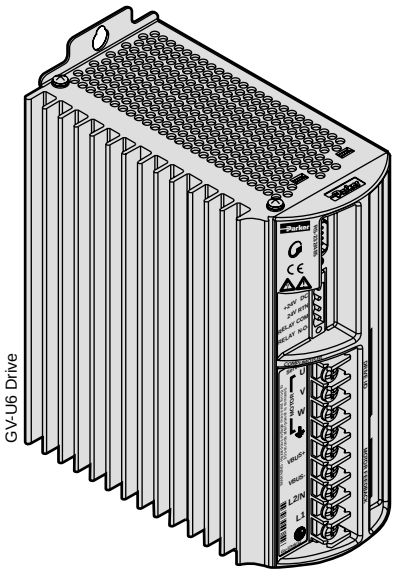
Automation

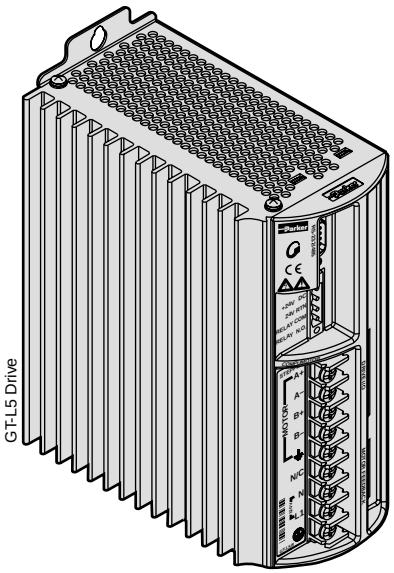
p/n 88-017778-01 E

Gemini Series Programmer's Reference

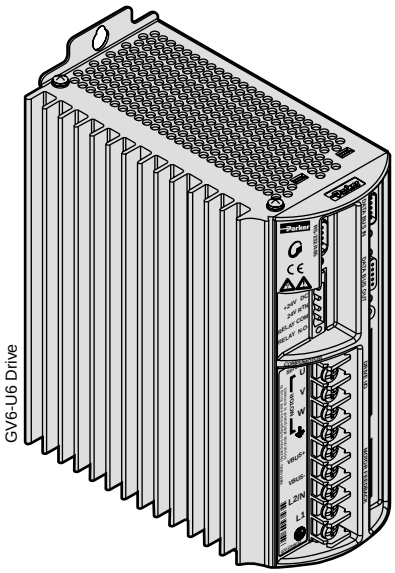
Effective: March 19, 2001



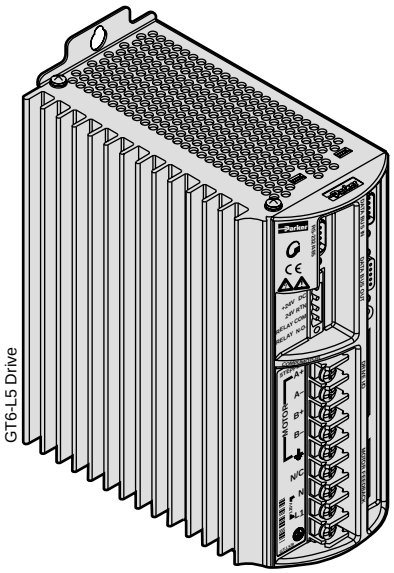
Gemini GV Series
Digital Servo Drives



Gemini GT Series
Digital Stepper Drives



Gemini GV6 Series
Digital Servo Controller/Drives



Gemini GT6 Series
Digital Stepper Controller/Drives

IMPORTANT

User Information

⚠ WARNING ⚠

Gemini 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.

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

Parker
Automation

Technical Support Email
tech_help@cmotor.com

Change Summary

Revision E

March 19, 2001

Revision E Changes: This document, 88-017778-01E, supersedes 88-017778-01D. Changes associated with operating system revisions and document clarifications and corrections are noted below.

Topic	Description
`SGINTE2` Added **(New in OS revision 1.70)**	For the `SGINTE` command, a new setting has been added. `SGINTE2` enables the integrator only at the end of the move. Refer to page 149.
PROFIBUS Option Added **(New in OS revision 1.70)**	A PROFIBUS option is now available (GV6-nnn-PB and GT6-nnn-PB). Several commands have been modified for the PROFIBUS option. • `ERROR` command: Bit 19 indicates FIELDBUS error. • `OUTFNC` command: identifier `I` added to signal a FIELDBUS error (`OUTFNCi-I`). • `TASX` command: Bit 27 indicates FIELDBUS error. • `TCS` command: two faults added, for `FBPIC` (-32158) and `FBPOC` (-32168). • `TER` command: Bit 19 indicates FIELDBUS error.
Document clarifications and corrections	• `DCMDZ` (page 68), command description rewritten. • `DNOTAD` (page 91), Lag/Lead Filter added to block diagram. • `GOSUB` (page 112), example rewritten. • `SMPER` (page 153), note added: Does not apply to GV operating in `DMODE2` or `DMODE4`. • `SMVER` (page 154), note added: Does not apply to GV operating in `DMODE2`. • `TANI` (page 158), description has been rewritten and an example has been added. • In `TASX` (page 161), Bit 5 indicates resolver failure (not encoder failure). • Various typographical errors have been corrected throughout this document.
New commands added to this document:	
ANICDB.....Analog Input Center Deadband ERASE.....Erase All Programs and Profiles INUFD.....User Fault Input Delay OUTBD.....Brake Output Delay TASF.....Transfer Axis Status (full-text report)	TASXF.....Transfer Extended Axis Status (full-text report) TERF.....Transfer Error Status (full-text report) TPRA.....Transfer Absolute Resolver Position TSSF.....Transfer System Status (full-text report) VARCLR.....Variable Clear

Revision D Changes: This document, 88-017778-01D, supersedes 88-017778-01C. Changes associated with operating system revisions and document clarifications and corrections are noted below.

Topic	Description
Integer variables capability **(New in OS revision 1.60)**	The GT6 and GV6 drives now allow you to define up to 99 user variables (integer variables). Integer variables are represented by the syntax VARI*n*, where "*n*" is the number of the variable (range is 1-99). Integer variables may be used for: • Variable assignments and math operations (e.g., VARI3=13, VARI9=PC/2, VARI4=VARI+2). Assignment options are: - Integer constant (range is –2,147,483,648 to +2,147,483,647) - A system variable: A (accel), AD (decel), D (distance), V (velocity), PC (commanded position), PE (encoder/resolver position), or PER (position error). - Math operations between constants, other VARI variables, and system variables. Available math operations are + (addition), - (subtraction), * (multiplication) and / (division). • Command value substitutions (e.g., L(VARI2)). This is applicable to the T (time delay), L (loop), A (accel), AD (decel), V (velocity), and D (distance) commands only. • Variable comparisons in IF and WAIT conditional expressions (e.g., IF(VARI2<VARI1), WAIT(PE>=VARI2)). NOTE: A, AD, V and T values are real numbers (resolution of A, AD, and V is 0.0001, resolution of T is 0.001). When substituting or comparing integer variables, the integer is applied to the full decimal range (for example, if the value of VARI5 is 136298, the substitution A(VARI5) yields an acceleration value of A13.6298). The converse is true when assigning the value of A, AD, or V to an integer variable (for example, if the value of A is 22.0000, the integer assignment VARI4=A yields a VARI4 value of 220000). Refer to page 24, and to the VARI, IF, and WAIT commands, for details on how to use variables.
Additional operands available for IF and WAIT conditional expressions **(New in OS revision 1.60)**	Additional options for comparison operands: VARI (integer variables), A (accel), AD (decel), D (distance), V (velocity), PC (commanded position), PE (encoder/resolver position), or PER (position error). Additional comparison operators (in addition to =, equals): <> (not equal), > (greater than), < (less than), >= (greater than or equal to), and <= (less than or equal to). Examples: IF(VARI5<PE), IF(PC>PE), WAIT(PE>=16000). For additional details, refer to the IF command (page 116) and the WAIT command (page 184).
Document clarifications and corrections	• In THALL (page 169), added clarification of troubleshooting Hall sensor problems. • Added a description for the ERRDEF command, for customizing the program definition prompt — see page 103.

Revision C Changes:

Topic	Description
Changes for the release of OS revisions 1.02 and 1.50, as well as the GT6 and the GV6 drives:	
New name for this manual. Additional programmer's guide.	The name of this manual has changed from *Gemini Series Command Reference* to *Gemini Series Programmer's Reference*. The name was changed because it now includes the *Gemini Major Programmer's Guide* section (pages 19-56), which provides guidelines for implementing firmware features in the Gemini Major products (GT6 and GV6 drives).
Velocity limit and scaling	(OS 1.02) The maximum allowable `DMVLIM` (velocity limit) and `DMVSCL` (velocity scaling) settings for the GT/GT6 drives has been raised from 50.000000 to 60.000000.
Stall detect sensitivity	(OS 1.02) The maximum allowable `DSTALL` setting for the GT/GT6 drives has been raised from 40 to 50.
Variable PWM frequency	(OS 1.02) The `DPWM` command also applies to the GV-U12 drive (same options as for GV-U6).
Drive resolution and step/direction output resolution	(OS 1.02) The range for the `DRES` and `ORES` commands (when used with the GT drives) has changed from 200-100000 to 200-128000 counts/rev.
XON/XOFF support	(OS 1.50) The `XONOFF` command has been added for all Gemini drives (GT, GV, GT6, and GV6) to support ASCII handshaking.
Resolver feedback support	(OS 1.50) The GV and GV6 now support resolver feedback. The resolver functions the same as an encoder. Configuration commands (listed below) have been added to the motor data table, in support of the setup wizard in Motion Planner and Pocket Motion Planner. • Feedback source is selected with `SFB`. • Resolution is set with `ERES` • Offset angle is set with `SRSET` (can be checked with `TSROFF`) `DMODE11` was added to support automatically setting the resolver offset angle with the `SRSET` command. The resolver is automatically detected on reported in the `TREV` report (e.g., `*TREV-GV6-L3R_D1.50_F1.02`).
Accommodations for GT6 & GV6	(OS 1.50) I/O handling commands (e.g., `INDEB`, `INLVL`) and Status Commands (e.g., `TIN`, `TOUT`) have been modified to support the I/O set for the GT6 and GV6 products. See below for a list of new commands added to support the GT6 & GV6. Where appropriate, references are made to information in the *Gemini Major Programmer's Guide*, a separate document.

New commands for GT6 & GV6 (OS 1.50)

A ..... Acceleration
AA..... Average Acceleration
AD ..... Deceleration
ADA..... Average Deceleration
COMEXC..... Continuous Command Processing Mode
COMEXL ..... Continue Execution on Limit
COMEXR..... Continue Motion on Pause/Continue Input
COMEXS..... Continue Execution on Stop
D..... Distance
DEF PROF..... Begin Profile Definition
DEF PROG ..... Begin Program Definition
DEL PROF ..... Delete a Profile
DEL PROG..... Delete a Program
ELSE ..... Else Condition of IF Statement
END..... End Program/Profile Definition
ERROR ..... Error-Checking Enable
ERRORP..... Error Program Assignment
GO..... Initiate Motion
GOBUF..... Store a Motion Segment in Compiled Memory
GOSUB ..... Call a Subroutine
GOWHEN ..... Conditional GOBUF
HOM..... Go Home
HOMA..... Home Acceleration
HOMBAC..... Home Backup Enable
HOMDF ..... Home Final Direction
HOMEDG ..... Home Reference Edge
HOMV..... Home Velocity
HOMVF ..... Home Final Velocity
HOMZ..... Home to Encoder Z-Channel
IF ..... IF Statement
INFNC ..... Input Function
INSELP..... Select a Program using Inputs
JUMP..... Jump to a Program (and do not return)
K..... Kill Motion
KDRIVE..... Disable Drive on Kill
L ..... Loop
LH..... Hardware End-of-Travel Limit – Enable
LHAD..... Hardware End-of-Travel Limit Deceleration
LHADA ..... Hardware End-of-Travel Limit S-Curve Decel
LN..... End of Loop
LS..... Software End-of-Travel Limit – Enable
LSAD..... Software End-of-Travel Limit Deceleration
LSADA ..... Software End-of-Travel Limit S-Curve Decel
LSNEG ..... Software EOT Limit Negative Travel Range
LSPOS ..... Software EOT Limit Positive Travel Range
MA.....Absolute/Incremental Positioning Mode Enable
MC.....Preset/Continuous Positioning Mode Enable
NIF.....End IF Statement
OUT.....Output State
OUTFNC .....Output Function
PLN .....End of Loop (Compiled Motion)
PLOOP.....Beginning of Loop (Compiled Motion)
POUTA.....Compiled Output
PRUN PROF....Run a Compiled Profile
PS .....Pause Program Execution
PSET.....Establish Absolute Position
RE .....Registration Enable
REG .....Registration Distance
REGLOD.....Registration Lockout Distance
RUN PROG.....Run a Program
S.....Stop Motion
SFB .....Select Feedback Source
SGAF .....Acceleration Feedforward Gain
SGENB.....Enable a Gain Set
SGSET.....Save a Gain Set
SGVF .....Velocity Feedforward Gain
SHALL.....Hall Sensor Inversion
SRSET .....Resolver Offset Angle
STRGTD .....Target Distance Zone
STRGTE .....Target Zone Mode Enable
STRGTT.....Target Settling Timeout Period
STRGTV .....Target Velocity Zone
T .....Time Delay (Dwell)
TACC .....Transfer Command Acceleration
TACCA.....Transfer Actual Acceleration
TANI.....Transfer Analog Input Voltage
TDIR.....Transfer Programs/Profiles Stored in Memory
TGAIN .....Transfer Active Gains
TMEM.....Transfer Memory Usage
TRACE.....Program Trace Mode
TRGFN.....Trigger Interrupt Functions
TRGLOT.....Trigger Interrupt Lockout Time
TSGSET.....Transfer Gain Set
TSROFF.....Transfer Resolver Offset Angle
TSTLT .....Transfer Settling Time
V.....Velocity
VF.....Final Velocity
WAIT .....Wait for a Specific Condition

Appendix B (functional group quick reference for commands)

The format of Appendix B has changed. It lists all the commands, grouped in their respective functional groups. To review the command specifications, refer to Appendix A.

Revision B Changes:

Topic	Description
Documentation Errors	• Appendix A & B: TDIMAX is a servo-only (not stepper-only) command. • Appendix A & B: LH default for GT & GV is 0 (disabled), not 3 (enabled). • ORES: If the maximum output frequency (2.5 MHz) is exceeded, the drive will fault and TASX bit #29 and TER bit #19 is set. (previously it was stated that a configuration error occurred, reported by TASX bit #7). • TASX: - Bit #17 does not indicate a fault condition (" * " was removed). - Bits #29 and #30 do indicate fault conditions (" * " was added).

Changes for the release of OS revisions 1.01 and 1.02, and the GT:

Support for linear servo motors	(OS 1.01) The Gemini servo drives support Parker's new line of linear servo motors and positioning tables. To effectively use a linear motor, you must first provide the motor's *electrical pitch* with the DMEPIT command. **NOTE**: The Gemini 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) and the configuration tool in Pocket Motion Planner (page 11) make it easy to perform the setup in linear units. The setup/configuration tool automatically converts your setup parameters (in linear units) to the appropriate Gemini code in rotary units. You then download the generated code/file to the drive. If you are communicating to the Gemini drive 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. The commands that require conversion are: DMTD, DMTJ, DMTKE, DMTSCL, DMTLIM, DMTW, DMVLIM, DMVSCL, LDAMP, SMVER, TVE, and TVELA. For conversion equations and other details, refer to the DMEPIT description.
Motor temperature configuration	(OS 1.01) Two commands were added to enhance the internal real-time estimation of the motor winding temperature. When the winding temperature exceeds DMTMAX, the drive faults and TASX bit #30 is set. • DMTAMB (Motor Ambient Temperature) • DMTMAX (Maximum Motor Winding Temperature)
Variables added to analog monitor output A and B (DMONAV & DMONBV)	(OS 1.01) Variables #23 & #24 were added to DMONAV & DMONBV. These new variables are applicable to the GV only. • Variable #23 is *Position Setpoint*, based on the TPC value. This value is normalized as follows (value clips at ±1 rev or ±1 elec. pitch, regardless of the DMONAS or DMONBS setting): - Rotary motors: ±10V = ±1 rev (based on TPC ÷ ERES) - Linear motors: ±10V = ±1 elec. pitch (based on TPC ÷ DMEPIT) • Variable #24 is *Actual Position*, based on the TPE value. This value is normalized as follows (value clips at ±1 rev or ±1 elec. pitch, regardless of the DMONAS or DMONBS setting): - Rotary motors: ±10V = ±1 rev (based on TPE ÷ ERES) - Linear motors: ±10V = ±1 elec. pitch (based on TPE ÷ DMEPIT)
Pole pairs (DPOLE)	(OS 1.01) The minimum value for DPOLE was changed from 2 to 1.

Continued …

Torque/Force limit (DMTLIM), status	(OS 1.01) When the GV's commanded torque/force reaches the limit set by DMTLIM (TTRQ = DMTLIM), TAS bit #31 is set. TAS bit #31 remains set until you clear it with the DCLRLR command, or cycle power or issue a RESET. This is not considered a fault condition.
Command-to-product compatibility	(OS 1.01) If you attempt to send a command to a drive with which it is incompatible (e.g., sending a servo command to a stepper drive), the drive will respond with the error prompt (default error prompt is "?").
Zeroing the drive command offset	(OS 1.01) When the DCMDZ command is executed, the last voltage read at the command input will become the new zero reference point. When in velocity mode (DMODE4) or torque/force mode (DMODE2), this minimizes motor drift.
Notch filter depth	(OS 1.01) DNOTAD & DNOTBD were added to set the depth of the notch filters.
PWM Frequency	(OS 1.01) The DPWM command has been added to set the PWM frequency for the GV. This value is the internal PWM frequency as seen at the motor windings; the motor ripple current is twice this frequency.
Current loop gains (DIGN)	(OS 1.02) The valid range for DIGNA, DIGNB & DIGNC was changed from ±10 to 0-15. The valid range for DIGND was changed from ±2 to 0-1.
Configuration fault, addition	(OS 1.02) An additional TCS fault (-32367) was added to indicate when the GT drive resolution (DRES) value is too low for the number of pole pairs. To resolve the fault condition, make sure the DRES value is at least 4 times the DPOLE value.
Stall Detect Enable (ESTALL) command removed	(OS 1.02) Stall detection (GT) can be disabled only with the DSTALL0 command. Stalls are reported in TASX bit #17. If the Fault on Stall mode is enabled (ESK1), the occurrence of a stall will immediately stop pulses from being sent to the motor and will disable the drive (DRIVE0); in addition, the stall is also reported in TAS bit #12 and TER bit #1.
Velocity limit (DMVLIM)	(OS 1.02) DMVLIM is now applicable to both servo and stepper drives. Previously, it was applicable only to servo (GV) drives. Factory default values: DMVLIM50 for GT and DMVLIM200 for GV.
Steppers may use Velocity Control mode (DMODE4)	(OS 1.02) DMODE4 is now applicable to the GT drives. (The Velocity Control mode allows direct control of the motor velocity.) DMVLIM and DMVSCL have been added to the motor data table for stepper motors (thus, these parameters are auto-configured based on the stepper motor you select in the Motion Planner and Pocket Motion Planner configuration tools). The DMVSCL range for GT drives is 0-50, the default is 50. Leaving DMVSCL at zero no longer causes a motor configuration error. TASX bit #18 now applies to the GT, indicating that the commanded velocity has exceeded the DMVLIM value.

Introduction

Purpose of this Document

This document is designed as a reference for the firmware features used with the Gemini Series of digital stepper and servo drives. For hardware-related information (e.g., electrical wiring connections, specifications, setup/configuration and tuning procedures, etc.), refer to the relevant Gemini drive *Hardware Installation Guide*. These documents are available in Acrobat PDF format from the Compumotor web site (http://www.compumotor.com).

