
- on-the-fly motion MA & MC changes 63
- on-the-fly profile change not possible 303
- on-the-fly V, A & AD changes 63, 220
- operating hours, report 311
- operating modes, drive (DMODE) 86
- operating system update using Motion Planner 10
- operation priority level 36
- operator symbols
  - ' 31
  - ! 28
  - " 32
  - # 30
  - \$ 29
  - & 38
  - () 36
  - \* 37
  - . (bit select operator) 31
  - / 38
  - ; 29
  - @ 29
  - \\ 32
  - ^ 40
  - | 39
  - ~() 41
  - + 37
  - < 35
  - << 41
  - <= 35
  - <> 36
  - = 33
  - > 34
  - >= 34
  - >> 42
- or 368
- or, Boolean exclusive (^) 40
- or, Boolean inclusive operator (|) 39
- or, logical operators (OR) 233
- other input status (INO) 183
- outputs 11
  - analog monitor A
    - scaling 86
    - variable 87
  - analog monitor B
    - scaling 88
    - variable 89
  - brake output delay (OUTBD) 238
  - compiled (POUTA) 258
  - encoder 233
  - fault output and relay output
    - activate on fault condition 304, 307
  - programmable
    - activate 234
    - activate, multiple 236
    - active level (OUTLVL) 240
    - bit pattern 11
    - configuration 325
    - enable (OUTEN) 238
    - fault output 240
    - function assignments (OUTFNC) 239
    - limit encountered 239
    - maximum position error exceeded 240
    - moving/not moving 239

- output on position 240, 241
  - PLC 242
  - program in progress 239
  - stall indicator 239
  - status 234, 235, 236, 330
  - strobing 242
  - step and direction 233
- override analog inputs (ANIEN) 51
- override mode 77, 102
- over-temp fault
  - drive 307
  - motor 307

---

## P

- partitioning memory 222
- password, RP240 106
- pause active, status 295, 348
- pause program execution (PS) 260
- pause/continue input 180, 212
  - effect on motion & program execution 65
- peak current, motor 92
- phase A current offset 108
- phase B current offset 108
- phase balance 107
- phase, shift
  - continuous 153
  - position assignment/comparison 263
  - position transfer 338
  - preset 154
- phase, tracking 139
- Pi ( $\pi$ ) 253
- PLC
  - data inputs (INPLC) 184
  - inputs 184
  - strob outputs (OUTPLC) 242
- PLC program
  - compiling (PCOMP) 249
  - defining (PLCP) 253
  - execute compiled program from (PEXE) 252
  - execute program from (EXE) 137
  - initiate scan mode (SCANP) 276
  - memory storage 222
  - time of last scan 345
- pointer, data
  - location 109, 313
  - reset 74
  - set 73
- polarity
  - end-of-travel inputs 214
  - home inputs 214
  - master encoder input 224
  - programmable inputs 183
  - programmable outputs 240
  - trigger inputs 183
- pole pairs, motor 108
- PORT (selecting a COM port) 258
- position
  - absolute, establishing 251, 261
  - actual 251, 335, 336
  - capture
    - commanded 120, 245, 332, 343, 352
    - commanded position 180
    - encoder 120, 180, 246, 333, 343, 352
    - master encoder 247, 334
    - for registration 265
  - master 180
    - master cycle 248, 334
  - commanded 244, 251, 331, 336
  - captured 245
  - current feedback device 138
  - encoder 250, 335
    - assignment/comparison 138
    - master 256, 337
    - status 318
  - error
    - exceeded max. limit 123, 303, 314
    - setting max. allowable (SMPER) 293
    - status 251, 336
  - master cycle 142
  - master encoder
    - absolute, clearing 257
    - absolute, establishing 257
    - offset 257
  - master position prediction See master, master position prediction
  - offset 251, 261
  - output on position 241
  - position bandwidth ratio 287
  - position damping ratio 287
  - position loop bandwidth 106
  - positioning mode selection 219, 220
    - change on the fly 220
  - resolver 335
  - RP240 cursor (DPCUR) 107
  - set to zero after homing 164
  - setpoint 244, 331
  - shift
    - continuous 153
    - geared to master 139
    - preset 154
    - set to zero upon FOLEN1 143
  - tracking 289
- power dissipation circuit
  - active, status of 307
  - latched bit 78
  - fault, status of 57, 307
- power-up start program (STARTP) 296
- pre-emptive GOs See on-the-fly. See on-the-fly
- preset positioning mode (MCØ) 220
- preset shift See shift, preset
- priority level 36
- product revision 3, 341
- profile, compiling (PCOMP) 249
- program
  - branch condition 29, 158, 159, 202, 231
  - break point 59
  - comments 5
  - contents, display 337
  - data (DATP) 72
  - debug 340
    - command errors 309
  - definition 78, 120, 272
  - definition, prompt (ERRDEF) 126
  - directory 312
  - display contents 337
  - drive reset, effect of 113
  - erase 124
  - error handling 131
  - error responses 14
  - execution