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

Table of Contents

Description of Format

	1.	2.	3.
	DRES	**Drive Resolution**	
4.	Type	Drive Configuration	**Product** / **Rev**: GT 1.02; GV 1.00; GT6 1.50; GV6 1.50
5.	Syntax	<a_><!>DRES<i> (takes effect after RESET or cycle power)	
6.	Units	Rotary motors: i = counts/rev Linear motors: i = counts/electrical pitch	
7.	Range	GT/GT6: 200-128000; GV/GV6: 200-1024000	
8.	Default	GT/GT6: 25000; GV/GV6: 4000	
9.	Response	DRES: *DRES25000	
10.	See Also	DMEPIT, DRIVE, ERES	

Item Number	Description
1.	**Mnemonic Code**: This field contains the command's mnemonic code.
2.	**Full Name**: This field contains the command's full name.
3.	**Valid Product & Revision**: This field lists the Gemini Series products and the revision of each product when this command was incorporated or modified per the description. If the command does not apply to that particular product, the **Rev** is specified as "n/a". You can use the TREV command to determine which drive operating system revision you are using. For example, if the TREV response is * TREV-GV6-L3E_D1.05_F1.00, the drive operating system revision is 1.05 and the flash boot code revision is 1.00.
4.	**Type**: This field contains the command's type. In Appendix B, you will find a list of all Gemini Series commands organized by command type.
5.	**Syntax**: The proper syntax for the command is shown here. The specific parameters associated with the command are also shown. Definitions of the parameters are described in the *Syntax* sections below. If the command requires a RESET to take effect, that is indicated as well. If you use Gemini drives in a daisy-chain or multi-drop: To address a command to a particular unit in a daisy-chain or multi-drop, you must use the *address specifier* (e.g., 2_LH0 disables both end-of-travel limit inputs on unit #2). If you leave off the address specifier, the command will affect all units on the chain. To assign addresses to multiple drives, refer to the ADDR command.
6.	**Units**: This field describes what unit of measurement the parameter (b, i, or r) in the command syntax represents. If you are using a linear servo motor, refer to the command value conversion requirements noted on page 74.
7.	**Range**: This is the range of valid values that you can specify. If you enter a value outside of the valid range, the Gemini drive will indicate an *error prompt* (see ERRLVL). Due to the digital nature of the Gemini drive, non-integer numeric data entries are rounded to the nearest valid value. The maximum difference between the entered data (to the drive) and the reported value (from the drive) is given by the ±0.nnn tolerance window. The maximum response field length is command dependent, as shown in the Response field.
8.	**Default**: The default setting for the command is shown in this field. A command will perform its function with the default setting if you do not provide a value. If you see the "**AUTO-SETUP**" note in the command description, this means that the default is automatically set according to the Parker motor you selected with the configuration wizard in Motion Planner (see page 6) or configuration tool in Pocket Motion Planner (see page 11). Some data-related commands (commands that accept binary or numeric data fields) are automatically retained in EEPROM memory *(GT6 & GV6: These commands are saved in EEPROM only if if they are executed outside of a program)*; these commands are denoted with "☒" in the list on page 187.
9.	**Response**: Some commands allow you to check the status of the command. In the example above, when you enter the DRES command by itself you will receive the response *DRES25000 (response indicates the drive resolution is set to 25,000 counts/rev). The example responses provided are based on the default error level, established with the ERRLVL4 command. If you use Gemini drives in a daisy-chain or multi-drop: To check the status on a particular unit in a daisy-chain or multi-drop, you must use the address specifier (e.g., 2_DRES to check the drive resolution of unit #2). If you leave off the address specifier, the response will be garbled characters because all units in the chain will be responding at once. To assign addresses to multiple drives, refer to the ADDR command.
10.	**See Also**: Commands related or similar to the command described are listed here.

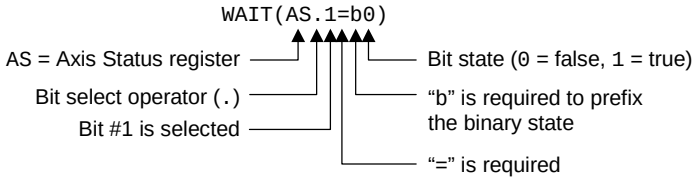
Syntax -- Letters and Symbols

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.

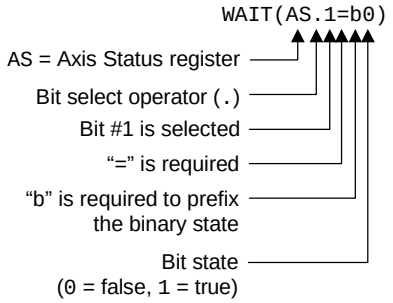
DRES — Drive Resolution

			Product	Rev
	Type	Drive Configuration		
→	Syntax	<a_><!>DRES<i> (does not take effect until RESET or cycle power)	GT	1.02
	Units	Rotary motors: i = counts/rev Linear motors: i = counts/electrical pitch	GV GT6	1.00 1.50
	Range	GT/GT6: 200-128000; GV/GV6: 200-1024000	GV6	1.50
	Default	GT/GT6: 25000; GV/GV6: 4000		
	Response	DRES: *DRES25000		
	See Also	DMEPIT, DRIVE, ERES		

Letter/Symbol	Description
a_	Represents an address specifier, numeric value from 0 to 99. An address specifier is required if multiple Gemini 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 values 1, 0, X or x; does not require field separator between values.
c	Represents a character (A to Z, or a to z).
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.
!	(GT6 and GV6 only) Represents an immediate command. Changes a buffered command to an immediate command. Immediate commands are processed immediately, even before previously entered buffered commands.
.	(GT6 and GV6 only) Bit select operator for use in IF and WAIT conditional expressions. The bit select operator allows you to base the conditional expression on one bit in the binary format of certain status registers. For example, WAIT(AS.1=b0) is a wait statement based on the condition of bit #1 in the axis status register (see TAS).
,	Represents a field separator. Commands with the symbol r or i in their Syntax description require field separators. Commands with the symbol b in their Syntax description **do not** require field separators (but they may be included).
< >	Indicates that the item contained within the < > is optional, not required by that command.



Syntax -- General Guidelines

Guideline Topic	Guideline	Examples
Command Delimiters: • Carriage return (<cr>) • Line feed (<lf>) • Colon (:)	All commands must be separated by a delimiter: carriage return, line feed, or colon. A carriage return is the most commonly used. The colon (:) allows you to place multiple commands on one line of code.	Set encoder resolution to 4000 counts/rev: ERES4000<cr> ERES4000<lf> ERES4000: DRES25000: DMODE2 <cr>
Neutral Characters • Space (<sp>) • Tab (<tab>)	Using neutral characters anywhere within a command will not affect the command.	Set encoder resolution to 4000 counts/rev: ERES<sp>4000<cr>
Comment Delimiter (;)	All text between a comment delimiter and a command delimiter is considered *program comments* (program comments are not stored in the Gemini).	Add a comment to the command: ERES4000<tab> ; set encoder res.
Case Sensitivity	There is no case sensitivity. Use upper or lower case letters within commands.	Set input active levels on inputs 1-3: INLVL001<cr> inlvl001<cr>
Field Separator (,)	Some commands with the symbol r or i in their Syntax description require field separators. Commands with the symbol b in their Syntax description **do not** require field separators.	Set the beginning-of-transmission characters: BOT13,10,26<cr> Set outputs 1-6 active low and 7 to active high: OUTLVL0000001<cr>
Bit Select Operations	Within IF and WAIT conditional expressions, you can use the *bit select* method to base the IF or WAIT operation on the condition of a specific status register bit. This allows a short-cut alternative to typing in the entire binary condition of the selected status register. The general syntax for the expression is as follows: (<register ID> . <bit #> = b <bit state>) **NOTE**: If you upload a program from the Gemini, bit-select conditional expressions are translated to the equivalent masked binary expression. For example, IF(AS.12=b1) translates to IF(AS=bXXXXXXXXXXX1).	Wait until no motion is commanded – this condition exists when bit #1 of the axis status register (see TAS) is set to zero (0). 

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

Programming Interface Tools

Two graphical programming interfaces, Motion Planner™ and Pocket Motion Planner™, are provided as tools for programming your Gemini 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 (Motion Planner only)
- 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. Pocket Motion Planner runs on a hand-held PC using the Windows CE operating system.

Motion Planner and Pocket Motion Planner are installed from the "Motion Planner" CD which is included in your Gemini 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 Gemini (as well as 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 on page 195.

NOTE

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

Using Motion Planner with a Gemini 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 Gemini product.

Installing Motion Planner:

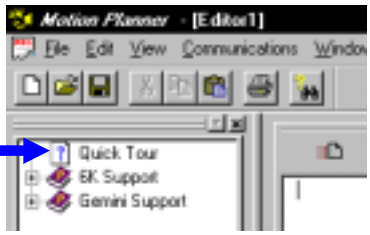
System Requirements:

- 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:

Click the "Install Motion Planner" button. The installation program installs Motion Planner on your hard drive and creates a Motion Planner group in the Programs menu accessed from the Start button.

Orientation to Motion Planner: After you launch Motion Planner, browse the "Quick Tour" to orient yourself with the Motion Planner interface.



6K Users **CAUTION** **6K Users**

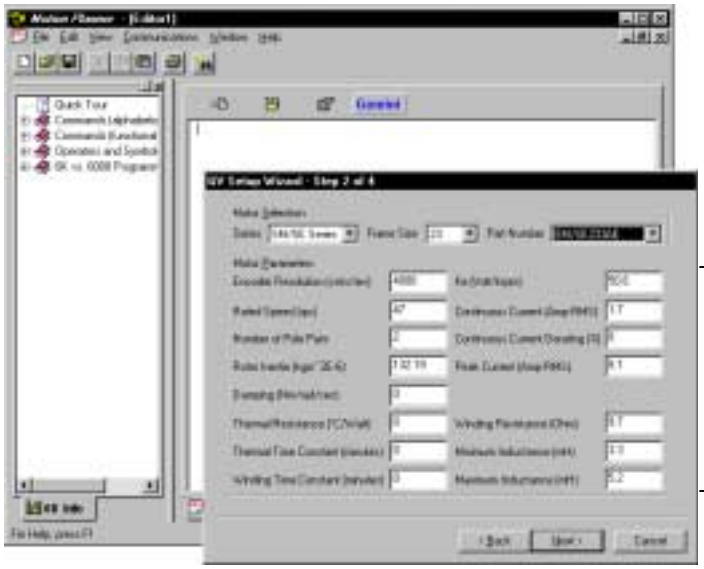
If you are using a 6K Controller and Gemini Drives, you should try to use two COM ports – one for the 6K and one for the Gemini Drive. If your computer has only one COM port, you will have to swap the serial cable connection between the 6K and the Gemini. Be aware that you must quit Motion Planner before swapping the serial cable between products. After the cable swap, you may launch Motion Planner and select the newly connected product. Failure to quit Motion Planner before the swap will corrupt communications with the attached Gemini or 6K.

Configuring Your Gemini Drive (see graphic below):

1. Make sure that you have installed the Gemini according to the instructions in the *Hardware Installation Guide*.
2. Launch Motion Planner. When the product selection dialog appears, select a Gemini drive and select the COM port to which the Gemini is connected.
3. In the Editor window, click on the Gemini button to launch the setup wizard.
4. Select either "Express" or "Full" setup, and select one of these Wizard Initialization methods:
   - Use factory defaults. This method populates the wizard with the defaults as noted in the command descriptions in this document.
   - Use the current contents of the Editor window. This method populates the wizard with the data from the Editor window. The general intent of this method is to revise the drive's configuration, based on the contents of a saved file or the current configuration of the attached Gemini drive.
   - Use the current configuration of the attached drive. This method uploads the drive's configuration file and populates the wizard with those settings.
5. Click the "Next" button to proceed with the wizard. Fill in the wizard dialogs as prompted. At the end of the wizard, click the "Finish" button; this creates the setup code and places it in the Editor window (at the location of the cursor).

6. Optional: Modify the setup code if needed.
7. Click the button to save the setup code to a file (*.prg) on your hard drive.
8. Click the button to download the setup code (contents of the Editor window) to the Gemini drive. When the download is complete, Motion Planner asks you if you wish to reset the drive to implement the new setup code.

   Drive setup is complete. All of the setup parameters (command values) are stored in the Gemini drive's EEPROM and are automatically recalled when you cycle power or reset the drive. If you wish to return the drive to factory settings, use the `RFS` command.



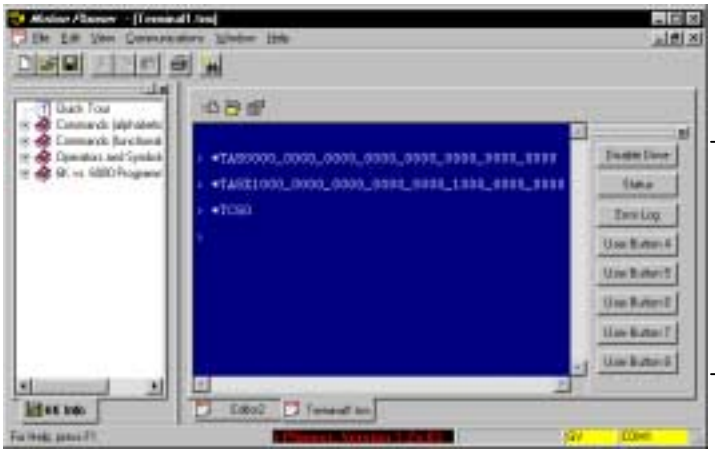
These motor parameters are automatically filled in based on the selected Parker motor.

If you are using a non-Parker motor, or a custom motor that is not listed, you will have to fill in all of these motor parameters by hand. Failure to do so will result in a *motor configuration error*, which prevents the drive from being enabled.

This illustration shows an example of the motor configuration portion of the wizard.

Communicating with the Gemini drive (see graphic below):

Click on the Terminal tab to view the terminal emulator window. From the terminal window you can type in status commands and setup commands as needed. You can also download stored files to the drive (click on), and upload the drive's current configuration to a file (click on).



These buttons can be configured to send one or more commands to the Gemini drive.

In this example, the "Disable Drive" button sends the `DRIVE0` command, "Status" sends the `TAS : TASX : TCS` command string, and "Error Log" sends `TERRLG`. The window shows the response from pressing the "Status" button.

For configuration details, open the Quick Tour and see "Terminal Emulator".

Updating the Drive's Operating System:

Gemini 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*_*nn*.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. Connect the Gemini drive to your computer's RS-232 serial communication port (see instructions in the Gemini drive's *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).
2. Launch Motion Planner.
3. In the Default Communications Settings dialog box, select your Gemini 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 Gemini 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. During the download, the Gemini drive's left-hand LED flashes red and the right-hand LED flashes yellow.
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`").

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

Using Pocket Motion Planner

Pocket Motion Planner is a programming interface for the Gemini product family. Pocket Motion Planner runs on a hand-held and palm PCs using the Windows CE operating system.

To Install Pocket Motion Planner:

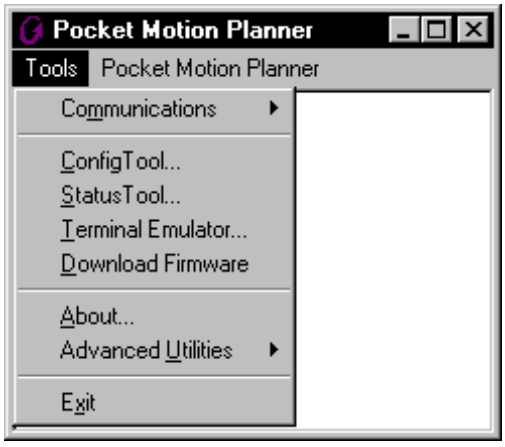
System Requirements:

- IBM-compatible PC with:
  - CD-ROM drive or internet access for installation
  - Windows CE Services v2.2 or greater for installation
  - Serial port
- Hand-held or palm PC with:
  - Microsoft Windows CE v2.0 or greater
  - 4MB RAM (8MB recommended for speed)
  - Hard disk space: 500 KB minimum
  - Display: 240x320 or 640x240; gray scale or full color
  - Keyboard: soft or built-in
  - Processor: MIPS 4000 or SH3
  - RS-232C serial port, configurable for 9600 baud; null modem serial cable

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

Click the "Install Pocket Motion Planner" button. The installation program installs Pocket Motion Planner on your hand-held or palm PC and creates a Motion Planner shortcut that is accessed from the Start button.

Overview of Pocket Motion Planner Tools:



- Communications (see page 10) – Set device address, select COM port, test COM port.
- Config Tool (see page 10) – Configure the drive's operational settings.
- Status Tool (see page 12) – View drive status reports. Particularly useful for troubleshooting and initial system setup.
- Terminal Emulator (see page 13) – Interactive communication to send commands and receive status information from any (or all) Gemini drive(s) connected via RS-232.
- Download Firmware (see page 13) – Download an updated operating system to the drive.
- About – Display the revision of Pocket Motion Planner
- Advanced Utilities (see page 14) – For complex troubleshooting tasks, to be used only at the direction of Compumotor support personnel.

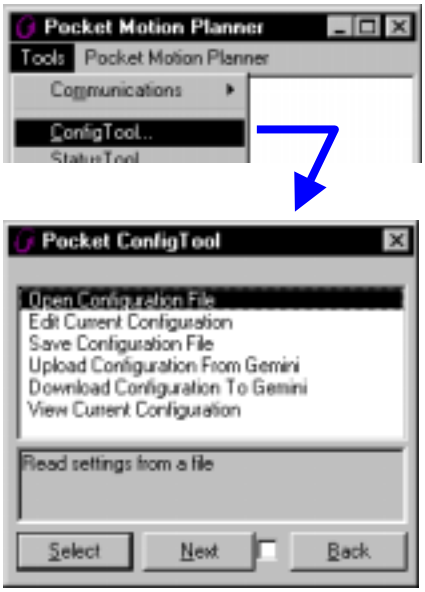
◆ Communications Settings:

Comm Port – Allows you to select the appropriate RS-232 Comm port; Gemini requires 9600 baud.

Device Address – Sets the default device address for communications within the Config Tool, Status Tool, and Download Firmware utility. The default address will reset to 0 each time Pocket Motion Planner is started, which is the factory default device address for Gemini drives. Pocket Motion Planner assumes communication with one drive only, though that drive could be part of a daisy chain.

Test Comm Port – Tests your Comm Port and RS-232 cabling by sending out a command and expecting a reply from a connected Gemini drive. You are notified if the test has passed or failed. Tips on RS-232 troubleshooting can be found in the Troubleshooting section of the GT or GV *Hardware Installation Guide*.

◆ Configuration Tool: (drive configuration procedure on page 11)



Open Configuration File – Allows you to browse you PC for previously saved configuration files and bring them into the configuration editor. Configuration files are typically saved with the .PMP extension, but other files may be opened.

Edit Current Configuration – Allows you to view and modify all available configuration parameters relevant to your Gemini drive. NOTE: The Config Tool is an *editor*; as values are modified, they are not sent to the drive until you use "Download Configuration to Gemini" (exceptions: stepper Interactive Motor Matching and servo Graphical Tuning). Editor contents may include:

- Factory default values
- A previously saved configuration file
- Configuration uploaded from a Gemini drive
- Any user modifications to the above

Save Configuration File – Saves changes to currently active configuration file. The default file extension is .PMP and the default directory is My Documents\Gemini; however, other extensions and locations are allowed.

Upload Configuration from Gemini – Retrieves the configuration from the connected Gemini drive. A communication check is done when an upload is attempted and you are notified of communication problems if the upload can not be completed. Upon a successful upload, parameters are immediately available to you by choosing Edit Configuration File or View Current Configuration.

Download Configuration to Gemini – Sends the current contents of the configuration editor to the connected Gemini drive. A communication check is done when a download is attempted and you are notified of communication problems if the download can not be completed. You can either download to one particular device address, or all units connected in a daisy chain.

View Current Configuration – View of configuration editor contents. Parameters can not be modified in this function, simply viewed for quick reference.

Configuring Your Gemini Drive:

Before you start: Make sure that you have installed the Gemini according to the instructions in the *Installation Guide*.

1. Select **Config Tool** from the Pocket Motion Planner **Tools** menu.

2. Select **Edit Configuration File** from the Config Tool main menu. If starting from scratch, do this immediately upon entry into the Pocket Config Tool. To modify a previous configuration, first choose **Upload Configuration from Gemini** to use the current configuration from your drive or choose **Open Configuration File** to edit a previously saved configuration file.

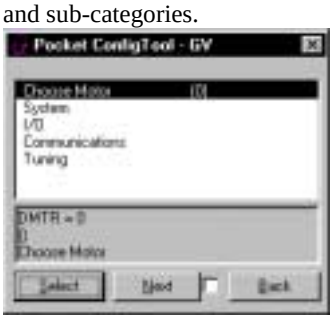
3. Select **Drive Type**. Choose you drive from the list or choose **Auto-Detect** if the drive is connected via RS-232. If the editor contains a previously saved, modified, or uploaded configuration, then the Drive Type will already be set.

4. Select either **Express Configuration** or **Full Configuration**:

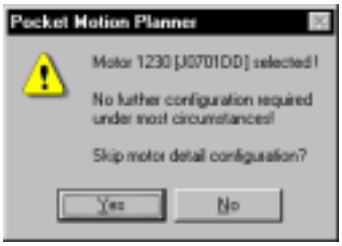
   Express Configuration displays a small subset of available drive parameters. This is useful for quick bench top tests, or when minimal modification of parameters is required.

   Full Configuration displays all available drive parameters, grouped into relevant categories and sub-categories.

5. Select **Choose Motor**.
   If using a Parker motor:
   When you select your Parker motor, Pocket Motion Planner automatically configures all the necessary motor parameters accordingly (see parameter list on page 84). Factory motor settings can be modified (**Configure** button), but this is not recommended and should be done with extreme caution, as using settings other than factory defaults can have unexpected and/or hazardous results. If you modify Parker motor settings and wish to revert to the factory defaults, re-select the motor. Click the **OK** button when finished; you will see a message similar to this:

   In most situations, click **Yes** to skip the motor detail configuration, because the Parker motor selection already configures this for you.

   If using a non-Parker motor, or a custom motor not on the available listing, select "**Other**" within the Choose Motor menu (**Other** is available only if you select **Full Configuration** in step 4 above). It is mandatory that all motor details be configured for "Other" motors to avoid a *motor configuration error*, which prevents you from enabling your drive. You will automatically be placed in the motor configuration menu when you select **OK** to finalize your motor selection of "Other". When you finish in the motor configuration menu, click the Back button.

6. Modify other relevant drive parameters within the Express or Full Configuration menu. For help on each configuration item, refer to the respective command description in this manual.

7. Exit the configuration editor by clicking on Back until you see the main Config Tool window. →

   Note that clicking on the ✕ in the upper right hand corner of any screen within Edit Current Configuration will attempt to close the entire Config Tool. The Back button is recommended for this basic procedure.

8. Select **Save Configuration File** to save the current configuration to a file.

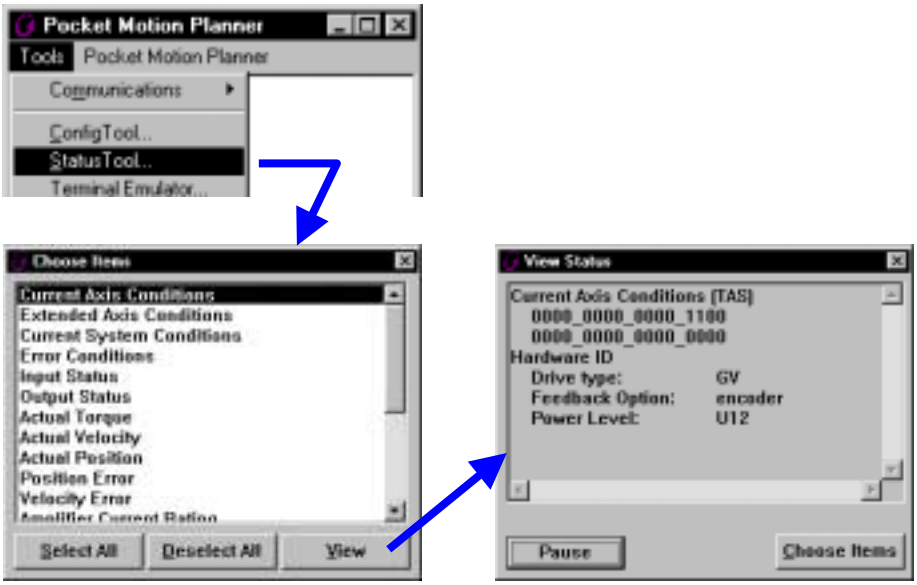
9. Optional: Before you download the configuration file to the drive, it may be useful to **View Current Configuration** to make sure you are sending the appropriate settings.

10. Select **Download Configuration to Gemini** to download the current configuration file to your Gemini drive. When prompted to Reset the drive, click **Yes** to make sure the new configuration is activated in the drive.

    Drive configuration is complete. All of the configuration parameters (command values) are stored in the Gemini drive's EEPROM and are automatically recalled when you cycle power or reset the drive. If you wish to return the drive to factory settings, use the RFS command.

◆ Status Tool:

Particularly useful for troubleshooting and initial system setup, the Status Tool allows you to select from an extensive list of drive status reports and watch them update continuously.



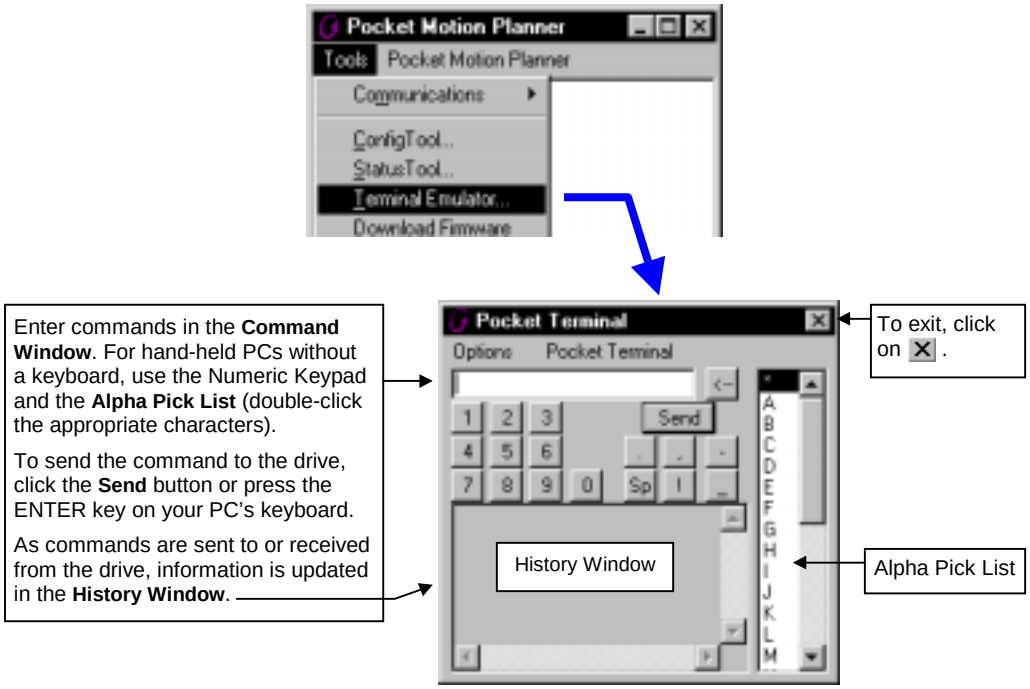
Select one or more items from the list available in the **Choose Items** screen. Click on **View** to view the items in the View Status screen.

The display window is updated every 0.1 seconds. The update can be frozen by clicking on **Pause**, and restarted with **Resume**.

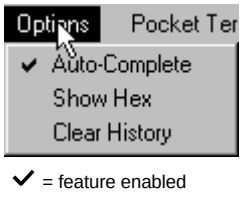
To exit the Status Tool, click on ✕ in the upper right hand corner of the window.

◆ Terminal Emulator:

The Terminal Emulator allows interactive communication with any (or all) Gemini drive(s) connected via RS-232.



Options Menu:



Auto-Complete – Automatically adjusts the available alpha characters in the Alpha Pick List based on your current selection in the Command Window. Also completes the command entry when there are no more alpha character options. *This is particularly useful for hand-held PCs with no keyboard interface.*

Show Hex – Allows you to enter standard ASCII commands and view the hexadecimal equivalents of the commands and responses in the History Window.

Clear History – Clears the contents of the History Window.

◆ Updating the Drive's Operating System (using the "Download Firmware" tool):

Gemini 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_nn*.ops. For example, the operating system file for version 1.50 is called GEM_1_50.ops. Be sure to note the hard drive directory to which you download the file.

Update Procedure:

1. Transfer the new operating system file to your hand-held PC (or other Windows CE device) using the Win CE Services utilities provided with the hand-held PC. Place the file in the My Documents\Gemini directory.
2. Connect the Gemini drive to your hand-held PC's RS-232 serial communication port (see instructions in the Gemini drive's *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, no RS-485).
3. Launch Pocket Motion Planner. You may need to select the serial port to which the drive is connected (from the **Tools** menu, select **Communications** and then **Comm Port...**).
4. From the **Tools** menu, select **Download Firmware**. When presented with the **Choose Firmware File** dialog, locate the operating system file and click the **Open** button. When you are asked if the drive is already in download mode, check to see if the drive's left-hand LED is flashing red — if it is click **Yes**, if it is not click **No**.

   This initiates the download to the drive and displays the download status dialog. During the download, the Gemini drive's left-hand LED flashes red and the right-hand LED flashes yellow. When the download is completed successfully, Pocket Motion Planner displays a confirmation message. Also, the drive automatically resets itself (you may notice that the fault relay toggles at this time).

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

5. From the **Tools** menu, select **Terminal Emulator** to open the terminal emulator window.
6. In the terminal emulator, issue the TREV command to verify that the proper operating system revision is now in the drive. For example, the response "`*TREV-GV-L3E_D1.05_F1.00`" indicates that the drive is using revision 1.05, denoted by "`D1.05`".

◆ Advanced Utilities:

> ⚠ **WARNING**: The Advanced Utilities are intended for complex troubleshooting tasks and should only be used at the direction of Compumotor support personnel. Modifying values or changing settings within the Advanced Utilities can lead to computer or Pocket Motion Planner problems if not done properly.

Translator – This tool allows you to translate the standard ASCII commands (referred to as "6000 commands") into the actual hexadecimal value that is internally used by the Gemini drive, and vice versa. Note that no actual communication to a product takes place within the translation utility, it simply allows you to see the actual hex command format of commands sent to the Gemini drive.

Advanced Options – Allows you to change these default operational parameters for Pocket Motion Planner:

- Document Path: This is the default path used for file operations (save and open).
- Motor File: This is the file used to auto-configure the motor parameters by Parker motors.
- Log File
- Verbose Mode
- Terminal Emulator Option menu settings (see page 13):
  - Auto-complete
  - Show Hex

Troubleshooting *(refer also to the back cover)*

Error Messages

Depending on the error level setting (set with the ERRLVL command), when a command error is created, the Gemini drive will respond with an error message and/or and error prompt. A list of all possible error messages is provided in the 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 Gemini drive responds with both the error message and the error prompt. At error level 3 (ERRLVL3), the Gemini drive responds with only the error prompt.

Error Response	Possible Cause
INVALID_ADDRESS	Address format is not correct (e.g., while attempting to address the DRES command to daisy-chain unit #2, you issue the 2DRES command instead of the 2_DRES).
INVALID_DATA	Data for a command is illegal (e.g., INLVL0011XX).
INVALID_DATA_HIGH	Data for a command is above the maximum value.
INVALID_DATA_LOW	Data for a command is below the minimum value.
UNDEFINED_LABEL	Command issued to the Gemini is not a recognized command or program name.

Diagnostic LEDs

Green/Red (left-hand LED)	**Yellow/Green** (right-hand LED)	**Meaning**
--- *none* ---	Yellow	Initialization in progress. This occurs during power-up and reset.
Green	--- *none* ---	Drive Ready.
Green	Green (flashing)	Incoming Steps (variable rate).
Green	Yellow/Green (flashing)	Autorun Mode (DMODE13).
Red (flashing)	--- *none* ---	Awaiting new operating system download. (Motion Planner users see page 8 for download procedure; Pocket Motion Planner users see page 13 for download procedure)
Red (flashing)	Yellow (flashing)	Downloading new operating system.
Red	Green	Keep-Alive Mode. No AC power, but running off 24VDC only.
Red	--- *none* ---	Drive Faulted. Check TAS and TASX responses to find out why.

Status Commands

Status commands are provided to assist your diagnostic efforts. These commands display status information such as, axis-specific conditions, general system conditions, error conditions, etc.

Checking Specific Setup Parameters
One way to check the conditions that are established with a specific setup command is to simply type in the command name without parameters. For example, type "ERES" to check the encoder resolution setting; the response would look something like: *ERES4000.

Below is a list of the status commands that are commonly used for diagnostics. Additional status commands are available for checking other elements of your application (refer to "transfer" in the index). For more information on each status command, refer to the respective command description in this manual.

GT6 & GV6 only: To send a status command to the Gemini drive while a program is executing, prefix the command with an "!" (e.g., !TAS). This allows you the opportunity to check certain conditions concurrent with program and machine processes.

- Commonly used status commands (binary status bits are numbered 1 to n, from left to right):

  TERRLG.....Error log reports the last 10 error conditions, which includes the TAS, TASX, TDHRS, TDTEMP, and TMTEMP reports. The error log may be cleared with the CERRLG command.

  TAS.....General report, including fault conditions.

  TASX.....Additional report of conditions not covered with TAS.

  TCS.....If TASX bit #7 or bit #28 is set, you can identify the cause with TCS.

  TER.....Error status report. (Many of the error conditions are also reported with TAS or TASX.)

  TINO.....Bit #6 indicates the hardware status of the Enable input ("1" = OK to enable drive).

  TIN.....Status of digital inputs on the DRIVE I/O connector.
  GT6 & GV6: You can use INFNC*n* to report the state and programmed function of input *n*.

  TOUT.....Status of digital outputs on the DRIVE I/O connector, including the Relay output.
  GT6 & GV6: You can use OUTFNC*n* to report the state and programmed function of output *n*.

  TPROG .....(GT6 & GV6 only) Displays the contents of the specified program (e.g., TPROG PROG2 displays the contents of program #2, defined with the DEF PROG2 command). TPROG cannot be used to display profiles defined with DEF PROF.

Trace Mode (GT6 and GV6 only)

The Trace Mode helps you debug programs. When in Trace Mode, enabled with the TRACE1 command, all commands executed are or transferred out the serial connection (RS-232 or RS-485) and displayed in the Motion Planner Terminal window; this allows you to track, command-by-command, the entire program as it runs. TSS bit #8 is set when the Trace Mode is enabled. A sample scenario is provided on page 174.

Solutions to Common Problems

Refer also to these troubleshooting tools:

Problem	Cause	Remedy
I can't communicate with the Gemini when using my own terminal emulation program.	The Gemini drive requires a translator, which is available only when using Motion Planner, Pocket Motion Planner, and the COM6SRVR Communications Server.	Use Motion Planner (page 5), Pocket Motion Planner (page 9), or the COM6SRVR Communications Server (page 195) to communicate with the Gemini drive. These tools are available for Windows 95/98 and Windows NT operating systems.
I've entered new command values, but they are not invoked in the Gemini drive.	Some commands require a reset (cycle power, RESET, or Reset input) before the new values are invoked in the drive. The command list on page 187 identifies the commands that require a reset. GT6/GV6: Commands that are executed within a program are not stored in the Gemini's EEPROM memory and are lost after a reset.	Execute the RESET command or activate the Reset input. Refer to page 18 for additional details.
All programmed settings have returned to their factory default values. GT6/GV6: All stored programs and profiles have been deleted.	You executed the RFS command. Before executing RFS, make sure you have a backup of the configuration file and program files. If you didn't save the original files, you can upload the present settings and programs from the Gemini and save them to files on your hard drive: • Motion Planner (Editor window): Click  to upload and view, then click  to save the file. • Pocket Motion Planner: Open the Config Tool (see page 10), select "Upload Configuration From Gemini", then select "Save Configuration File" to save the file. (**NOTE**: Program files cannot be uploaded and saved with Pocket Motion Planner.)	GT/GV: If you have saved the drive configuration file, download the file from Motion Planner or Pocket Motion Planner. If you did not save drive configuration file, the settings are lost. GT6/GV6: Restore the settings by downloading the program files saved to your hard drive (see programming process demonstrated on page 20).
GT6/GV6: There is a long pause (cannot communicate with the drive) when I download a program or profile, or execute END, DEL, RESET or RFS.	These processes require writing to the Gemini's EEPROM. When XON/XOFF is enabled (this is the factory default condition – see XONOFF), the Gemini temporarily disallows (XOFF) communication from the PC during the write to the EEPROM, and when the command buffer is full (e.g., when streaming large amounts of commands to the Gemini). If the serial connection is lost (cable disconnect, drive is powered down, etc.) during an XOFF, you will have to restart Motion Planner or Pocket Motion Planner (or the COM6SRVR).	N/A
GT6/GV6: Motion does not occur, the motor has lost torque, output #2 is activated, and the dry contact relay is activated.	A "Fault condition" exists. ← (The outputs will be activated only if they are left in their factory default OUTFNC configuration.) →	One or more conditions can cause a "Fault condition", which automatically disables the drive (DRIVE0), activates output #2, and opens the dry contact relay. Check the possible causes with TAS and TASX.
GT6/GV6: When I execute a GO, no motion occurs. The only indication of an error is the ERRBAD prompt (which, by default, is a "?" character). No errors are reported with TAS, TASX, or TER.	You have executed an s-curve average accel/decel command (AA, ADA, LHADA, or LSADA) with an illegal value.	Change the s-curve accel/decel command value to a valid range, or set it to zero to restore trapezoidal profiling. Refer to the table on page 54 for assistance.

When are Commands Executed?

GT and GV Drives:

All commands are either implemented immediately upon being sent to the drive, or they are implemented after you cycle power to the drive or reset the drive. To reset the drive, issue a RESET command or activate the Reset input (pin #3 on the DRIVE I/O connector). If a command requires a reset, it is noted in the command description.

GT6 and GV6 Drives:

All commands are "buffered", which means they are placed in the drive's command buffer and executed in the order in which they are received. If the command syntax contains "<!>", then it may be made "immediate" by prefixing it with the "!" character (e.g., A is buffered, !A is immediate). When an immediate command is sent to the drive, it is executed ahead of any existing buffered commands.

Some commands are not implemented until after you cycle power to the drive or reset the drive. To reset the drive, issue a RESET command or activate the Reset input (pin #3 on the DRIVE I/O connector). If a command requires a reset, it is noted in the command description.

Gemini Major Programmer's Guide

This portion of the manual provides guidelines for implementing GT6 and GV6 firmware features.

Before attempting to implement firmware features, make sure you have installed the Gemini drive according to the instructions provided in the drive's *Hardware Installation Guide*.

In this section:

Programming Fundamentals

Creating Programs

A *program* is a series of commands. These commands are executed in the order in which they are programmed. Immediate commands (commands that begin with an exclamation point [!]) cannot be stored in a program. Only buffered commands may be used in a program. As shown in the scenario below, the program is defined with the `DEF PROG` command, followed by the contents of the program, and completed by the `END` command. Up to 32 programs may be defined and stored in the Gemini drive.

A *subroutine* is defined the same as a program, but it is executed with an unconditional branch command, such as `GOSUB` or `JUMP`, from another program. Subroutines can be nested up to 16 levels deep. NOTE: The Gemini products do not support recursive calling of subroutines.

Compiled profiles are defined similar to programs, using the `DEF PROF` and `END` commands (the profile is automatically compiled when the Gemini drive executes the `END` command), and executed with the `PRUN PROF` command. Up to 16 compiled profiles may be defined and stored in the Gemini drive. Compiled profiles also affect a different part of the product's memory, called *compiled memory*. For information on compiled profiles, refer to page 49.

Programming Scenario

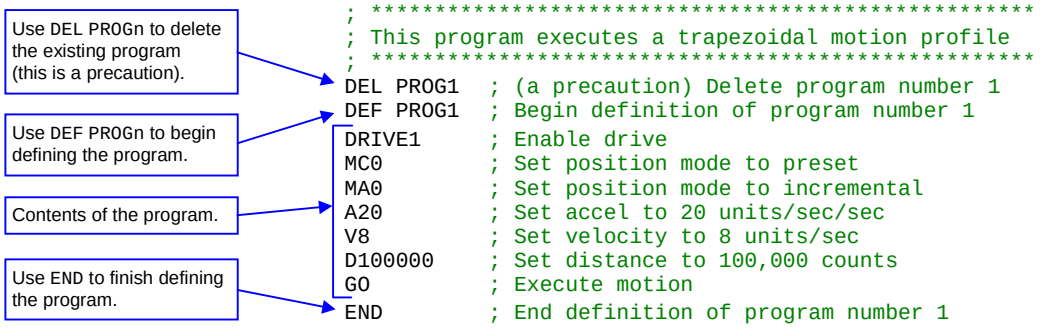
To best understand the process of developing Gemini programs, we invite you to follow along on a program development scenario. The scenario covers these programming tasks:

1. Create a simple motion program in the Motion Planner Editor.
2. Save the program file.
3. Download the program to the Gemini product.
4. Verify that the program is stored in the Gemini's memory (`TDIR`, `TMEM`)
5. Execute (run) the program from the Terminal.

FIRST: If you have not already done so, install Motion Planner (see page 5) and launch the Motion Planner application. Also, establish a serial connection between the Gemini drive and your computer and power up the Gemini drive.

1. Create a program.

Using Motion Planner's Editor, type in the commands as shown in the sample below. The sample also shows program comments to help you understand the purpose of each command and the implications of executing motion. Notice that the comments are placed after the comment delimiter (`;`).




```
; ************************************************
; This program executes a trapezoidal motion profile
; ************************************************
DEL PROG1   ; (a precaution) Delete program number 1
DEF PROG1   ; Begin definition of program number 1
DRIVE1      ; Enable drive
MC0         ; Set position mode to preset
MA0         ; Set position mode to incremental
A20         ; Set accel to 20 units/sec/sec
V8          ; Set velocity to 8 units/sec
D100000     ; Set distance to 100,000 counts
GO          ; Execute motion
END         ; End definition of program number 1
```

2. Save the program.

CAUTION: Programs and profiles stored in the Gemini drive's EEPROM can be deleted with the `DEL PROG` and `DEL PROF` commands, and by executing the `RFS` command. Therefore, to safeguard the contents of your programs, you should save your program files to your hard drive.

To save the program file, use the **File > Save** command or click the  button. Call it "example.prg".

3. Download the program.

To download the program to the Gemini drive, click the  button in the Editor window. When you are presented with a dialog asking if you wish to reset the drive after the download, click "No" (this sample program does not require a reset).

4. Verify the download.

Switch to Motion Planner's Terminal window, type `TDIR` and press ENTER. The drive should respond with "`*PROG1`". This verifies that the program you just downloaded actually resides in the Gemini drive's EEPROM.

TIP: To check the amount of memory (bytes) available for storing additional programs and profiles, use the `TMEM` command. To check the contents of a specific program, use the `TPROG PROGn` command ("n" is the program number); `TPROG` cannot be used to check the contents of defined profiles.

5. Run the program.

WARNING

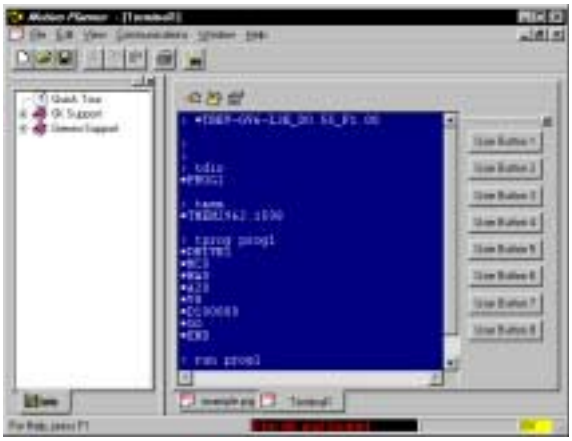
Executing the sample program will cause motion. Make sure it is safe to move the load without damaging equipment or injuring personnel.

In the Terminal, type `RUN PROG1` or `PROG1` to run the program. The motor should make one 100,000-count move in the positive direction.

Congratulations! You've just created and executed your first Gemini motion program. The illustrations below show the contents of Motion Planner's Editor window and Terminal window after completion of the scenario.

Motion Planner (viewing the Editor window)

Motion Planner (viewing the Terminal window)



Using a Setup Program

The intent of the Setup program is to place the Gemini drive in a ready state for subsequent motion control. The setup program typically contains elements such as drive, motor and feedback device configuration, tuning gain selections, programmable I/O definitions, homing configuration, etc.