- from compiled program (EXE)
  - 137
  - status 295, 317, 348
  - termination 60, 163
  - upon power-up 296
- flow control 163, 202, 225, 226, 271
  - WAIT statements 366
- interrupts 229
- jump (branch) 202
- label 29
- list all programs 312
- memory allocation 222
- name 79, 185
- pause (PS) 260
- PLC
  - definition of (PLCP) 253
  - memory storage 222
  - PLC scan (SCANP) 276
  - power-up program 296
  - reset, effect of 271
  - run 272
  - security 181, 213
  - selection 181, 185, 213
  - size restriction 222
  - step through 30, 296
  - storage 222, 327
  - trace mode 339, 340
  - translation mode 340
  - upload 337
- program/profile
  - definition 120
- programmable inputs See inputs, programmable
- programmable outputs See outputs, programmable
- programming
  - programming tools 7
- programming examples See application examples
- prompt
  - error 126
  - good (no errors) 129
  - program definition 126
- PWM frequency 109

---

## R

- radian 264
- rated speed, motor 101
- ratio of follower to master 148, 149
  - change on the fly 149
  - final ratio (in compiled profile) 150
  - status 151, 318
- read a value (READ) 267
- read data from parallel I/O 242
- read RP240 data (DREAD) 110, 111
- read RP240 function key (DREADF) 110, 111
- registration
  - distance 268
  - lock-out 269
  - enable 265
  - input debounce 343
  - single-shot (REGSS) 269
  - status, profile not possible 303
  - status, trigger occurred 303
  - trigger interrupt 180
- relational operators 172, 271, 356, 366, 368

- relay output
  - activate on fault condition 304, 307
  - status 330
- repeat 226, 271, 356
- REPEAT 50, 271
- reset
  - data pointer (DATRST) 74
  - drive (DRESET) 113
  - drive (RESET) 271
- RESET 271
- resolution
  - drive 113
  - encoder 124
  - encoder input 124
  - encoder output 233
  - resolver 124
  - step & direction output 233
- resolver
  - offset angle 294
  - status 347
  - position report 335
  - resolution 124
  - transfer absolute position (TPRA) 337
- responses
  - beginning-of-transmission characters 59
  - end of line characters 120
  - end-of-transmission characters 121
  - error 14
  - send to both COM ports 42, 43
  - restart master cycle counting 141, 342
- return to factory settings 272
- revision level 341
- rollover of master cycle position 256, 336
- rotor inertia, motor 93
- round-off error, square root 294
- RP240
  - COM port setup 114
  - connection verified 295
  - connection verified 348
  - data read 110
  - data read immediate mode 111
  - display binary variable (DVARB) 116
  - display integer variable (DVARI) 117
  - display layout 107, 118
  - display numeric variable (DVAR) 116
  - display variable 116
  - jog mode 83
  - LEDs 84
  - password 106
  - position cursor 107, 110
  - read function key 110
  - stop key enable/disable 116
  - value of key pressed 83
  - write text 117
- RS-232C
  - auto-addressing (ADDR) 49
  - COM port setup 114
  - enable/disable communication 118
  - lock COM port 215
- RS-485
  - auto-addressing (ADDR) 49
  - COM port setup 114
  - disable XON/XOFF 371

- enable/disable communication 118
- lock COM port 215
- run, compiled program (PRUN) 260
- run, PLC program (SCANP) 276
- run, program (RUN) 272