The basic process of creating a setup program is:

1. Create a program to be used as the setup program. The easiest way to create the setup program is to use the setup wizard. If you are using Motion Planner, refer to page 6. If you are using Pocket Motion Planner, refer to page 11.
2. Save the program and download it to the Gemini drive.
3. There are two main options for executing the setup program:
   a. Execute the `STARTP` command to assign your new program as the "start-up" program (e.g., `STARTP PROG1` assigns program #1 as the start-up program). Thereafter, the assigned start-up program is automatically executed when the Gemini drive is powered up, when the `RESET` command is executed, or when the hardware Reset input (pin 3 on the DRIVE I/O connector) is activated.
   b. Call the setup program from the main program for your application. For example, if the setup program is program #1, at the appropriate location in the main program you could put a `JUMP PROG1` command or a `GOSUB PROG1` command to branch to program #1.

Executing Programs (options)

Following is a list of the primary options for executing programs stored in your controller:

Method	Description
Execute from a terminal emulator	In the Motion Planner terminal or the Pocket Motion Planner terminal, type in the `RUN PROGn` command or the `PROGn` command to execute program *#n*. This is demonstrated in the programming scenario beginning on page 20.
Execute as a subroutine from a "main" program	Use a branch (`GOSUB`, `RUN PROG` or `JUMP`) from the main program to execute another stored program.
Execute automatically when the controller is powered up	Assign a specific program as a startup program with the `STARTP PROGn` command. Thereafter, the assigned start-up program is automatically executed when the Gemini drive is powered up, when the `RESET` command is executed, or when the hardware Reset input (pin 3 on the DRIVE I/O connector) is activated.
Execute a specific program with BCD weighted inputs	Define programmable inputs to function as BCD select inputs, each with a BCD weight. A specific program (identified by its number) is executed based on the combination of active BCD inputs. Related commands: `INSELP` and `INFNCi-B`.
Execute from your own custom Windows program	Use a programming language (e.g., Visual Basic, Visual C++, etc.) and the Gemini Communications Server (provided on the Motion Planner CD) to create your own windows application to control the Gemini product. Refer to page 195 for details on the Communications Server.

Controlling the Execution of Programs and the Command Buffer

The Gemini drive allows you to determine the drive's handling of motion and program/command execution during normal operating events.

`COMEXC` (page 62) controls processing of motion commands received while motion is in progress.

`COMEXL` (page 63) controls whether the command buffer is saved upon encountering a hardware or software end-of-travel limit.

`COMEXR` (page 63) controls how the activation of a pause/continue input (`INFNCi-E`) impacts motion and program execution.

`COMEXS` (page 64) controls how the execution of a stop command (`S`) or the activation of a stop input (`INFNCi-D`) impacts motion and program execution and the command buffer.

Program Flow Control

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

You can affect program flow with:

Control Method	Related Commands
Unconditional Loop. Repeats the execution of a group of commands (located between the `L` and `LN` commands) for a predetermined number of iterations.	`L`.....pg. 125 `LN`.....pg. 128
Unconditional Branch. Flow of program execution ("control") passes to the program specified in the branch command. Using `GOSUB`, `RUN PROG`, or `PROG`, processing returns to the original ("calling") program. Using `JUMP`, processing does not return to the calling program.	`GOSUB` .....pg. 112 `RUN PROG` or `PROG` ....pg. 146 `JUMP` .....pg. 123
Wait Statement. Pauses program execution until the specified condition evaluates true.	`WAIT` .....pg. 184
"If" Statement. Executes a specific set of commands (between `IF` and `NIF`) only if a certain condition exists.	`IF`.....pg. 116 `NIF`.....pg. 132 `ELSE` .....pg. 100
Time Delay (Dwell). Impose a time delay between moves or other programmed events.	`T`.....pg. 157
Pause. Activate a "pause" input (an input configured with the `INFNCi-E` command) to pause program execution (and motion if in the `COMEXR1` mode). To resume, deactivate the input or send a `!C` command to the Gemini. Another way to pause program execution (not motion) is to place a `PS` command in the program – when the program is run, it will pause at the `PS` command. To resume, send a `!C` command.	`INFNC` .....pg. 118 `COMEXR` .....pg. 63 `C`.....pg. 61 `PS`.....pg. 140
Stop. Use a `S1` or `!S1` command to stop motion only. Use a `S` or `!S` command or use a "stop" input (an input configured with the `INFNCi-D` command) to stop motion and program execution. `COMEXS` controls whether motion and program can be resumed with `!C` or a resume input (`INFNCi-E`).	`S`.....pg. 147 `INFNC` .....pg. 118 `COMEXS` .....pg. 64 `C`.....pg. 61
Kill. Use a `K1` or `!K1` command to kill motion only Use a `!K` or `K` command, or a "kill" input (an input configured with the `INFNCi-C` command) to kill motion and terminate program execution.	`K`.....pg. 124 `INFNC` .....pg. 118

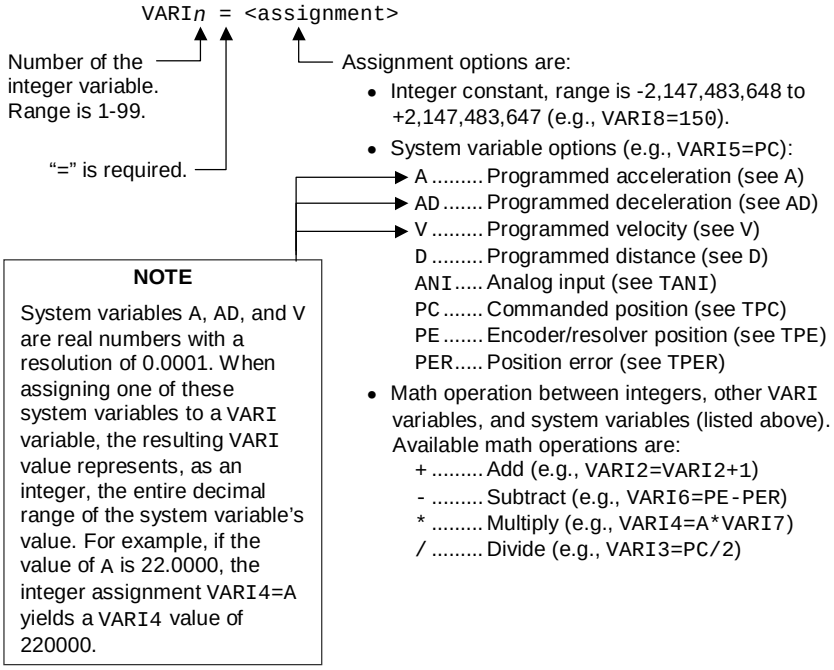
Using Variables

With the release of OS version 1.60, the GT6 and GV6 drives now allow you to define up to 99 user variables (integer variables). Integer variables are represented by the syntax VARI*n*, where "*n*" is the number of the variable (range is 1-99). Integer variables may be used for:

- Variable assignments and math operations
- Command value substitutions
- Variable comparisons in conditional expressions

Variable Assignments and Math Operations

Variable assignment allows you to specify the value of integer variables in terms of integer constants, system variables, or mathematical operations between integers, other VARI variables, and system variables.



To check the present value of an integer variable, execute the VARI*n* command with no other arguments. The drive responds with the present integer value.

NOTE

System variables A, AD, and V are real numbers with a resolution of 0.0001. When assigning one of these system variables to a VARI variable, the resulting VARI value represents, as an integer, the entire decimal range of the system variable's value. For example, if the value of A is 22.0000, the integer assignment VARI4=A yields a VARI4 value of 220000.

Command Value Substitutions

Variable substitution allows you to substitute the value of a VARI variable as the parameter value for certain commands. For example, if VARI9 is substituted for the L loop parameter L(VARI9), and the value of VARI9 is 7 when the L command is executed, the Gemini will initiate a 7-iteration loop.

The commands that allow parameter substitutions are:

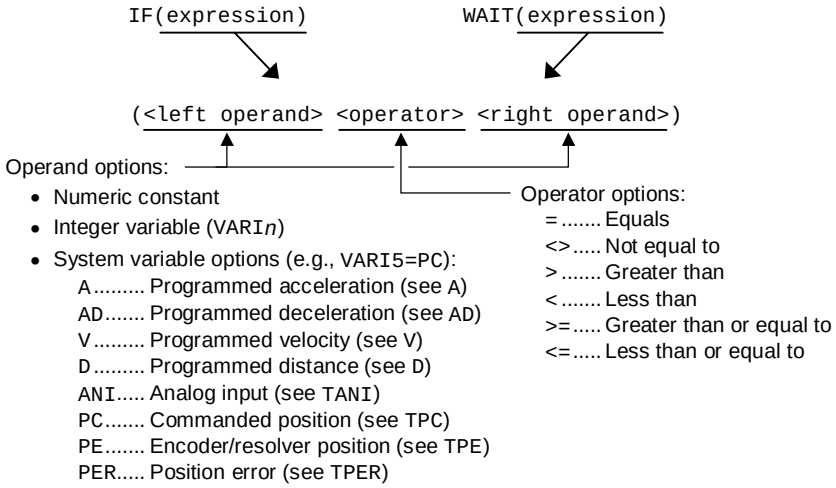
T ..... Time delay (real number, resolution is 0.001)
L ..... Loop
D ..... Distance
A ..... Acceleration (real number, resolution is 0.0001)
AD ..... Deceleration (real number, resolution is 0.0001)
V ..... Velocity (real number, resolution is 0.0001)
REG ..... Registration distance
PSET ..... Establish absolute position

NOTE

The command parameter values for A, AD, V, and T are real numbers. The resolution of A, AD, and V is 0.0001, for T it is 0.001. When substituting VARI variables in these commands, the VARI integer value is applied to the full decimal range. For example, if the value of VARI5 is 136298, the substitution A(VARI5) yields an acceleration value of A13.6298.

Variable Comparisons in Conditional Expressions

Variable comparison allows program flow to be affected by IF and WAIT conditional expressions, based on comparisons between integer variables and system variables or integer values. Relative to variable comparisons, the options for the comparison operands (left and right) and the comparison operator are:



Examples:

IF(VARI5<PE) .....Compares the value of integer variable #5 (VARI5) with the present encoder position (PE). The condition evaluates true when the value of VAR5 is less than the integer value of the encoder position.

WAIT(PE>=16000) .....Command execution pauses until the encoder or resolver position is greater than or equal to 16000.

For details on IF and WAIT conditional statements, refer to the IF command (page 116) and the WAIT command (page 184).

Programming Scenario

This program helps a manufacturer cut web stock a specific length in batches of ten. The Gemini drive controls the web feed motion and it uses one of its outputs to control the saw carriage to cut the stock.

```
DEL PROG1         ; Delete the existing PROG1 program (a precaution)
DEF PROG1         ; Begin definition of setup program
VARI1=10          ; Initialize a batch count variable (VARI1) to 10
VARI2=0           ; Initialize a loop count variable (VARI2) to zero
L                 ; Begin loop (infinite)
 IF(VARI1=VARI2)  ; If the batch is complete, stop the program
  OUTxxxxx1       ; Activate the "batch complete" output
  S               ; Stop
 ELSE             ; If the batch is not complete, cut more stock
 MA0              ; Select incremental positioning mode
 A10              ; Set acceleration and deceleration
 V5               ; Set velocity
 D249888          ; Set distance to increment the stock
 GO1              ; Initiate motion
 WAIT(AS.24=b1)   ; Wait until move complete (within "target zone")
 OUT1             ; Activate output #1 to engage the saw assembly
                  ; to make the cut
 WAIT(IN.5=b1)    ; Wait until the saw assembly returns to its
                  ; ready position (it turns on its "ready" output)
 VARI2=VARI2+1    ; Increment the loop counter
 NIF              ; End of IF code
LN                ; End of loop
END
```

Error Handling

The Gemini drive offers error-handling features that enable you to provide programmed responses to certain error conditions 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.

These are the primary elements of implementing error handling:

- **Define an "error program"**. This is a program that you wish to be invoked when an error condition exists. The program is defined like any other program (with `DEF PROGn` and `END`), and it is designated as the "error program" with the `ERRORP` command. For example, the `ERRORP PROG9` command assigns program #9 (an existing program stored in memory) as the error program. An example error program is provided below.
- **Tell the Gemini drive to check for certain error conditions**. Use the `ERROR` command to enable specific error-checking bits. For example, the `ERRORx1x1` command tells the Gemini drive to start checking After an error-checking bit is enabled, the Gemini drive will branch to the assigned error program when the condition occurs. To understand the type of branch and how to recover from the error condition, refer to the `ERRORP` command description (page 107).

  Error Conditions (see `ERROR` and `ERRORP` for details):

  Error Bit #1 ..... Stall detected (GT6 only).
  Error Bit #2 ..... Hardware end-of-travel limit hit.
  Error Bit #3 ..... Software end-of-travel limit hit.
  Error Bit #4 ..... Drive Fault detected.
  Error Bit #5 ..... Commanded stop or kill (`S`, `!S`, `K`, or `!K`).
  Error Bit #6 ..... Kill input (`INFNCi-C`) activated.
  Error Bit #7 ..... User fault input (`INFNCi-F`) activated.
  Error Bit #8 ..... Stop input (`INFNCi-D`) activated.
  Error Bit #9 ..... Enable input not grounded.
  Error Bit #10 ..... OTF or registration move not possible.
  Error Bit #11 ..... Target zone timeout (GV6 only).
  Error Bit #12 ..... Exceeded maximum allowable position error (GV6 only).

NOTE: Regardless of the error checking, the errors are still reported with the `TER` command, and can be used in a conditional statement. For example, `IF(ER.4=b1)` evaluates true if a drive fault (reported with `TER` bit #4) occurs.

Sample setup code and program:

```
DEF PROG8      ; Define program #8
               ; (which will be the error program)
IF(ER=bXX1)    ; If the error is caused by the occurrence of
               ; a software end-of-travel limit (bit #3 set
               ; to one), back off the software limit, reset
               ; the position, & continue.
D~             ; Change direction in preparation to back off
               ; the soft limit
D1             ; Set distance to 1 count (just far enough to
               ; back off the soft limit)
GO1            ; Initiate the 1-count move to back off the
               ; soft limit
PSET0          ; Reset the position to zero
NIF            ; End IF statement
END            ; End definition of program #8

ERRORP PROG8   ; Set program #8 to be the error program.
               ; Branch to program #8 upon encountering a
               ; software end-of-travel limit.

ERRORXX1       ; Set error condition bit #3 to check for
               ; a software end-of-travel limit
```

Basic Operation Setup

Use the Setup Wizard

Using a Setup Program:

The resulting code from the setup wizard, plus other application-specific commands as deemed necessary, is typically placed in a "setup program". The purpose of the setup program is to place the Gemini drive in a ready state for subsequent motion control. The setup program is usually executed on power up or reset (assigned as the start-up program – see STARTP on page 155), or it is executed from the primary or "main" program which controls the overall application.

The process for creating a setup program is presented on page 22.

For many setup parameters (see list below), you should use the setup wizard in Motion Planner (see page 6) or the configuration wizard in Pocket Motion Planner (see page 11). In each wizard, you have the option of performing an "express" setup (just enough to get up and running) or a "full" detailed setup.

If you need help understanding the items you are configuring, refer to the relevant command description.

Motor Setup:

Selection:

If you are using a Parker motor, select the motor series/frame size/part number from the pull-down lists. The wizard automatically fills in all of the motor parameters and many other drive/system parameters (denoted by **), based on the selected Parker motor.

If you are not using a Parker motor, or you are using a custom motor that is not listed, you will have to fill in all motor parameters. **CAUTION**: It is mandatory that all of these parameters be configured to avoid a motor configuration error, which prevents you from enabling the drive.

Parameters: (not necessary if using Parker motor)

Parameter	GT	GV	GT6	GV6	Command
** Rated Speed	–	X	–	X	DMTW
** Number of Pole Pairs	X	X	X	X	DPOLE
** Electrical Pitch (linear motors)	–	X	–	X	DMEPIT
** Rotor Inertia; Forcer Mass	X	X	X	X	DMTJ
** Damping	–	X	–	X	DMTD
** Thermal Time Constant	–	X	–	X	DMTTCM
** Winding Thermal Resistance	–	X	–	X	DMTRWC
Motor Ambient Temperature	–	X	–	X	DMTAMB
** Max. Motor Winding Temp.	–	X	–	X	DMTMAX
** Voltage Constant (Ke)	–	X	–	X	DMTKE
** Continuous Current	X	X	X	X	DMTIC
** Continuous Current Derating	–	X	–	X	DMTICD
** Peak Current	–	X	–	X	DMTIP
** Winding Time Constant	–	X	–	X	DMTTCW
** Winding Resistance	X	X	X	X	DMTRES
** Minimum Inductance	–	X	–	X	DMTLMN
** Maximum Inductance	–	X	–	X	DMTLMX
** Static Torque	X	–	X	–	DMTSTT
** Inductance	X	–	X	–	DMTIND
** Current Loop Gains	X	–	X	–	DIGNA/B/C/D

Drive, Load and Fault Settings:

Drive:

Drive Control Mode	X	X	X	X	DMODE
** Drive Resolution	X	X	X	–	DRES
Drive PWM Frequency	–	X	–	X	DPWM
** Feedback Source Selection	–	X	–	X	SFB
** Encoder Resolution	–	X	–	X	ERES
** Encoder Output Resolution	–	X	–	X	ORES
** Step & Dir. Output Resolution	X		X	–	ORES
** Linear Motor Electrical Pitch	–	X	–	X	DMEPIT
** Auto Current Standby	X		X	–	DAUTOS
** Torque Limit	–	X	–	X	DMTLIM
** Torque Scaling	–	X	–	X	DMTSCL
** Velocity Limit	X	X	X	X	DMVLIM
** Velocity Scaling	X	X	–	–	DMVSCL

(Continued)

Load:

Parameter	GT	GV	GT6	GV6	Command
Load/Rotor Inertia or Mass Ratio	X	X	X	X	LJRAT
Load Damping	–	X	–	X	LDAMP

Fault:

Fault on Startup Indexer Pulses	X	X	–	–	FLTSTP
Fault on Drive Disable	X	X	X	X	FLTDSB
Fault on Stall Detect	X	–	X	–	ESK
Stall Detect Sensitivity	X	–	X	–	DSTALL
Current Foldback	–	X	–	X	DIFOLD
Disable Drive on Kill	–		X	X	KDRIVE
Max Allowable Position Error	–	X	–	X	SMPER
Max Allowable Velocity Error	–	X	–	X	SMVER

I/O Setup (on DRIVE I/O connector):

Digital Inputs:

Input Function Assignment	–	–	X	X	INFNC
Active Level	X	X	X	X	INLVL
Enable End-of-Travel Limits	X	X	X	X	LH
End-of-Travel Limit Decel	–	–	X	X	LHAD
End-of-Travel Limit S-curve Decel	–	–	X	X	LHADA
Input Debounce	X	X	X	X	INDEB

Digital Outputs:

Output Function Assignment	–	–	X	X	OUTFNC
Active Level	X	X	X	X	OUTLVL

Analog Monitor Outputs:

Variables for Monitor A	X	X	X	X	DMONAV
Scaling for Monitor A	X	X	X	X	DMONAS
Variables for Monitor B	X	X	X	X	DMONBV
Scaling for Monitor B	X	X	X	X	DMONBS

Stepper Motor Matching and Damping:

Motor Matching:

Phase A Offset	X	–	X	–	DPHOFA
Phase B Offset	X	–	X	–	DPHOFB
Phase Balance	X	–	X	–	DPHBAL
Waveform	X	–	X	–	DWAVEF

Electronic Damping:

Active Damping Gain	X	–	X	–	DACTDP
Electronic Viscosity Gain	X	–	X	–	DELVIS
Damping During Acceleration	X	–	X	–	DDAMPA
ABS Damping	X	–	X	–	DABSD

Servo Tuning Gains:

Primary Gains:

** Current Loop Bandwidth	–	X	–	X	DIBW
** Velocity Loop Bandwidth	–	X	–	X	DVBW
** Position Loop Bandwidth	–	X	–	X	DPBW

Advanced Gains:

** Current Loop Bandwidth	–	X	–	X	DIBW
** Velocity Loop Bandwidth	–	X	–	X	DVBW
** Position Loop Bandwidth	–	X	–	X	DPBW
Velocity/Position Bandwidth Ratio	–	X	–	X	SGPSIG
Current Damping Ratio	–	X	–	X	SGIRAT
Velocity Damping Ratio	–	X	–	X	SGVRAT
Position Damping Ratio	–	X	–	X	SGPRAT
Notch Filter A Frequency	–	X	–	X	DNOTAF
Notch Filter A Quality Factor	–	X	–	X	DNOTAQ
Notch Filter A Depth	–	X	–	X	DNOTAD
Notch Filter B Frequency	–	X	–	X	DNOTBF
Notch Filter B Quality Factor	–	X	–	X	DNOTBQ
Notch Filter B Depth	–	X	–	X	DNOTBD
Lead Filter Break Frequency	–	X	–	X	DNOTLD
Lag Filter Break Frequency	–	X	–	X	DNOTLG
Integrator, Enable/Disable	–	X	–	X	SGINTE

Resetting the Gemini Drive

There are different ways to reset the Gemini drive, depending on what method and outcome you desire:

- Cycle power, execute the RESET command, or activate the hardware Reset input (pin 3 on the DRIVE I/O connector).

  What is saved in the Gemini drive's memory?
  - Existing programs (DEF PROG) and profiles (DEF PROF). NOTE: One of the programs can be assigned as the "Startup Program" (see STARTP command), which is automatically executed on power up or reset.
  - Some data-related commands, *but only if they are executed outside of a program*; these commands are denoted with "⊠" in the list on page 187. For those commands that are not automatically saved in EEPROM, they may be executed on power up or reset by placing them in a program (DEF PROG) and assigning the program as the "Startup Program" (see STARTP command).