---

## S

- save command buffer on limit 65
- scaling 22
  - acceleration 23, 279
  - analog monitor output A 86
  - analog monitor output B 88
  - distance 24, 280
  - enabling 275
  - examples 24
  - force command 99
  - master 24, 281
  - torque command 99
  - velocity 23, 282
- scan PLC program (SCANP) 276
- scan time of last PLC program 345
- s-curves 19
  - acceleration 46
  - deceleration 49
  - hard limit deceleration 208
  - homing acceleration 167
  - homing deceleration 168
  - jogging deceleration 191
  - joystick acceleration 196
  - joystick deceleration 197
  - soft limit deceleration 217
- S-curves
  - jogging acceleration 190
- security of programs 181, 213
- segment See compiled motion
- select bit 31
- servo
  - commanded position 244, 331
  - data gathering, status 295, 348
  - feedback source selection 284, 318
  - gain sets
    - display 346
    - enable 285
    - saving 288
  - gains See gains
  - move completion criteria 297
  - position error
    - max. allowable 293
  - position tracking 289
  - target distance zone 297
  - target velocity zone 299
  - target zone mode enable 297
  - target zone settling time 350
  - target zone settling timeout period 298
- servo tuning
  - current damping ratio (SGIRAT) 287
  - current loop bandwidth (DIBW) 80
  - integrator enable (SGINTE) 286
  - load damping (LDAMP) 205
  - load-to-rotor inertia ratio (LJRAT) 214
  - notch filter A depth (DNOTAD) 102
  - notch filter A frequency (DNOTAF) 103

notch filter A quality factor (DNOTAQ) 104  
 notch filter B depth (DNOTBD) 104  
 notch filter B frequency (DNOTBF) 104  
 notch filter B quality factor (DNOTBQ) 105  
 notch lag filter break freq (DNOTLG) 106  
 notch lead filter break freq (DNOTLD) 105  
 position damping ratio (SGPRAT) 287  
 position loop bandwidth (DPBW) 106  
 presently active gains, display 321  
 torque/force limit (DMTLIM) 94  
 velocity damping ratio (SGVRAT) 289  
 velocity limit (DMVLIM) 101  
 velocity/position bandwidth ratio (SGPSIG) 287  
 set data pointer (DATPTR) 73  
 settling time See target zone  
 setup wizards  
   Motion Planner 8  
 shift  
   continuous 153  
     application example 153  
     position  
       assignment/comparison 263  
       transfer 338  
   geared to master 139  
   L to R (bit 1 to bit 32) 42  
   preset 154  
     application example 155  
     position  
       assignment/comparison 263  
       transfer 338  
   R to L (bit 32 to bit 1) 41  
   status 151, 319  
 shutdown the drive 114, 203, 204  
 sine 291  
 sine wave (virtual master)  
   amplitude (SINAMP) 291  
   angle (SINANG) 292  
   start (SINGO) 292  
 single step mode 30, 296  
 single-shot registration (REGSS) 269  
 soft limit  
   deceleration (LSAD) 217  
   effect on command buffer 65  
   enable (LS) 216  
   negative-direction range (LSNEG) 218  
   positive-direction range (LSPOS) 218  
   s-curve deceleration 217  
 software revision level 341  
 space (neutral character) 5  
 square root 294  
 stall detection  
   drive (DSTALL)  
     enable 115  
 stall detection  
   drive (DSTALL)  
     sensitivity 115  
   encoder (ESTALL) 136  
     backlash deadband (ESDB) 135  
 start timer (TIMST) 323  
 start-up program (STARTP) 296  
 static torque 99  
 statistics, drive config. & status 350  
 status  
   axis 55, 302  
   axis, extended 56, 305, 306, 308  
   clear latched bits (DCLRLR) 77  
   command error 18, 309  
   commanded position 244, 331  
     captured 245, 332  
   compiled motion memory 295, 348  
   data pointer location 109, 313  
   enable input 325  
   encoder  
     master position 256, 337  
   encoder position 250, 335  
     captured 246, 333  
   error 122, 314, 316  
   error log 317  
   Ethernet address 330  
   expansion I/O 325  
   Following 151, 318, 320  
   free segments, compiled memory 284, 346  
   gain set 346  
   gains, presently active 321  
   inputs 356  
     enable 325  
     programmable 174  
   interrupt 356  
   labels 326  
   limits 209, 214, 327  
   master encoder position  
     captured 247, 334  
   memory 312, 327  
   motor configuration errors 310  
   moving/not moving 224  
   outputs 234, 235, 236, 330  
   pause 295, 348  
   position capture 343, 352  
   position error 251, 336  
   program contents 337  
   program directory 312  
   program execution 295, 317, 348  
   resolver position 335  
   settling time 350  
   software revision level 341  
   system 294, 348, 349, 356  
   timer 351  
   user 175, 353, 357  
   velocity  
     feedback device 354  
     motor 353  
     voltage input for ANI 301  
     wait 295, 348  
 step & direction from master encoder (MESND) 224  
 step through a program 30, 296  
 stop  
   command 272  
   effect on program execution 67  
   input (INFNCi-D) 67, 180, 274  
   input (LIMFNCi-D) 212  
 stop key (RP240) 116  
 stop timer (TIMSTP) 323  
 string variable (VARS) 32, 362, 371  
   clearing 360  
 strobe  
   PLC 242  
   thumbwheels 243  
   time 185  
 subroutine  
   branch condition 231  
   definition 78, 120  
   effect of drive reset 113  
   effect of reset 271  
   name 79  
   substitutions, command values 6  
     binary variable (VARB) 360  
     data assignment (DAT) 71  
     numeric data read (READ) 268  
     numeric variable (VAR) 359  
     RP240 Function Key (DREADF) 111  
     RP240 numeric data (DREAD) 110  
     thumbwheel data (TW) 355  
   subtraction 37  
   support, technical assistance 164  
   swapping, tasks  
     controlling 347  
     status 299, 351  
   syntax 3, 4  
     guidelines 5  
   system status (SS) 294