- Execute the RFS command. This effectively returns the Gemini drive to factory default conditions (with the exception of the ERROK prompt and the TDHRS value). NOTE: All stored programs (DEF PROG) and profiles (DEF PROF) are deleted; therefore, you should make sure you save backup copies of original program/profile files on your hard drive (this is demonstrated in the programming scenario on page 20).

End-of-Travel Limits

The Gemini drive can respond to both hardware and software end-of-travel limits. The purpose of hardware and software end-of-travel limits is to prevent the motor's load from traveling past defined limits. Software and hardware limits are typically positioned in such a way that when the software limit is reached, the motor/load will start to decelerate toward the hardware limit, thus allowing for a much smoother stop at the hardware limit. Software limits can be used regardless of incremental or absolute positioning (MA). When a hardware or software end-of-travel limit is reached, the Gemini drive stops motion using the hardware deceleration rate (set with LHAD & LHADA) or software limit deceleration rate (set with LSAD & LSADA).

Related Commands:

LH.....Hardware end-of-travel limit enable
LHAD .....Hardware end-of-travel limit deceleration
LHADA .....Hardware end-of-travel limit deceleration (s-curve)
INFNC .....Limit input function assignments
INLVL .....Limit switch polarity
LS.....Software end-of-travel limit enable
LSAD .....Software end-of-travel limit deceleration
LSADA .....Software end-of-travel limit deceleration (s-curve)
LSNEG .....Software end-of-travel limit (negative direction)
LSPOS .....Software end-of-travel limit (positive direction)
TIN.....Limit input hardware status (factory default: see bit #1 and #2)
TAS.....Bit #15 indicates if a positive direction hardware limit was encountered.
Bit #16 indicates if a negative direction hardware limit was encountered.
Bit #17 indicates if a positive direction software limit was encountered.
Bit #18 indicates if a negative direction software limit was encountered.
TER.....Bit #2 indicates if a hardware end-of-travel limit was encountered;
Bit #3 indicates if a software end-of-travel limit was encountered
ERROR .....If bit #2 or #3 is enabled, the Gemini drive will branch to the ERRORP program if a hardware or software end-of-travel limit is encountered.

How to set up hardware end-of-travel limits:

1. Connect the end-of-travel limit inputs according to the instructions in your *Hardware Installation Guide*. To help assure safety, connect normally-closed switches and leave the active level at default "active low" setting (set with the `INLVL` command).
2. (Optional) Use the `INFNC` command (page 118) to define the inputs to be used as end-of-travel inputs for the respective axes. **NOTE**: When the Gemini drive is shipped from the factory, inputs #1 and #2 are configured with the `INFNC` command to function as the positive and negative end-of-travel limit inputs, respectively.
3. Set the hard limit deceleration rate (`LHAD` & `LHADA`) to be used when the limit switch is activated. The `LHADA` command allows you to define an s-curve deceleration (see page 53 for details on s-curve profiling).

NOTES ON HARDWARE LIMITS

- The Gemini drive is shipped from the factory with the hardware end-of-travel limits enabled (`LH3`), but not connected. Therefore, motion will not be allowed until you do one of the following:
  - Install limit switches or jumper the end-of-travel limit terminals to ground (*refer to your product's Installation Guide for wiring instructions*).
  - Disable the limits with the `LH0` command (only if the load is not coupled).
  - Reverse the active level of the limits by executing the `INLVL00` command.
- If a hardware limit is encountered while limits are enabled (`LH3`), motion must occur in the opposite direction before a move in the original direction is allowed.

How to set up software end-of-travel limits:

1. Use the `LS3` command to enable the software end-of-travel limits.
2. Define the positive-direction limit with the `LSPOS` command, and define the negative-direction limit with the `LSNEG` command.

   The `LSPOS` and `LSNEG` positions are specified as absolute positions, relative to the absolute zero position. The absolute zero position is established upon power-up and after a successful homing operation, and can be reset using the `PSET` command. Be sure to set the `LSPOS` value greater than the `LSNEG` value.

   For example, `LSPOS+80000` establishes the positive-direction limit at absolute position +80,000 and `LSNEG-60000` establishes the negative-direction limit at absolute position -60,000.

NOTES ON SOFTWARE LIMITS

- If a soft limit is encountered while limits are enabled (`LS3`), motion must occur in the opposite direction before a move in the original direction is allowed.
- To ensure proper motion when using software end-of-travel limits, be sure to set the `LSPOS` value to an absolute value greater than the `LSNEG` value.

Homing

The *homing operation* is a sequence of moves that position an axis using the Home Limit input and/or the Z Channel input of an incremental encoder. The goal of the homing operation is to return the load to a repeatable initial starting location.

Zero Reference After Homing: As soon as the homing operation is successfully completed, the absolute position register is reset to zero, thus establishing a zero reference position.

The homing operation has several potential homing functions you can customize to suit the needs of your application (illustrations of the effects of these commands are presented below):

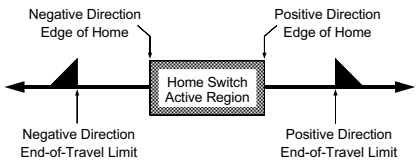
Command	Homing Function	Default
HOM	Initiate the homing move. To start the homing move in the positive direction, use HOMØ; to home in the negative direction, use HOM1.	(do not home)
HOMA	Acceleration and deceleration while homing.	HOMA1Ø (10 revs/sec^2)
HOMBAC	Back up to home. The load will decelerate to a stop after encountering the active edge of the home region, and then will move in the opposite direction at the HOMVF velocity until the active edge of the home region is encountered. Allows the use of HOMEDG and HOMDF.	HOMBAC0 (function disabled)
HOMDF	Final approach direction – during backup to home (HOMBAC) or during homing to the Z channel input of an incremental encoder (HOMZ).	HOMDF0 (positive direction)
HOMEDG	Specify the side of the home switch on which to stop (either the positive-travel side or the negative-travel side).	HOMEGD0 (positive side)
INLVL	Define the home limit input active level (i.e., the state, high or low, which is to be considered an activation of the input). The factory default setting (INLVL110) requires a normally-open switch.	INLVL110 (active-low, use a normally-open switch)
HOMV	Velocity while seeking the home position (see also HOMVF).	HOMV1 (1 rev/sec)
HOMVF	Velocity while in final approach to home position—during backup to home (HOMBAC) or during homing to the Z channel input of an incremental encoder (HOMZ).	HOMVF.1 (0.1 revs/sec)
HOMZ	(GV6 only) Home to the Z channel input from an incremental encoder. NOTE: The home limit input must be active prior to homing to the Z channel.	HOMZ0 (function disabled)

Homing Status:

Status of homing moves is stored in bit #5 of the axis status register (indicates whether or not the home operation was successful). To display the status, use the TAS command. To use the status in a conditional expression (e.g., for an IF statement), use the AS assignment/comparison operator.

NOTES ABOUT HOMING

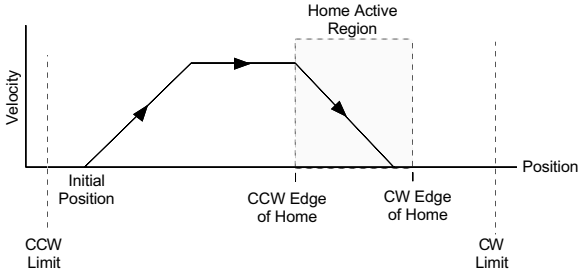
- Avoid using pause and resume functions during the homing operation. A pause command (PS or !PS) or pause/resume input (input configured with the INFNCi-E command) will pause the homing motion. However, when the subsequent resume command (C or !C) or pause/resume input occurs, motion will resume at the beginning of the homing motion sequence.
- "Positive" vs. "negative" direction:



- If an end-of-travel limit is encountered during the homing operation, the motion will be reversed and the home switch will be sought in the opposite direction. If a second limit is encountered, the homing operation will be terminated, stopping motion at the second limit.

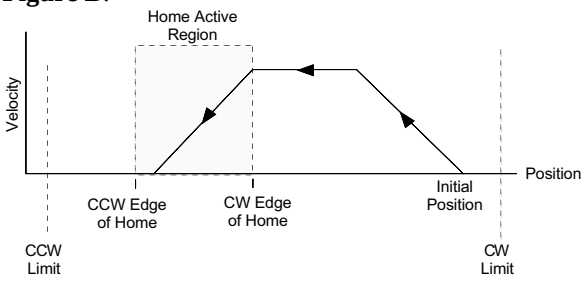
Figures A and B show the homing operation when HOMBAC is not enabled. *"CW" refers to the positive direction and "CCW" refers to the negative direction.*

Figure A:



Home Profile Attributes:
- Start home move in positive direction (HOM0)
- Backup To Home disabled (HOMBAC0)

Figure B:



Home Profile Attributes:
- Start home move in negative direction (HOM1)
- Backup To Home disabled (HOMBAC0)

Positive Homing, Backup to Home Enabled

The seven steps below describe a sample homing operation when HOMBAC is enabled (see Figure C). The final approach direction (HOMDF) is CW and the home edge (HOMEDG) is the CW edge. *"CW" refers to the positive direction and "CCW" refers to the negative direction.*

NOTE

To better illustrate the direction changes in the backup-to-home operation, the illustrations in the remainder of this section show the backup-to-home movements with varied velocities. In reality, the backup-to-home movements are performed at the same velocity (HOMVF value).

Step 1 A CW home move is started with the HOMØ command at the HOMA acceleration. Default HOMA is 10 counts/sec/sec.

Step 2 The HOMV velocity is reached (move continues at that velocity until home input goes active).

Step 3 The CCW edge of the home input is detected, this means the home input is active. At this time the move is decelerated at the HOMA command value. It does not matter if the home input becomes inactive during this deceleration.

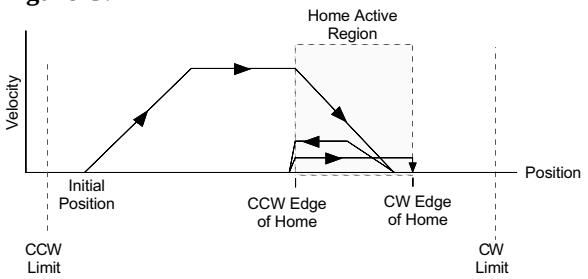
Step 4 After stopping, the direction is reversed and a second move with a peak velocity specified by the HOMVF value is started.

Step 5 This move continues until the CCW edge of the home input is reached.

Step 6 Upon reaching the CCW edge, the move is decelerated at the HOMA command value, the direction is reversed, and another move is started in the CW direction at the HOMVF velocity.

Step 7 As soon as the home input CW edge is reached, this last move is immediately terminated. The load is at home and the absolute position register is reset to zero.

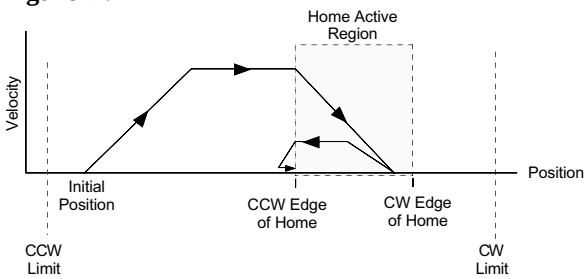
Figure C:



Home Profile Attributes:
- Start home move in positive direction (HOM0)
- Backup To Home enabled (HOMBAC1)
- Final approach direction is positive (HOMDF0)
- Stop on the positive-travel side of the home switch active region (HOMEDG0)

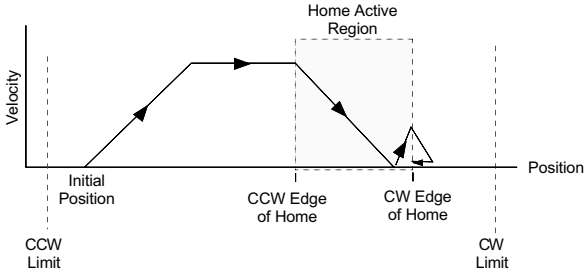
Figures D through F show the homing operation for different values of HOMDF and HOMEDG, when HOMBAC is enabled. *"CW" refers to the positive direction and "CCW" refers to the negative direction.*

Figure D:



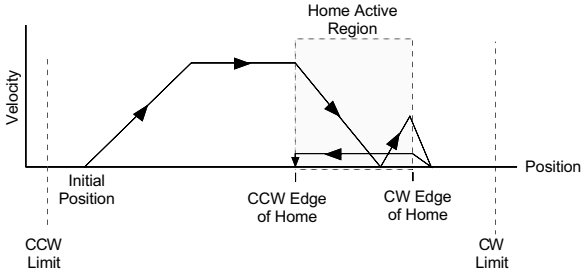
Home Profile Attributes:
- Start home move in positive direction (HOM0)
- Backup To Home enabled (HOMBAC1)
- Final approach direction is positive (HOMDF0)
- Stop on the negative-travel side of the home switch active region (HOMEDG1)

Figure E:



Home Profile Attributes:
- Start home move in positive direction (HOM0)
- Backup To Home enabled (HOMBAC1)
- Final approach direction is negative (HOMDF1)
- Stop on the positive-travel side of the home switch active region (HOMEDG0)

Figure F:

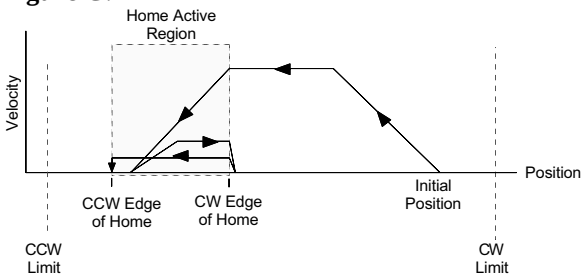


Home Profile Attributes:
- Start home move in positive direction (HOM0)
- Backup To Home enabled (HOMBAC1)
- Final approach direction is negative (HOMDF1)
- Stop on the negative-travel side of the home switch active region (HOMEDG1)

Negative Homing, Backup to Home Enabled

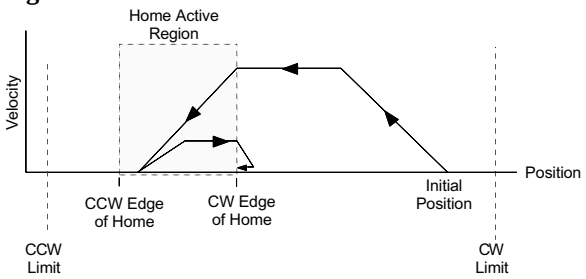
Figures G through J show the homing operation for different values of HOMDF and HOMEDG, when HOMBAC is enabled. *"CW" refers to the positive direction and "CCW" refers to the negative direction.*

Figure G:



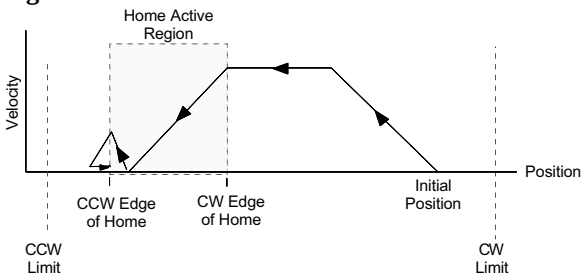
Home Profile Attributes:
- Start home move in negative direction (HOM1)
- Backup To Home enabled (HOMBAC1)
- Final approach direction is negative (HOMDF1)
- Stop on the negative-travel side of the home switch active region (HOMEDG1)

Figure H:



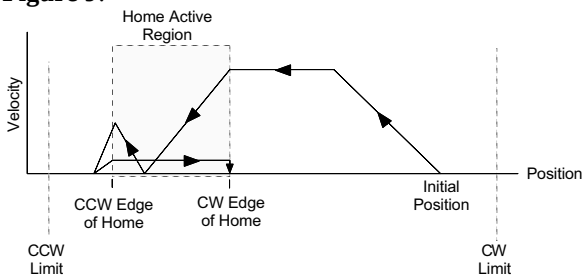
Home Profile Attributes:
- Start home move in negative direction (HOM1)
- Backup To Home enabled (HOMBAC1)
- Final approach direction is negative (HOMDF1)
- Stop on the positive-travel side of the home switch active region (HOMEDG0)

Figure I:



Home Profile Attributes:
- Start home move in negative direction (HOM1)
- Backup To Home enabled (HOMBAC1)
- Final approach direction is positive (HOMDF0)
- Stop on the negative-travel side of the home switch active region (HOMEDG1)

Figure J:

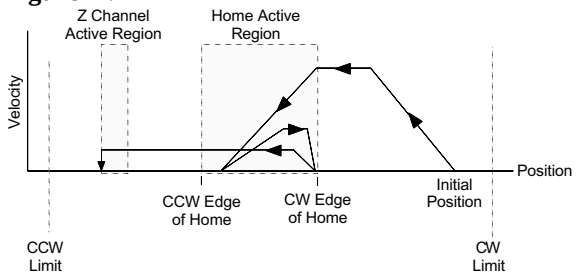


Home Profile Attributes:
- Start home move in negative direction (HOM1)
- Backup To Home enabled (HOMBAC1)
- Final approach direction is positive (HOMDF0)
- Stop on the positive-travel side of the home switch active region (HOMEDG0)

Homing Using The Z-Channel (GV6 only)

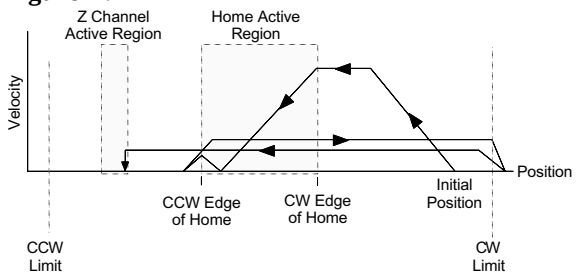
Figures K through O show the homing operation when homing to an encoder index pulse, or Z channel, is enabled (HOMZ1). The Z-channel will only be recognized after the home input is activated. It is desirable to position the Z channel within the home active region; this reduces the time required to search for the Z channel. *"CW" refers to the positive direction and "CCW" refers to the negative direction.*

Figure K:



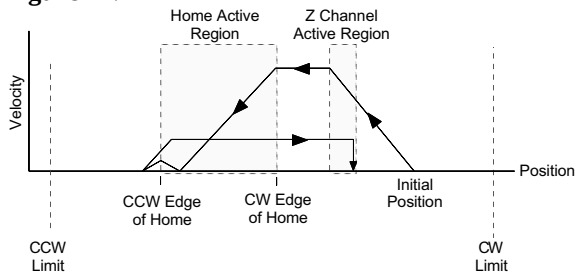
Home Profile Attributes:
- Z-Channel homing enabled (HOMZ1)
- Start home move in negative direction (HOM1)
- Backup To Home enabled (HOMBAC1)
- Final approach direction is negative (HOMDF1)
- Stop on the negative-travel side of the z-channel active region (HOMEDG1)

Figure L:



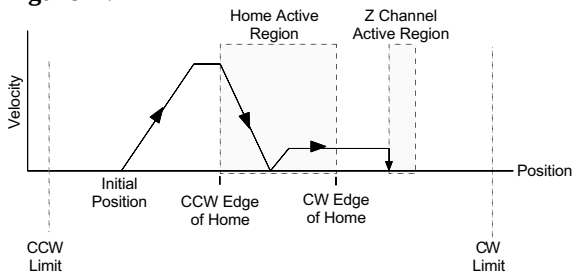
Home Profile Attributes:
- Z-Channel homing enabled (HOMZ1)
- Start home move in negative direction (HOM1)
- Backup To Home enabled (HOMBAC1)
- Final approach direction is positive (HOMDF0)
- Stop on the positive-travel side of the z-channel active region (HOMEDG0)

Figure M:



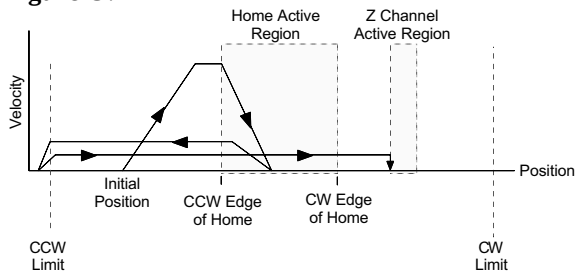
Home Profile Attributes:
- Z-Channel homing enabled (HOMZ1)
- Start home move in negative direction (HOM1)
- Backup To Home enabled (HOMBAC1)
- Final approach direction is positive (HOMDF0)
- Stop on the positive-travel side of the z-channel active region (HOMEDG0)

Figure N:



Home Profile Attributes:
- Z-Channel homing enabled (HOMZ1)
- Start home move in positive direction (HOM0)
- Backup To Home disabled (HOMBAC0)
- Final approach direction is positive (HOMDF0)
- Stop on the positive-travel side of the z-channel active region (HOMEDG0)

Figure O:



Home Profile Attributes:
- Z-Channel homing enabled (HOMZ1)
- Start home move in positive direction (HOM0)
- Backup To Home enabled (HOMBAC1)
- Final approach direction is positive (HOMDF0)
- Stop on the positive-travel side of the z-channel active region (HOMEDG0)

Servo Tuning (GV6 only)

The goal of the tuning process is to identify the gain settings (see command list below) required to achieve optimum servo performance in your application. The tuning commands are typically placed into a setup program (see page 22).

Follow the tuning procedures provided in the *GV6 Hardware Installation Guide*. The tuning parameters that you send to the Gemini drive during the tuning procedure will be remembered (stored in the Gemini drive's EEPROM memory); however, for safe-keeping, you should also place these commands in a program file (see page 20) that you can later download to the Gemini drive in the event that the tuning values are changed or the `RFS` command is executed.

Tuning Related Commands

Below is a list of the tuning related commands. All of these commands, collectively, are regarded as a "gain set". You have the option of invoking different gain sets, to accommodate the dynamics of your servo system (see below).