---

## T

tangent (TAN) 301  
 target zone 303  
   display actual settling time 350  
   enabling 297  
   setting the distance zone 297  
   setting the timeout period 298  
   setting the velocity zone 299  
   timeout error 303  
 task supervisor 27  
 tasks See multi-tasking  
   direct with prefix (%) 27  
 teach mode  
   data assignment (DAT) 71  
   data pointer (DATPTR) 73  
   data pointer reset (DATRST) 74  
   data program (DATP) 72  
   data program size (DATSIZ) 74  
   data statement (DATA) 72  
   data storage (DATTC) 76  
   memory requirement 222  
 technical support 164  
 temperature fault  
   drive 307  
   motor 307  
 temperature status  
   drive 313  
   motor 328  
 temperature, motor (ambient) 90  
 temperature, motor winding 96  
 terminal emulator  
   Motion Planner 9  
 terminate loop (LX) 219  
 terminate program execution 60, 131, 163  
 thermal time constant, motor 100  
 thumbwheel  
   assignment (TW) 355  
   data inputs (INSTW) 186  
   strobe outputs (OUTTW) 243  
 time delay (T) 300  
 timeout, target zone 303  
 timer 351  
   assignment of value (TIM) 322  
   start 323  
   stop 323



- value to cause alarm (TIMINT) 322
- torque
  - actual, status 352
  - commanded, status 352
  - limit 94
    - status 78, 307
  - scaling 99
  - static 99
- trace mode 339, 340
- transfer
  - actual torque/force (TTRQA) 352
  - actual velocity (TVELA) 354
  - analog input voltage, ANI (TANI) 301
  - axis status (TAS) 302
  - axis status, extended (TASX) 305, 306, 308
  - bus voltage (TDVBUS) 313
  - captured commanded position (TPCC) 332
  - captured master cycle position (TPCMS) 334
  - command error 309
  - commanded acceleration (TACC) 300
  - commanded position of follower (TPSLV) 339
  - commanded torque/force (TTRQ) 352
  - commanded velocity (TVEL) 353
  - configuration status (TCS) 310
  - continuous current rating (TDICNT) 311
  - data pointer location (DTPTR) 313
  - drive temperature (TDTEMP) 313
  - error log (TERRLG) 317
  - error status (TER) 314, 316
  - Ethernet address (TNTMAC) 330
  - expansion I/O configuration (TIO) 325
  - Following status (TFS) 318, 320
  - free segments (TSEG) 346
  - gains (TGAIN) 321
  - hall sensor values (THALL) 321
  - input status programmable (TIN) 324
  - labels (TLABEL) 326
  - limits (TLIM) 327
  - master cycle number (TNMCY) 328
  - master cycle position (TPMAS) 336
  - master velocity (TVMAS) 355
  - maximum current rating (TDIMAX) 312
  - memory usage (TMEM) 327
  - motor temperature (TMTEMP) 328
  - net position shift (TPSHF) 338
  - operating hours (TDHRS) 311
  - other input status (TINO) 325
  - output status (TOUT) 330
  - PLC scan time 345
  - position commanded (TPC) 331
  - position error (TPER) 336
  - position of captured encoder (TPCE) 333
  - position of captured master encoder (TPCME) 334
  - position of encoder/resolver (TPE) 335
  - position of master encoder (TPME) 337
  - position of selected feedback device (TFB) 318
  - program (TPROG) 337
  - program directory (TDIR) 312
  - program execution status (TEX) 317
  - resolver (TPRA) 337
  - resolver offset angle (TSROFF) 347
  - revision level (TREV) 341
  - servo gain set (TSGSET) 346
  - servo settling time (TSTLT) 350
  - statistics (TSTAT) 350
  - system status (TSS) 348, 349
  - timer (TTIM) 351
  - user status (TUS) 353
  - velocity error (TVE) 353
  - translation mode 340
  - transmitting message strings 369
  - trigger inputs
    - active level 183
    - debounce 175
      - if interrupt inputs 343
    - I/O bit pattern 11
    - position capture 180, 245, 246, 247, 248, 332, 334
      - status 343, 352
    - programmed
      - GOWHEN function 342
      - restart master cycle counting 342
        - status 151, 319
    - programmed functions 178, 211
    - registration 180, 265
      - status 174, 324
  - trigger interrupt input
    - registration 268
  - trigonometric operators 253, 264, 291, 359
  - troubleshooting See Programmer's Guide
    - axis status 302
    - axis status, extended 305
    - configuration status report 310
    - drive statistics 350
    - enable input status 325
    - error log 317
    - error messages 14
    - Following status 318, 320
    - Hall sensors 321
    - identify bad command 18
    - system status 294
  - truncation
    - acceleration/deceleration 23, 279
    - distance 24, 281, 282
    - velocity 23, 283

## V

- uncompile a compiled profile (PUCOMP) 264
- unconditional looping 205, 215
  - compiled 254, 255
- units of measurement 3, 22, 276
  - linear motors 84
- until 226, 271, 356
- UNTIL 50
- updates
  - drive operating system

- using Motion Planner 10
- motor data table 97
- upload/download files
  - using Motion Planner 8
- user fault 180, 212, 240
- user fault input
  - active level 183
  - error status 314
- user status 175, 353, 356
  - basis for gosub 231
  - definition (INDUST) 176

## V

- value substitution, command fields 6
- variable
  - analog monitor output A 87
  - analog monitor output B 89
- variables 31
  - binary 357
    - display on RP240 116
    - writing 370
  - clearing (VARCLR) 360
  - conversion between numeric and
    - binary 363
  - indirect 359
  - integer 361
    - display on RP240 117
    - writing 370
  - numeric 359
    - display on RP240 116
    - teach data 71, 76
    - writing 370
  - string 362
    - writing 371
- velocity 357
  - actual (feedback device) 354
  - assignment 358
    - actual 365
      - commanded 364
    - bandwidth, ratio with position 287
    - change on-the-fly 63, 220
    - commanded 353
    - damping ratio 289
    - error
      - setting max. allowable (SMVER) 293
        - status 353
    - feedforward gain (SGVF) 288
    - final (compiled motion) 365
    - jog, high 192
    - jog, low 192
    - limit 101
    - master
      - assignment/comparison 366
      - transfer 355
    - maximum, follower axis (steppers) 140
    - scaling (SCLV) 275, 282
    - target zone 299
  - virtual input override 173
  - virtual master
    - count acceleration (FVMACC) 155
    - count frequency (FVMFRQ) 156
  - viscosity, electronic 80
  - voltage
    - ANI input 51, 301
    - voltage constant (Ke) of motor 93
    - voltage range for ANI inputs 52

---

## W

wait 226, 366  
WAIT 50  
    compared to GOWHEN 162  
    status 295, 348  
waveform 117  
while 226, 228, 368  
WHILE 50  
winding resistance, motor 98  
winding temperature, max. 96  
winding thermal resistance, motor 98

winding time constant, motor 100  
wizards, setup  
    Motion Planner 8  
write 32, 369  
    binary variable 370  
    integer variable 370  
    message 369  
    numeric variable 370  
    RP240 text 117  
    string variable 371

---

## X

XON/XOFF, enable & disable 371

---

## Z

z-channel 171  
zero absolute position after homing  
    164  
zero master cycle position 142