`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
`DVBW` ..... Velocity 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

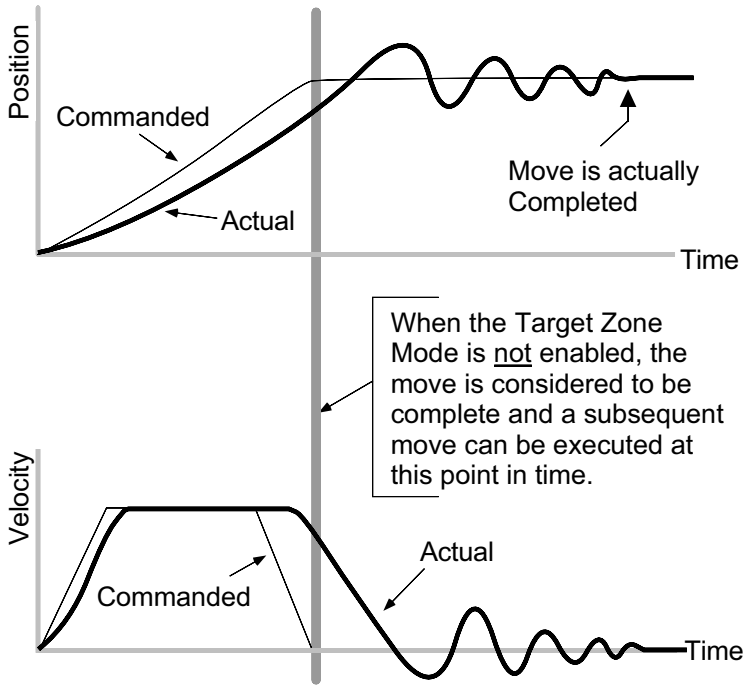
Using Gain Sets

The `SGSET` command saves the presently active gain values (see list above) 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.

Target Zone (GV6 only)

Under default operation (Target Zone Mode not enabled), the Gemini drive's move completion criteria is simply derived from the move trajectory. The Gemini considers the current preset move to be complete when the commanded trajectory has reached the desired target position; after that, subsequent commands/ moves can be executed. Consequently, the next move or external operation can begin before the actual position has settled to the commanded position (see diagram below).



To prevent premature command execution before the actual position settles into the commanded position, use the *Target Zone Mode*. In this mode, enabled with the `STRGTE1` command, the move cannot be considered complete until the actual position and actual velocity are within the *target zone* (that is, within the distance zone defined by `STRGTD` and less than or equal to the velocity defined by `STRGTV`). If the load does not settle into the target zone before the timeout period set with the `STRGTT` command, the Gemini detects a *timeout error* (see illustration below).

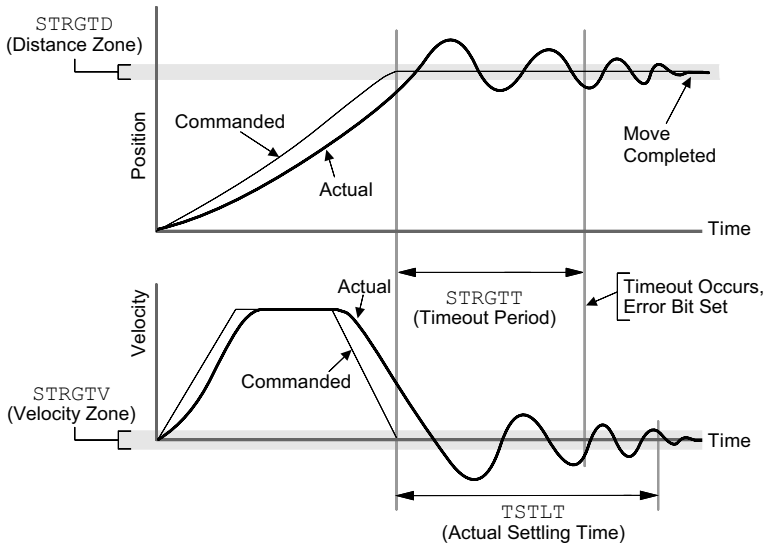
If the timeout error occurs, you can prevent subsequent command/move execution only if you enable the `ERROR` command to continually check for this error condition, and when it occurs to branch to a programmed response you can define in the `ERRORP` program.

As an example, setting the distance zone to ±5 counts (`STRGTD5`), the velocity zone to ≤0.5 revs/sec (`STRGTV0.5`), and the timeout period to 1/2 second (`STRGTT500`), a move with a distance of 8,000 counts (`D8000`) must end up between position 7,995 and 8,005 and settle down to ≤0.5 rps within 500 ms (1/2 second) after the commanded profile is complete.

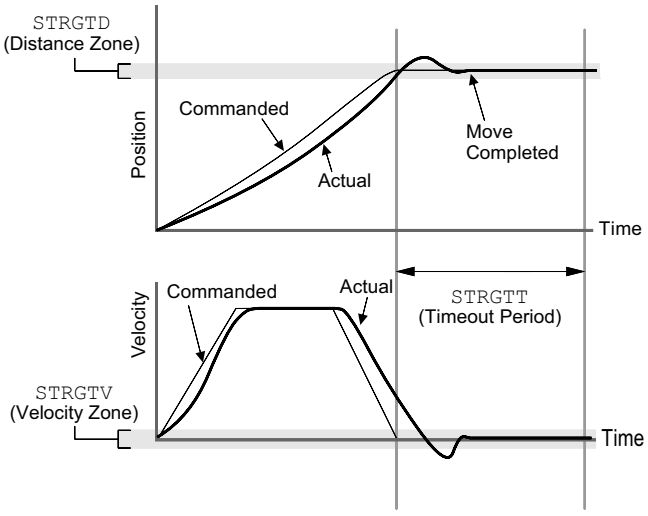
Damping is Critical

To ensure that a move settles within the distance zone, it must be damped to the point that it will not move out of the zone in an oscillatory manner. This helps ensure the actual velocity falls within the target velocity zone set with the STRGTV command (see illustration below).

Failed Move Completion



Successful Move Completion



Checking the Settling Time

Using the TSTLT command, you can 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.

The TSTLT command is usable whether or not the Target Zone Settling Mode is enabled with the STRGTE command.

Programmable Inputs and Outputs

Programmable inputs and outputs allow the Gemini drive to detect and respond to the machine processes and program conditions (e.g., state of switches, thumbwheels, electronic sensors, and outputs of other equipment such as PLCs).

What to know about programmable inputs and outputs:

- Each input can be assigned a function that provides a programmed response to external conditions. See INFNC (page 118) for details on each function.

  INFNCi-A .....General-purpose
  INFNCi-B .....BCD program selection
  INFNCi-C .....Kill
  INFNCi-D .....Stop
  INFNCi-E .....Pause/Continue
  INFNCi-F .....User fault
  INFNCi-H .....Trigger interrupt (for registration or TRGFN functions)
  INFNCi-R .....End-of-travel limit, positive direction
  INFNCi-S .....End-of-travel limit, negative direction
  INFNCi-T .....Home limit

- Each output can be assigned a function that provides the ability to affect external conditions. See OUTFNC (page 134) for details on each function.

  OUTFNCi-A .....General-purpose (can be activated with OUT and POUTA)
  OUTFNCi-B .....Moving/not moving indicator
  OUTFNCi-C .....Program in progress indicator
  OUTFNCi-D .....End-of-travel limit encountered indicator
  OUTFNCi-E .....Stall indicator (GT6)
  OUTFNCi-F .....Fault indicator
  OUTFNCi-G .....Position exceeds the max. allowable SMPER limit (GV6)

- The status of each input can be used in a conditional IF statement to control program flow (see IF description on page 116).
- The programmable I/O conditions are updated once per millisecond.

The table below lists each programmable input, its pin number on the DRIVE I/O connector, and its factory default programmed conditions.

Input #	Pin #	Function Assignment (INFNC)	Debounce (INDEB)	Active Level (INLVL)	Status Bit (TIN)
1	28	INFNC1-R (Positive end-of-travel limit)	none	Active high	1
2	29	INFNC2-S (Negative end-of-travel limit)	none	Active high	2
3	31	INFNC3-T (Home limit)	none	Active low	3
4	34	INFNC4-H (Trigger interrupt)	50 ms	Active low	4
5	35	INFNC5-A (General-purpose)	50 ms	Active low	5
6	37	INFNC6-A (General-purpose)	50 ms	Active low	6
7	38	INFNC7-A (General-purpose)	50 ms	Active low	7
8	39	INFNC8-A (General-purpose)	50 ms	Active low	8

The table below lists each programmable output, its pin number on the DRIVE I/O connector (except the Relay), and its factory default programmed conditions.

Output #	Pin #	Function Assignment (OUTFNC)	Active Level (OUTLVL)	Status Bit (TOUT)
1	41	OUTFNC1-A (General-purpose; control with OUT, POUTA)	Active low	1
2	43	OUTFNC2-F (Fault)	Active low	2
3	45	OUTFNC3-D (End-of-travel limit encountered)	Active low	3
4	46	OUTFNC4-E (Stall – GT6) or OUTFNC4-G (Exceeded SMPER limit – GV6)	Active low	4
5	48	OUTFNC5-B (Moving/not moving)	Active low	5
6	49	OUTFNC6-A (General-purpose)	Active low	6
7	Relay	OUTFNC8-F (Fault)	Active low	7

Communication

The Gemini drive has several serial communication setup parameters that you can customize for your application. The list below identifies each parameter and the factory default condition. To explore optional settings, refer to the respective command description later in this manual. Refer to your drive's *Hardware Installation Guide* for RS-232 and RS-485 connection instructions.

<table>
<tr><th>Command</th><th>Description</th><th>Default Setting</th></tr>
<tr><td>E</td><td>Enables/disables serial communication.</td><td>E1 (enabled)</td></tr>
<tr><td>ECHO</td><td>Enables/disables command echo.
If using an RS-232 daisy-chain, use ECHO1.
If using an RS-485 multi-drop, use ECHO0.</td><td>ECHO1 (enabled)</td></tr>
<tr><td>ERRLVL *</td><td>Determines which characters are displayed as part of the response from the Gemini drive.

Characters transmitted when an error is not detected:
<table>
<tr><th></th><th>BOT/EOT/EOL</th><th>*</th><th>ERROK</th></tr>
<tr><td>ERRLVL4:</td><td>Yes</td><td>Yes</td><td>Yes</td></tr>
<tr><td>ERRLVL3:</td><td>Yes</td><td>Yes</td><td>Yes</td></tr>
<tr><td>ERRLVL2:</td><td>Yes</td><td>Yes</td><td>No</td></tr>
<tr><td>ERRLVL0:</td><td>Yes</td><td>No</td><td>No</td></tr>
</table>
Characters transmitted when an error is detected:
<table>
<tr><th></th><th>BOT/EOT/EOL</th><th>ERRBAD</th><th>Err Msg.</th></tr>
<tr><td>ERRLVL4:</td><td>Yes</td><td>Yes</td><td>Yes</td></tr>
<tr><td>ERRLVL3:</td><td>No</td><td>Yes</td><td>No</td></tr>
<tr><td>ERRLVL2:</td><td>No</td><td>No</td><td>No</td></tr>
<tr><td>ERRLVL0:</td><td>No</td><td>No</td><td>No</td></tr>
</table>
Recommendation for RS-485 multi-drop: Use ERRLVL2 to avoid having to address each command to the specific unit in the multi-drop. Be aware, however, that when you use ERRLVL2 there will be no ERROK or ERRBAD prompts and no error messages.</td><td>ERRLVL4</td></tr>
<tr><td>BOT *</td><td>Beginning-of-transmission ASCII characters, placed at the beginning of every response from the Gemini drive. (an ASCII table is provided on page 193)</td><td>BOT0,0,0
(no characters sent)</td></tr>
<tr><td>EOT *</td><td>End-of-transmission ASCII characters, placed at the end of every response from the Gemini drive.</td><td>EOT13,0,0
(carriage return only)</td></tr>
<tr><td>EOL *</td><td>End-of-line ASCII characters, placed at the end of each line of a multi-line response. The last line is terminated with the EOT characters.</td><td>EOL13,10,0
(carriage return and line feed)</td></tr>
<tr><td>ERRBAD *</td><td>ASCII characters sent after an erroneous command has been entered.</td><td>ERRBAD13,10,63,32
(carriage return, line feed, "?", and a space)</td></tr>
<tr><td>ERROK *</td><td>ASCII characters sent after a command has been entered correctly.</td><td>ERROK13,10,62,32
(carriage return, line feed, ">", and a space)</td></tr>
<tr><td>ADDR</td><td>Automatically configures unit addresses for a daisy-chain or multi-drop. Refer to the ADDR description (page 59) or to the RS-232 Daisy Chain or RS-485 Multi-Drop configuration procedures in your drive's *Hardware Installation Guide*.</td><td>ADDR0</td></tr>
<tr><td>XONOFF</td><td>Enables/disables XON/OFF ASCII handshaking with the Gemini drive. If you are using a RS-485 multi-drop, disable XON/XOFF handshaking (XONOFF0).</td><td>XONOFF1 (enabled)</td></tr>
</table>

* These commands are intended to be used only during live terminal communication with the drive. Do not download these commands to the drive, or place them in a program.

Motion Programming

Basic Motion Parameters

Accel, Decel, Velocity, Distance

The basic motion profile comprises acceleration, deceleration, velocity, and distance commands. Motion is initiated with the `GO` command. The table below identifies these commands and their units of measure.

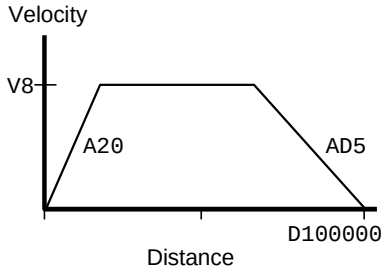
Profile Element	Command	Units *
Acceleration Deceleration	A AD	Revs/sec/sec (rps^2)
Velocity	V	Revs/sec (rps)
Distance	D	Counts

*** NOTE**: If you are using a linear motor, refer to page 44 for instructions on calculating rotary motion parameters to accommodate linear applications.

The following program produces a preset (incremental) 100,000-count move in a nominally trapezoidal profile.

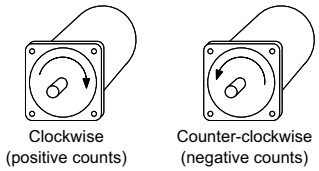
```
DEL PROG6       ; Delete program #6
DEF PROG6       ; Begin definition of program #6
MC0             ; Use the preset positioning mode
MA0             ; Use the incremental (preset) positioning mode
A20             ; Accelerate at 20 revs/sec/sec
AD5             ; Decelerate at 5 revs/sec/sec
V8              ; Set velocity to 8 revs/sec
D100000         ; Set distance to 100,000 counts
GO              ; Execute the move
END             ; End definition of program #6
```

The resulting profile from the above program is:



Direction of Motion for Rotary Motors

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



Positioning Modes

As demonstrated in the program example above, the motion profile is affected by the "positioning mode". There are two main positioning modes:

- Preset Modes (`MC0`): Incremental (`MA0`) or Absolute (`MA1`) – see page 42
- Continuous Mode (`MC1`) – see page 43

Select the mode that is most appropriate for your application. For example, a repetitive cut-to-length application requires incremental positioning. X-Y positioning, on the other hand, is better served in the absolute mode. Continuous mode is useful for applications that require constant movement of the load based on internal conditions or inputs, not distance.

On-The-Fly (Pre-emptive Go) Motion Profiling

While motion is in progress (regardless of the positioning mode), you can change these motion parameters to affect a new profile:

- Acceleration (`A`)
  (s-curve acceleration is not allowed during on-the-fly changes)
- Deceleration (`AD`)
  (s-curve deceleration not allowed during on-the-fly changes)
- Velocity (`V`)
- Distance (`D`)
- Preset or Continuous Positioning Mode Selection (`MC`)
- Incremental or Absolute Positioning Mode Selection (`MA`)

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, see *On-The-Fly Motion Profiling* on page 44.

Preset Positioning Modes

A *preset* move is a point-to-point move of a specified distance. You can select preset moves by putting the Gemini drive into preset mode (canceling continuous mode) using the `MC0` command. Preset moves allow you to position the motor/load in relation to the previous stopped position (*incremental mode*—enabled with the `MA0` command) or in relation to a defined zero reference position (*absolute mode*—enabled with the `MA1` command).

Incremental Mode Moves

The incremental mode is the Gemini's default power-up mode. When using the Incremental Mode (`MA0`), a preset move moves the motor/load the specified distance from its starting position. For example, if you start at position *N*, executing the `D6000` command in the `MA0` mode will move the motor/load 6,000 counts in the positive direction from the *N* position. Executing the `D6000` command again will move the motor/load an additional 6,000 positive counts, ending the move 12,000 counts from position *N*.

You can specify the direction of the move by using the optional sign `+` or `-` (e.g., `D+6000` or `D-6000`). Whenever you do not specify the direction (e.g., `D6000`), the direction defaults to positive (`+`).

Example

```
MC0      ; Select Preset Positioning Mode
MA0      ; Select Incremental (Preset) Positioning Mode
A2       ; Set acceleration to 2 revs/sec/sec
V5       ; Set velocity to 5 revs/sec
D4000    ; Set distance to 4,000 positive counts
GO1      ; Initiate motion (move 4,000 positive counts)
GO1      ; Repeat the move (move 4,000 addition counts)
D-8000   ; Set distance to -8,000 counts (return to original position)
GO1      ; Initiate motion (move 8,000 counts in the negative
         ; direction and end at the original starting position)
```

Absolute Mode Moves

A preset move in the Absolute Mode (MA1) moves the motor/load the distance that you specify from the *absolute zero position*.

> **Establishing a *Zero Position***
>
> One way to establish the zero position is to issue the PSETØ command when the load is at the location you would like to reference as absolute position zero.
>
> The zero position is also established when the Go Home (HOM) command is issued, the absolute position register is automatically set to zero after reaching the home position, thus designating the home position as position zero.

The direction of an absolute preset move depends upon the motor's/load's position at the beginning of the move and the position you command it to move to. For example, if the motor/load is at absolute position +12,500, and you instruct it to move to position +5,000 (e.g., with the D5000 command), it will move in the negative direction a distance of 7,500 steps to reach the absolute position of +5,000.

The Gemini retains the absolute position, even while it is in the incremental mode. To ascertain the absolute position, use the TPC command.

Example

```
MC0      ; Select Preset Positioning Mode
MA1      ; Select Absolute (Preset) Positioning Mode
PSET0    ; Set the present absolute position to zero
A5       ; Set acceleration to 5 revs/sec/sec
V3       ; Set velocity to 3 revs/sec
D4000    ; Set move to absolute position +4,000 counts
GO1      ; Initiate motion (move to absolute position +4,000)
D8000    ; Set move to absolute position +8,000 counts
GO1      ; Move (starting from position +4000, move another 4,000
         ; counts in the positive direction to position +8000)
D0       ; Set move to absolute position zero
GO1      ; Move (starting at absolute position +8,000, move 8,000
         ; counts in the negative direction to position zero)
```

Continuous Positioning Mode

The Continuous Mode (MC1) is useful in these situations:

- Applications that require constant movement of the load
- Synchronize the motor to external events such as trigger input signals
- Changing the motion profile after a specified distance or after a specified time period (T command) has elapsed

You can manipulate the motor movement with either buffered or immediate commands. After you issue the GO command, buffered commands are not executed unless the continuous command execution mode is enabled (COMEXC1). Once COMEXC1 is enabled, buffered commands are executed in the order in which they were programmed (see page 62 for details).

The command can be specified as *immediate* by placing an exclamation mark (!) in front of the command. When a command is specified as immediate, it is placed at the front of the command queue and is executed immediately.

Example A

```
DEL PROG22    ; Delete program #22
DEF PROG22    ; Begin definition of program #22
COMEXC1       ; Enable continuous command processing mode
COMEXS1       ; Allow command execution to continue after stop
MC1           ; Select Continuous Positioning Mode
A10           ; Set acceleration to 10 revs/sec/sec
V1            ; Set velocity to 1 rev/sec
GO1           ; Initiates move (Go)
WAIT(AS.4=b1); Wait to reach commanded velocity
T5            ; Time delay of 5 seconds
S1            ; Stop the move
WAIT(AS.1=b0); Wait for no commanded motion
COMEXC0       ; Disable continuous command processing mode
END           ; End definition of program #22
; When the move is executed, the load will accelerate to 1 rev/sec,
; continue at 1 rps for 5 seconds, and then decelerate to a stop.
```

Example B

```
DEL PROG23    ; Delete program #23
DEF PROG23    ; Begin definition of program #23
COMEXC1       ; Enable continuous command processing mode
COMEXS1       ; Allow command execution to continue after stop
MC1           ; Select Continuous Positioning Mode
A10           ; Set acceleration to 10 revs/sec/sec
V1            ; Set velocity to 1 rev/sec
GO1           ; Initiate move (Go)
WAIT(AS.4=b1); Wait to reach commanded velocity
T3            ; Time delay of 3 seconds
A50           ; Set acceleration to 50
V10           ; Set velocity to 10
GO1           ; Initiate on-the-fly accel and velocity changes
T5            ; Time delay of 5 seconds
S1            ; Stop the move
WAIT(AS.1=b0); Wait for no commanded motion
COMEXC0       ; Disable continuous command processing mode
END           ; End definition of program #23
```

While in continuous mode, typical means to stop motion are:

- You issue an immediate Stop (`!S`) or Kill (`!K`) command.
- A hardware or software end-of-travel limit is encountered.
- The load trips a registration (`INFNCi-H`) input.
- The load or operator activates a kill input (`INFNCi-C`), a stop input (`INFNCi-D`), or a user fault input (`INFNCi-F`).

NOTE

While the axis is moving, you cannot change the parameters of some commands (such as `DRIVE` and `HOM` — see command list on page 62). This rule applies during the `COMEXC1` mode and even if you prefix the command with an immediate command identifier (`!`).

Linear Motion

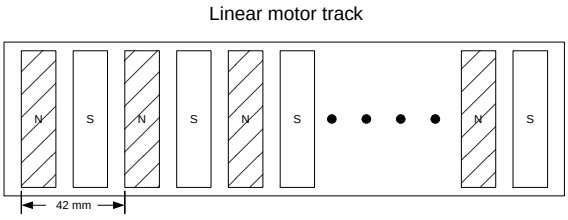
All Gemini drives operate internally in rotary units. If you use a linear motor, you must convert some of the motion parameters to their rotary equivalents:

`A`	Acceleration
`AA`	Acceleration (S-Curve)
`AD`	Deceleration
`ADA`	Deceleration (S-Curve)
`D`	Distance/Position
`HOMA`	Home Acceleration
`HOMV`	Home Velocity
`HOMVF`	Home Final Velocity
`LHAD`	Hardware EOT Limit Decel
`LHADA`	Hardware EOT Limit Decel (S-Curve)
`LSAD`	Software EOT Limit Decel
`LSADA`	Software EOT Limit Decel (S-Curve)
`PSET`	Absolute Position Reference
`REG`	Registration Distance
`REGLOD`	Registration Lockout Distance
`STRGTD`	Target Zone Distance
`STRGTV`	Target Zone Velocity
`V`	Velocity

NOTE: Motor setup is handled differently. If you use the Motion Planner setup wizard (page 6) or the Pocket Motion Planner configuration wizard (page 11), you may enter the motor data in linear units and the software converts them to rotary units before they are downloaded to the drive.

To make the conversion from linear to rotary and vice versa, the motor electrical pitch, or `DMEPIT`, must be known. The electrical pitch relates the linear distance required for the equivalent of one rotary motor revolution. Mechanically, the

definition of the electrical pitch is the linear distance between two magnets comprising a full magnetic cycle. The illustration (below) shows an example of an electrical pitch of 42mm (DMEPIT42). NOTE: Parker linear motors have an electrical pitch of 42mm.



Definition of DMEPIT (Electrical Pitch)

The feedback resolution (ERES) value must be set to the number of counts for the electrical pitch (1 rev).

$$ERES = \underbrace{\frac{\#\,counts}{1\,m}}_{\text{from encoder}} \times \underbrace{\frac{1\,m}{1000\,mm}}_{\text{conversion}} \times \underbrace{\frac{DMEPIT\,(mm)}{1\,(rev)}}_{\text{from motor}} = \frac{\#\,counts}{1\,(rev)}$$

Most linear encoders are metric, and most of those are 1 micron (1 µm):

$$1\ micron\ encoder = \frac{1\ count}{1\ \mu m} = \frac{1{,}000{,}000\ counts}{m}$$

Use the following formulas to convert from linear to rotary units. An sample conversion for position, acceleration, and velocity is provided on page 46.

Linear Position

- D (distance)
- PSET (absolute position reference)
- REG (registration distance)
- REGLOD (registration lockout distance)
- STRGTD (target zone distance)

Metric:

$$Position_{rotary}\,(counts) = \underbrace{\frac{\#\,counts}{1\,(m)}}_{\text{from encoder}} \times Position_{linear}\,(m)$$

English:

$$Position_{rotary}\,(counts) = \underbrace{\frac{\#\,counts}{1\,(m)}}_{\text{from encoder}} \times \frac{.0254\,(m)}{1\,(in)} \times Position_{linear}\,(in)$$

$$Position_{rotary}\,(counts) = \underbrace{\frac{\#\,counts}{1\,(m)}}_{\text{from encoder}} \times \frac{.0254\,(m)}{1\,(in)} \times \frac{12\,(in)}{1\,(ft)} \times Position_{linear}\,(ft)$$

Linear Velocity

- V (velocity)
- HOMV (home velocity)
- HOMVF (home final velocity)
- STRGTV (target velocity)

Metric:

$$Velocity_{rotary}\,(rps) = \frac{1}{DMEPIT\,(mm)} \times \frac{1000\,(mm)}{1\,(m)} \times Velocity_{linear}\,(m/s)$$

English:

$$Velocity_{rotary}\,(rps) = \frac{1}{DMEPIT\,(mm)} \times \frac{25.4\,(mm)}{1\,(in)} \times Velocity_{linear}\,(in/s)$$

$$Velocity_{rotary}\,(rps) = \frac{1}{DMEPIT\,(mm)} \times \frac{25.4\,(mm)}{1\,(in)} \times \frac{12\,(in)}{1\,(ft)} \times Velocity_{linear}\,(ft/s)$$

Linear Acceleration

- A (acceleration)
- AA (s-curve accel)
- AD (deceleration)
- AD (s-curve decel)
- HOMA (home accel)
- LHAD (hard limit decel)
- LHADA (hard limit s-curve decel)
- LSAD (soft limit decel)
- LSADA (soft limit s-curve decel)

Metric:

$$Accel_{rotary}(rpss) = \frac{1}{DMEPIT(mm)} \times \frac{1000\,(mm)}{1\,(m)} \times Accel_{linear}(m/s^2)$$

English:

$$Accel_{rotary}(rpss) = \frac{1}{DMEPIT(mm)} \times \frac{25.4\,(mm)}{1\,(in)} \times Accel_{linear}(in/s^2)$$

$$Accel_{rotary}(rpss) = \frac{1}{DMEPIT(mm)} \times \frac{25.4\,(mm)}{1\,(in)} \times \frac{12\,(in)}{1\,(ft)} \times Accel_{linear}(ft/s^2)$$

Conversion Example

Using a Parker 406-LXR-M-D15-E2 linear motor, the electrical pitch is 42mm (DMEPIT42) and the encoder is 1 micron. Here, we will convert an acceleration of 10 inches/sec^2, a velocity of 5 inches/sec, and a distance of 20 inches:

1. Calculate ERES in counts/rev (result is ERES42000):

$$ERES = \underbrace{\frac{1\,000\,000\ counts}{1\,(m)}}_{\text{from encoder}} \times \underbrace{\frac{1\,(m)}{1000\,(mm)}}_{\text{conversion}} \times \underbrace{\frac{42\,(mm)}{1\,(rev)}}_{\text{from motor}} = \frac{42\,000\ counts}{1\,(rev)}$$

2. Convert an acceleration of 10 inches/sec^2 to its rotary equivalent in revs/sec^2:

$$Accel_{rotary}(rpss) = \frac{1}{42\,(mm)} \times \frac{25.4\,(mm)}{1\,(in)} \times 10\,(in/s^2) = 6.0476\ rpss$$

3. Convert a velocity of 5 inches/sec to its rotary equivalent in revs/sec:

$$Velocity_{rotary}(rps) = \frac{1}{42\,(mm)} \times \frac{25.4\,(mm)}{1\,(in)} \times 5\,(in/s) = 3.0238\ rps$$

4. Convert a distance of 20 inches to its rotary equivalent in counts:

$$Position_{rotary}(counts) = \underbrace{\frac{1\,000\,000\ counts}{1\,(m)}}_{\text{from encoder}} \times \frac{.0254\,(m)}{1\,(in)} \times 20.0\,(in) = 508\,000$$

The program values resulting from these conversions are:

```
ERES42000     ; Set encoder resolution to 42000 counts/rev
A6.0476       ; Set acceleration to 6.0476 revs/sec/sec
              ; (10 inches/sec/sec)
V3.0238       ; Set velocity to 3.0238 revs/sec (5 inches/sec)
D508000       ; Set distance to 508,000 counts (20 inches)
```

On-The-Fly Motion Profiling

While motion is in progress, you can change these motion parameters to affect a new profile:

- Acceleration (A) — s-curve acceleration is not allowed
- Deceleration (AD) — s-curve deceleration is not allowed
- Velocity (V)
- Distance (D)
- Preset or Continuous Positioning Mode Selection (MC)
- Incremental or Absolute Positioning Mode Selection (MA)

The motion parameters can be changed by sending the respective command (e.g., A, V, D, MC) followed by the GO command. If the continuous command execution mode is enabled (COMEXC1), you can execute buffered commands; otherwise (COMEXC0), you must prefix each command with an immediate command identifier (e.g., !A, !V, !D, !MC, 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). On-the-fly motion changes are applicable only for motion started with the GO command, and not for motion started with HOM or PRUN.

On-the-fly motion changes are most likely to be used to change the velocity and/or goal position of a preset move already underway. In the event that the goal position is completely unknown before motion starts, a move may be started in continuous mode (MC1), with a switch to preset mode (MC0), a distance command (D), and a GO given later. In absolute positioning mode (MA1) the new goal position given with a pre-emptive GO is explicit in the D command. In incremental positioning (MA0) the distance given with a new pre-emptive GO is always measured from the at-rest position before the original GO. If a move is stopped (with the S command), and then resumed (with the C command), this resumed motion is considered to be part of the original GO. A subsequent distance given with a new pre-emptive GO is measured from the at rest position before the original GO, not the intermediate stopped position.

Programming Example: *This program creates a 2-tiered profile that changes velocity and deceleration when specific inputs are activated.*

```
DEL PROG17        ; Delete program #17
DEF PROG17        ; Begin definition of program #17
PSET0             ; Set position to zero
COMEXC1           ; Enable continuous command processing mode
MC0               ; Select preset positioning
MA0               ; Select incremental positioning
A20               ; Set accel to 20 revs/sec/sec
AD20              ; Set decel to 20 revs/sec/sec
V9                ; Set velocity to 9 revs/sec
D500000           ; Set distance to 500,000 counts
GO1               ; Initiate motion
WAIT(IN.5=b1)     ; Wait until input #5 is activated
V4                ; Slow down for machine operation
GO1               ; Initiate new profile with new velocity
WAIT(IN.6=b1)     ; Wait until input #6 is activated
AD5               ; Set decel for gentle stop
V1                ; Slow down for gentle stop
GO1               ; Initiate new profile with new velocity
END               ; End program definition
```

OTF Error Conditions

Further instructions about handling error conditions are provided on page 26 and in the ERROR and ERRORP command descriptions.

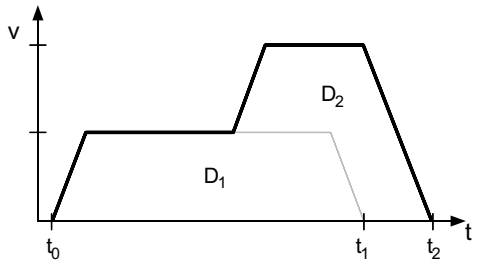
The ability to change the goal position on the fly raises the possibility that the new position goal of an on-the-fly GO cannot be reached with the present direction, velocity, and deceleration. If this happens, an error condition is flagged in axis status (TAS) bit #30 and in error status (TER) bit #10.

If the direction of the new goal position is opposite that of present travel direction, the Gemini will kill motion (decelerating at the LHAD value) and set TAS bit #30 and TER bit #10. Refer to scenario #2 below.

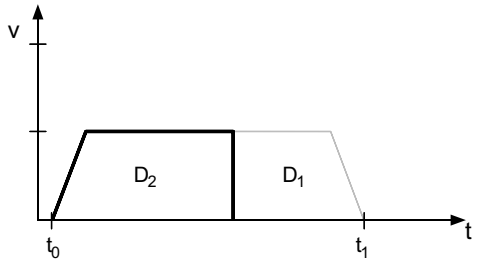
If there has not yet been an overshoot, but it is not possible to decelerate to the new distance from the present velocity using the specified AD value, this case is considered an overshoot — the Gemini with kill the move and set TAS bit #30 and TER bit #10.

Scenarios

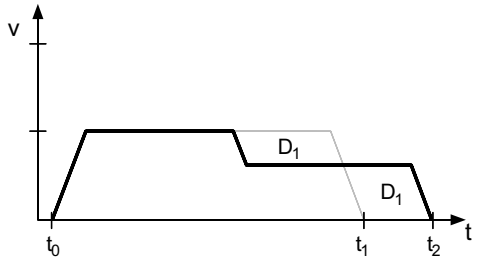
Scenario #1: OTF change of velocity and distance, where new commanded distance (D_2) is greater than the original distance (D_1) that was pre-empted [$D_2 > D_1$]. The distances are the areas under the profiles, starting at t_0 for both. If the original move had continued, D_1 would have been reached at time t_1. D_2 is reached at time t_2.



Scenario #2: OTF change of distance, where new commanded distance (D_2) is less than the original distance (D_1) that was pre-empted [$D_2 < D_1$]. In this example, the position where the OTF change was entered is already beyond D_2 (or D_2 can not be reached with the commanded deceleration). The result is an error and motion is killed (decel at the LHAD value) and TAS bit #30 and TER bit #10 are set.



Scenario #3: OTF change of velocity. Note that motion must continue for a longer time at the reduced velocity to reach the original commanded distance than if it had continued at the original velocity ($t_2 > t_1$).



Registration

The Gemini drive offers a “Registration” feature, which allows you to interrupt motion in progress with a predefined registration profile. The interrupt is triggered by activating a “Registration Input” (an input configured with the INFNCi-H command.

When a *registration input* is activated, the motion profile currently being executed is replaced by the 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) or continuous (MC1) move in progress. However, a registration move in progress cannot be interrupted by a secondary registration move.

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 Gemini is operating in the continuous command execution mode (COMEXC1).

For further details and programming samples, refer to the RE description on page 141.

Compiled Motion Profiling

The Gemini drive allows you to design compiled motion profiles. Because the motion and functions are *pre-compiled*, delays associated with command processing are eliminated during profile execution, allowing more rapid sequencing of actions than would be possible with programs which are not compiled. Command processing is then free to monitor other activities such as I/O and communications.

Compiled profiles are defined similar to programs, using the DEF PROF and END commands (the profile is automatically compiled when the Gemini drive executes the END command), and executed with the PRUN PROF command. The commands that can be used in a compiled profile are:

A.....Acceleration.
AD.....Deceleration.
D.....Distance.
MC.....Continuous or preset (incremental) positioning mode.
V.....Velocity.
VF.....Final velocity of the segment or profile.
GOBUF .....Store a compiled motion segment. A profile is constructed by sequentially appending motion segments using GOBUF commands. Each GOBUF motion segment may have its own distance to travel, velocity, acceleration, and deceleration.
GOWHEN ....Delay execution of the subsequent GOBUF statement until the specified time delay (in milliseconds) has been satisfied. During the time delay, the profile in progress continues at constant velocity. For example, when progress through the profile reaches the GOWHEN(T=500) command, execution of the subsequent GOBUF is paused for ½ second.
TRGFN .....Suspend execution of a subsequent GOBUF until a specified trigger interrupt input is activated. For example, TRGFND1 suspends execution of the GOBUF until you activate input #4.
PLOOP .....Beginning of loop.
PLN.....End of loop.
POUTA .....Turn on/off a “general-purpose” output. If you attempt to change an output that is not defined as a “general-purpose” output (OUTFNCi-A), the POUTA command will be ignored. By factory default, outputs #1 and #6 can be controlled. For example, POUTA.1-1 turns on output #1.
SGENB .....Enable a servo gain set.

GOWHEN & TRGFN Status: ➔
Axis Status (TAS) bit #26 is set when a GOBUF is suspended, pending activation of a TRGFN trigger input, or pending a GOWHEN time delay.

Up to 16 compiled profiles may be defined and stored in the Gemini drive. Compiled profiles are stored in the *compiled* portion of the Gemini's EEPROM memory. To check the amount of memory available, use the `TMEM` command (the right-hand value indicates the number of bytes remaining in compiled memory). To check which profiles are stored in the Gemini, use the `TDIR` command.

Constructing a Compiled Profile

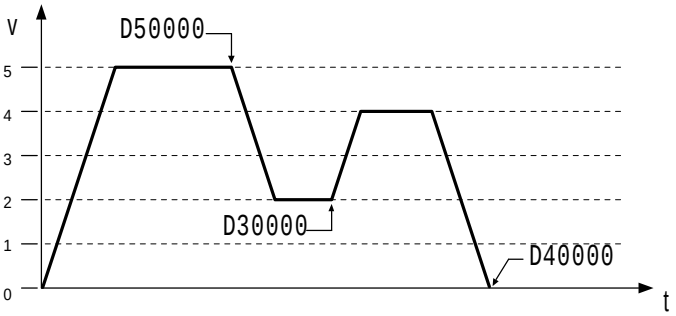
A compiled profile is constructed by sequentially appending motion segments using `GOBUF` commands. Each `GOBUF` motion segment may have its own distance to travel, velocity, acceleration, and deceleration – this is demonstrated in the following program.

The end of a `GOBUF` motion segment in preset mode (`MC0`) is determined by the specified distance (`D`). The end of a `GOBUF` motion segment in continuous mode (`MC1`) is determined by the specified goal velocity (`V` or `VF`). In both cases, the final velocity and position achieved by a segment will be the starting velocity and position for the next segment. If either type of segment is followed by a `GOWHEN` command, the segment's final velocity will be maintained until the `GOWHEN` condition evaluates true.

Example:

```
DEL PROF1 ; Delete profile #1
DEF PROF1 ; Begin definition of profile #1
MC0       ; Use the incremental preset positioning mode
D50000    ; Distance is 50000
A10       ; Acceleration is 10
AD10      ; Deceleration is 10
V5        ; Velocity is 5
GOBUF1    ; Store the first motion segment.
          ; Attributes are: MC0, A10, AD10, V5, D50000
D30000    ; Distance is 30000
V2        ; Velocity is 2
GOBUF1    ; Store the second motion segment.
          ; Attributes are: MC0, A10, AD10, V2, D30000
D40000    ; Distance is 40000
V4        ; Velocity is 4
GOBUF1    ; Store the third motion segment.
          ; Attributes are: MC0, A10, AD10, V4, D40000
          ; Because this is the last segment in a preset
          ; profile, the velocity will automatically end at zero.
END       ; End profile definition
```

The resulting profile from the above program when you execute `PRUN PROF1`:

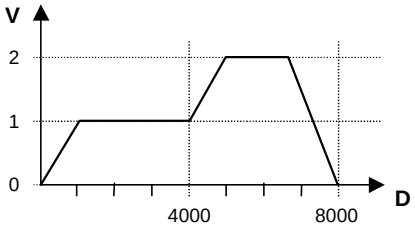


Rules for Using Velocity in Preset Compiled Motion

When defining preset mode (MC0) compiled profiles there are several rules that govern the velocity.

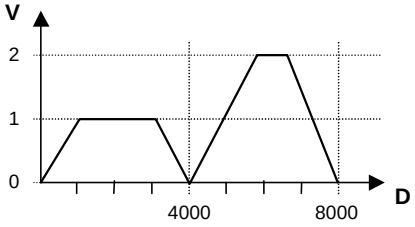
Rule #1: The last segment in the compiled profile will automatically end at zero velocity (only if not in a PLOOP/PLN loop).

```
DEF PROF6   ; Begin definition of profile #6
MC0         ; Select preset positioning
V1          ; Set velocity to 1 rev/sec
D4000       ; Set distance to 4000
GOBUF1      ; First motion segment (V1, D4000)
V2          ; Set velocity to 2 (second segment)
GOBUF1      ; Second motion segment (V2, D4000)
END         ; End definition of profile #6
; When you execute PRUN PROF5, the resulting profile is:
```



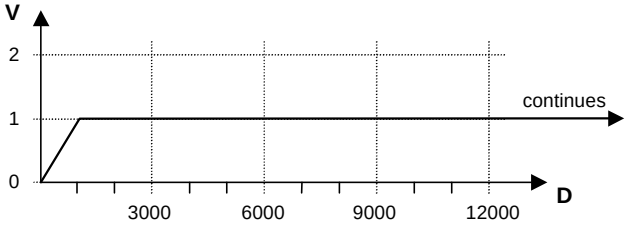
Rule #2: If you wish intermediate segments to end in zero velocity, use the VF0 command in the respective GOBUF segment.

```
DEF PROF5   ; Begin definition of profile #5
MC0         ; Select preset positioning
V1          ; Set velocity to 1 rev/sec
VF0         ; End this segment at zero velocity
D4000       ; Set distance to 4000
GOBUF1      ; First motion segment (V1, D4000, VF0)
V2          ; Set velocity to 2 (second segment)
VF0         ; End this segment at zero velocity
GOBUF1      ; Second motion segment (V2, D4000, VF0)
END         ; End definition of profile #5
; When you execute PRUN PROF6, the resulting profile is:
```



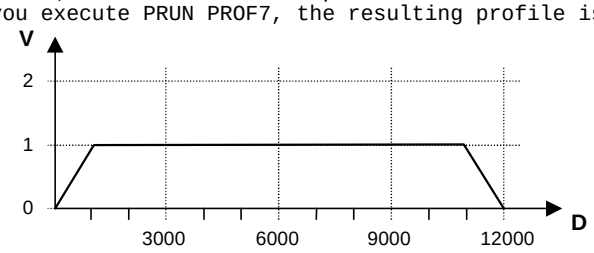
Rule #3: **WARNING**: With compiled loops (PLOOP and PLN), the last segment within the loop must end at zero velocity or there must be a final segment placed outside the loop. Otherwise, after the final segment is completed, the motor will continue moving at the last segment's velocity.

```
DEF PROF7   ; Begin definition of profile #7
MC0         ; Select preset positioning
D3000       ; Set distance to 3000
PLOOP4      ; Loop (between PLOOP & PLN) 4 times
V1          ; Set velocity to 1 rev/sec
GOBUF1      ; First motion segment
PLN1        ; End loop
END         ; End definition of profile #7
; When you execute PRUN PROF7, the resulting profile is:
```



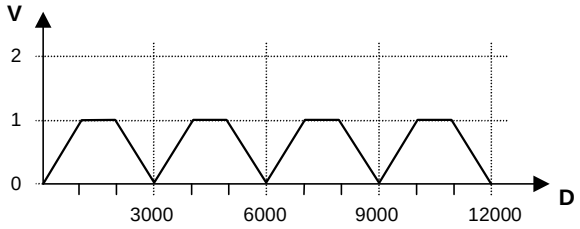
To fix the profile, reduce the PLOOP count by one and add a GOBUF statement after the PLN command:

```
DEF PROF7   ; Begin definition of profile #7
MC0         ; Select preset positioning
D3000       ; Set distance to 3000
PLOOP3      ; Loop (between PLOOP & PLN) 3 times
V1          ; Set velocity to 1 rev/sec
GOBUF1      ; Looped motion segment
PLN1        ; End loop
GOBUF1      ; Last motion segment (end at zero velocity)
END         ; End definition of profile #7
; When you execute PRUN PROF7, the resulting profile is:
```



Rule #4: With compiled loops (PLOOP and PLN), if you wish the velocity at the end of each loop to end at zero, use a VF0 command.

```
DEF PROF8   ; Begin definition of profile #8
MC0         ; Select preset positioning
D3000       ; Set distance to 3000
PLOOP4      ; Loop (between PLOOP & PLN) 4 times
V1          ; Set velocity to 1 rev/sec
VF0         ; End each segment at zero velocity
GOBUF1      ; Looped motion segment
PLN1        ; End loop
END         ; End definition of profile #8
; When you execute PRUN PROF8, the resulting profile is:
```



Dwells and Direction Changes

Compiled profiles may incorporate changes in direction only if the preceding motion segment has come to rest. This may be achieved by creating a continuous segment with a goal velocity of zero, or by preceding a preset segment with VF0. Motion within the profile comes to rest, although the profile is not yet complete. Even though the motor is not moving, axis status (TAS) bit 1 will remain set, indicating a profile is still underway. Only then can you change direction using the D+, D-, or D~ command within a profile. An attempt to incorporate changes in direction if the preceding motion segment has not come to rest will cause a fault.

In many applications, it may be useful to create a time delay between moves. For example, a machine cycle may require a move out, dwell for 2 seconds, and move back. To create this dwell, a compiled GOWHEN may be used between the two moves. The code within a compiled program may look like:

```
MC0             ; Preset positioning used
D26000          ; Target distance is 26000 counts
VF0             ; Motion comes to rest at end of move
GOBUF1          ; Create the "move out" segment
GOWHEN(T=2000)  ; Profile delays for 2 seconds
D-              ; Return to starting position (direction reversed)
VF0             ; Motion comes to rest at end of move
GOBUF1          ; Create the "move back" segment
```

Compiled Motion vs. On-the-Fly Motion

The two basic ways of creating a complex profile are with compiled motion or with on-the-fly pre-emptive GO commands (see page 44). With compiled motion, portions of a profile are built piece by piece, and stored for later execution. Compiled motion is appropriate for profiles with motion segments of pre-determined velocity, acceleration and distance. Compiled motion profiles allow for shorter motion segments, which results in faster cycle times because there is no command processing and execution delay, thus freeing program flow for other tasks, such as I/O, machine control, and host requests. The disadvantages to pre-defined compiled motion profiles are the amount of memory use and limited run-time decision making and I/O processing.

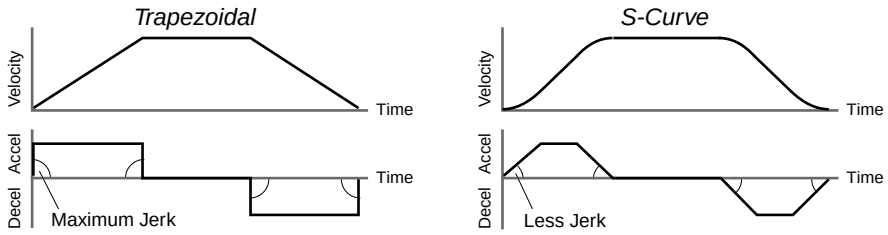
With pre-emptive GO moves, the motion profile underway is pre-empted with a new profile when a new GO command is issued. The new GO command constructs and launches the pre-empting profile. Pre-emptive GOs are appropriate when the desired motion parameters are not known until motion is already underway.

The table below summarizes the differences between the use of compiled motion and on-the-fly motion.

Command/Issue	Compiled Motion	On-The-Fly Motion
GOBUF	Constructs motion segment and appends to previously constructed segment.	N/A
PRUN PROF	Used to launch compiled motion profiles.	N/A
GO	Ignored while executing a compiled profile with PRUN PROF.	Constructs & launches profile, even if moving.
Direction changes	Only if previous motion segment comes to rest; otherwise, a fault may result (GT6: TAS bit 12 and TER bit 1; GV6: TAS bit 23 and TER bit 12).	Not allowed during motion; a fault will result (TAS bit 30 and TER bit 10 are set).
Insufficient room for AD (decel) value	Same as on-the-fly.	Motion is killed, TAS bit 30 and TER bit 10 are set.

S-Curve Accel/Decel Profiling

Gemini drives allow 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.



S-Curve Programming Requirements

To program an S-curve profile, you must use the *average accel/decel* commands provided in the Gemini programming language. For every maximum accel/decel command (A, AD, LHAD, and LSAD) there is an *average* command for S-curve profiling (see table below).

Maximum Accel/Decel Commands:		**"S-Curve" Accel/Decel Commands:**	
Command	**Function**	**Command**	**Function**
A	Acceleration	AA	Average Acceleration
AD	Deceleration	ADA	Average 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 $\boxed{\frac{1}{2}A \leq AA < A}$, 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 you will receive the "INVALID_DATA" error.
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 deceleration command equal to zero, you will receive the "INVALID_DATA" error.
AA ≠ zero and AA ≠ A	S-curve profiling is enabled **only for standard moves** (e.g., not for compiled motion, or on-the-fly motion changes). All subsequent standard moves must comply with this equation: $\frac{1}{2}A \leq AA < A$.
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 commands, ADA will track AA acceleration. However, once you change AD deceleration, ADA deceleration will no longer track changes in AA acceleration.

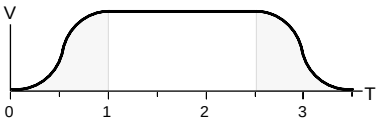
The calculation for determining S-curve average accel and decel move times is as follows (*calculation method identical for S-curve and trapezoidal moves*):

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

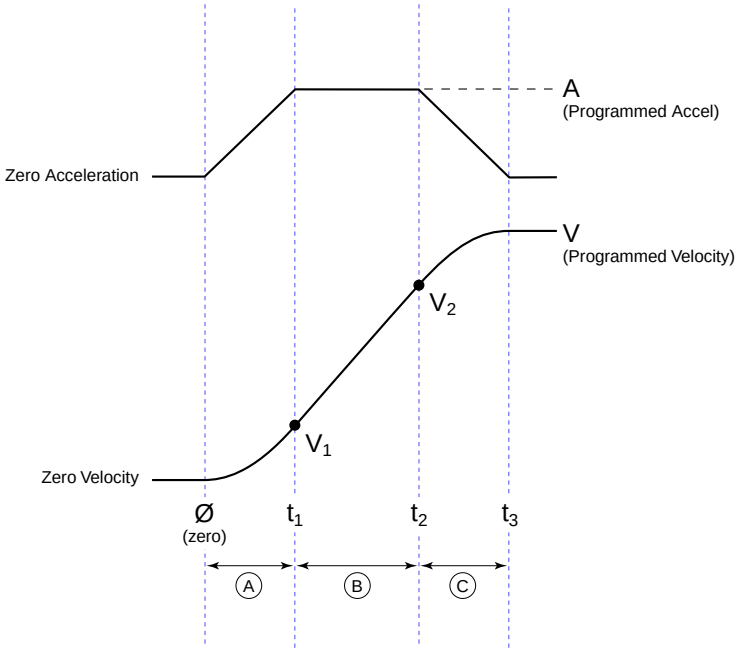
Programming Example

```
; In this example, the Gemini executes a pure S-curve and takes
; 1 second to reach a velocity of 5 rps.

DEF PROG16 ; Define program #16
MC0        ; Select preset positioning mode
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 program #16
```

Move profile:



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:

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

A = programmed acceleration (A, AD, LHAD, LSAD)
AA = average acceleration (AA, ADA, LHADA, LSADA)
V = programmed velocity (V)

(continued on next page)

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

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

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

NOTE: $t_3 - t_2 = t_1$

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

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

Ⓐ $t_1 \geq t \geq \emptyset$

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

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

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

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 "$(V_F - V_O)$" for "V" (V_F = final velocity, V_O = initial velocity). For example, the jerk equation would be:

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

Command Descriptions

A — Acceleration

		Product	Rev
Type	Motion	GT	n/a
Syntax	<a_><!>A<r>	GV	n/a
Units	r = revs/sec/sec (linear motors: see page 44 for linear/rotary conversion)	GT6	1.50
Range	0.0001 – 9999.9999	GV6	1.50
Default	10.0000		
Response	A: *A10.0000		
See Also	AA, AD, ADA, DMEPIT, DRES, ERES, GO, IF, MC, VARI, WAIT		

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

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).

The A 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 A command value. For details, see page 24.

Example:

```
DEL PROG7      ; Delete program #7
DEF PROG7      ; Begin definition of program #7
MA0            ; Incremental positioning mode
MC0            ; Preset positioning mode
A40            ; Set the acceleration to 40 revs/sec/sec
AD16           ; Set the deceleration to 16 revs/sec/sec
V1             ; Set the velocity to 1 revs/sec
D100000        ; Set the distance to 100000 counts
GO1            ; Initiate motion
END            ; End definition of program #7
```

AA — Average Acceleration

		Product	Rev
Type	Motion (S-Curve)	GT	n/a
Syntax	<a_><!>AA<r>	GV	n/a
Units	r = revs/sec/sec (linear motors: see page 44 for linear/rotary conversion)	GT6	1.50
Range	0, or 0.0001 – 9999.9999	GV6	1.50
Default	10.0000 (default is trapezoidal profiling, where AA tracks A; to restore trapezoidal profiling, set AA = 0)		
Response	AA: *AA10.0000		
See Also	A, AD, ADA, DMEPIT		

The Average Acceleration (AA) command allows you to specify the average acceleration for an S-curve motion profile. S-curve profiling provides smoother motion control by reducing the rate of change in

acceleration and deceleration; this accel/decel rate of change is known as *jerk*. Refer to page 53 for details on S-curve profiling.

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, program #1 executes a pure S-curve and takes 1 second
; to reach a velocity of 5 rps; program #2 executes a trapezoidal profile
; and takes 0.5 seconds to reach a velocity of 5 rps.
DEL PROG1     ; Delete program #1
DEF PROG1     ; Begin definition of program #1
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 program #1

DEL PROG2     ; Delete program #2
DEF PROG2     ; Begin definition of program #2
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

		Product	Rev
Type	Motion	GT	n/a
Syntax	<a_><!>AD<r>	GV	n/a
Units	r = revs/sec/sec (linear motors: see page 44 for linear/rotary conversion)	GT6	1.50
Range	0, or 0.0001 – 9999.9999	GV6	1.50
Default	10.0000 (AD tracks A; to restore tracking, set AD = 0)		
Response	AD: *AD10.0000		
See Also	A, AA, ADA, DMEPIT, DRES, ERES, GO, IF, LHAD, LSAD, MC, VARI, WAIT		

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

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

ON-THE-FLY CHANGES: 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).

The AD 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 AD command value. For details, refer to page 24.

Example:

```
DEL PROG7      ; Delete program #7
DEF PROG7      ; Begin definition of program #7
MA0            ; Incremental positioning mode
MC0            ; Preset positioning mode
A40            ; Set the acceleration to 40 revs/sec/sec
AD16           ; Set the deceleration to 16 revs/sec/sec
V1             ; Set the velocity to 1 revs/sec
D100000        ; Set the distance to 100000 counts
GO1            ; Initiate motion
END            ; End definition of program #7
```

ADA — Average Deceleration

Type	Motion (S-Curve)
Syntax	<a_><!>ADA<r>
Units	r = revs/sec/sec (linear motors: see page 44 for linear/rotary conversion)
Range	0, or 0.0001 – 9999.9999
Default	10.0000 (default is a constant deceleration ramp, where ADA tracks AA; to restore tracking, set ADA = 0)
Response	ADA: *ADA10.0000
See Also	A, AA, AD, DMEPIT, LHADA, LSADA

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

The Average Deceleration (ADA) command allows you to specify the average deceleration for an S-curve motion profile. S-curve profiling provides smoother motion control by reducing the rate of change in acceleration and deceleration; this accel/decel rate of change is known as *jerk*. Refer to page 53 for details on S-curve profiling.

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)

ADDR — Multiple Unit Auto-Address

Type	Communication Interface
Syntax	<a_><!>ADDR<i>
Units	i = unit number (address)
Range	0-99
Default	0
Response	ADDR: *ADDR0
See Also	E

Product	Rev
GT	1.02
GV	1.00
GT6	1.50
GV6	1.50

The factory default address for a Gemini drive is address zero (0). The ADDR command automatically configures unit addresses for a daisy-chain or multi-drop. This 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).

RS-232C Daisy Chain:

Sending ADDRi 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).

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_ADDRi command to it, where "i" is the address you wish the new unit to have.

To send a Gemini 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 Gemini product) may receive data from a multi-drop unit.

For more information on controlling multiple Gemini Series drives in an RS-232 daisy-chain or RS-485 multi-drop, refer to your Gemini drive's *Hardware Installation 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.
```

ANICDB — Analog Input Center Deadband

Type	Drive Configuration
Syntax	<a_><!>ANICDB<r>
Units	volts
Range	0.00 – 10.00
Default	0.04
Response	ANICDB: *ANICDB0.04
See Also	DCMDZ, DMTSCL, DMVSCL

Product	Rev
GT	1.61
GV	1.70
GT6	n/a
GV6	n/a

ANICDB allows the user to specify the voltage deadband for the command input. ANICDB is used with DCMDZ to configure the command input for DMODE2 and DMODE4. In DMODE4, the commanded velocity, $Vel_{command}$, is calculated from the input voltage, V_{in}, using DMVSCL as follows:

$$Vel_{command} = (V_{in} - \texttt{DCMDZ} - \texttt{ANICDB}) * \frac{\texttt{DMVSCL}}{10} \quad \text{when} \quad V_{in} > (\texttt{DCMDZ} + \texttt{ANICDB})$$

$$Vel_{command} = 0 \quad \text{when} \quad (\texttt{DCMDZ} - \texttt{ANICDB}) \le V_{in} \le (\texttt{DCMDZ} + \texttt{ANICDB})$$

$$Vel_{command} = (V_{in} - \texttt{DCMDZ} + \texttt{ANICDB}) * \frac{\texttt{DMVSCL}}{10} \quad \text{when} \quad V_{in} < (\texttt{DCMDZ} - \texttt{ANICDB})$$

Similarly, the commanded torque in DMODE2 is calculated using DMTSCL in the above equations.

BOT — Beginning of Transmission Characters

Type	Communication Interface
Syntax	<i>BOT<i>,<i>,<i>
Units	i = numeric equivalent for ASCII character
Range	i = 0-255
Default	0,0,0
Response	BOT: *BOT0,0,0
See Also	EOT, ERROK, ERRBAD, EOL

Product	Rev
GT	1.02
GV	1.00
GT6	1.50
GV6	1.50

The Beginning of Transmission Characters (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. If the first field is a zero, the drive will only accept zeros from the other two fields.

NOTE: This command is intended to be used only during live terminal communication with the drive. Do not download this command to the drive, or place it in a program.

For a more complete list of ASCII Equivalents, refer to the ASCII Table in Appendix C.

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
```

C — Continue Command Execution

Type	Program Flow Control
Syntax	<a_>!C
Units	n/a
Range	n/a
Default	n/a
Response	n/a
See Also	COMEXR, COMEXS, INFNC, PS, S

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

The Continue (!C) command ends a pause state (PS), or a stopped (S) condition. When the Gemini is in a paused state, 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. 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

COMEXS1       ; Enable command processing on stop
D50000        ; Set distance to 50000 counts
GO1           ; Initiate motion
!S            ; Stop motion
```

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

CERRLG — Clear the Error Log

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

Product	Rev
GT	1.02
GV	1.00
GT6	1.50
GV6	1.50

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 error occurs. 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 - GV only; GT always reports zero)

COMEXC Continuous Command Processing Mode

		Product	Rev
Type	Command Buffer Control	GT	n/a
Syntax	<a_><!>COMEXC<b>	GV	n/a
Units	b = 0 or 1	GT6	1.50
Range	0 = Disable, 1 = Enable	GV6	1.50
Default	0		
Response	COMEXC: *COMEXC0		
See Also	A, AA, AD, ADA, COMEXL, COMEXS, D, ERRORP, 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; the list below identifies all parameters that cannot be changed while motion is in progress.

- DRES.....Drive Resolution
- DRIVE .....Drive Enable/Shutdown
- DMVLIM.....Velocity Limit
- ERES.....Encoder Resolution
- HOM.....Initiate Home Move
- HOMA.....Home Acceleration
- HOMV.....Home Velocity
- HOMVF .....Home Final Velocity
- LHAD.....Hardware EOT Limit Deceleration
- LHADA .....Hardware EOT Limit Decel (S-Curve)
- LSAD.....Software EOT Limit Deceleration
- LSADA .....Software EOT Limit Decel (S-Curve)
- DMODE .....Drive Control Mode

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

- When trying to check the status of inputs while the Gemini product 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), and velocity (V) changes. (The buffered A, AA, AD, ADA, D, 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 page 42.
- 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.

CAUTION: Avoid Executing Moves Prematurely

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 on page 37 for details).

Example:

```
DEL PROG8      ; Delete program #8
DEF PROG8      ; Begin definition of program #8
COMEXC1        ; Enable continuous command execution mode
L50            ; Loop 50 times
D50000         ; Set distance to 50000 counts
GO1            ; Initiate motion
; Normally at this point, the Gemini 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
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:

```
DEL PROG9     ; Delete program #9
DEF PROG9     ; Begin definition of program #9
COMEXC1       ; Enable continuous command execution mode
MC1           ; Set mode to continuous
A10           ; Set acceleration to 10
V1            ; Set velocity to 1
GO1           ; Initiate move (Go)
T3            ; Time delay of 3 seconds
A50           ; Set acceleration to 50
V10           ; Set velocity to 10
GO1           ; Initiate move (Go)
T2            ; Time delay of 2 seconds
S1            ; Initiate stop of move
WAIT(AS.1=b0) ; Wait for no commanded motion
COMEXC0       ; Disable continuous command execution mode
END           ; End definition of program vsteps
```

COMEXL Continue Execution on Limit

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

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

This command 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
Syntax	<a_><!>COMEXR<b>
Units	b = 0 or 1
Range	0 = disable, 1 = enable
Default	0
Response	COMEXR: *COMEXR0
See Also	C, COMEXS, INFNC

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

The Continue Motion on Pause/Continue (COMEXR) command determines the functionality of programmable inputs defined as pause/continue inputs with the INFNCi-E command. When the input is activated, the current 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
INFNC1-E      ; Define input 1 as a pause/continue input
```

COMEXS Continue Execution on Stop

		Product	Rev
Type	Command Buffer Control	GT	n/a
Syntax	<a_><!>COMEXS<i>	GV	n/a
Units	i = function identifier	GT6	1.50
Range	0, 1, or 2	GV6	1.50
Default	0		
Response	COMEXS: *COMEXS0		
See Also	C, COMEXC, COMEXL, COMEXR, INFNC, S		

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

COMEXS0: Under factory default conditions (COMEXS0), when the Gemini drive 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.
- If S, !S or Stop input:
  - All commands in the Gemini drive's command buffer are discarded.
  - Program execution is terminated and cannot be resumed.
- If S1, or !S1:
  - All commands in the Gemini drive's command buffer are retained.
  - Program execution continues.

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

Stop Method	What Stops? Motion	Program	**Resume Motion Profile**. Allow resume with a !C command or a resume input (INFNCi-E).	**Resume Program**. Allow resume with a !C command or a resume input (INFNCi-E).	**Save Command Buffer**. Save the commands that were in the command buffer when the stop was commanded.
!S or S	Yes	Yes	Yes	Yes	Yes
!S1 or S1	Yes	No	No	No	Yes
Stop input	Yes	Yes	Yes	Yes	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 input that can be used to resume motion and program execution after a stop.

COMEXS2: Using the COMEXS2 mode, the drive responds as it does in the COMEXS0 mode, with the exception that you can still use the BCD inputs to select programs (INSELP value is retained). For more details on BCD program selection, refer to INFNC and INSELP.

D — Distance

Type	Motion
Syntax	<a_><!>D<r>
Units	r = distance (counts)
Range	-2,147,483,648 to +2,147,483,647
Default	4000
Response	D: *D+4000
See Also	DMEPIT, DRES, ERES, GO, IF, MA, MC, PSET, VARI, WAIT

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

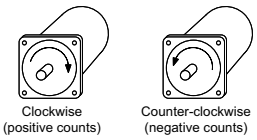
The Distance (D) command defines either the number of counts the motor will move or the absolute position it will seek after a GO command. In the incremental mode (MAØ), the distance value represents the total number of units you wish the motor to move. In the absolute mode (MA1) the distance value represents 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 D 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 D command value. For details, see page 24.

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:
```
DEL PROG2      ; Delete program #2
DEF PROG2      ; Begin definition of program #2
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
```

DABSD — Enable ABS Damping

Type	Drive Configuration
Syntax	<a_><!>DABSD<b>
Units	b = enable bit
Range	0 (disable) or 1 (enable)
Default	0 (disabled)
Response	DABSD: *DABSD1
See Also	DACTDP, DDAMPA, DELVIS

Product	Rev
GT	1.02
GV	n/a
GT6	1.50
GV6	n/a

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).

Gemini Damping Features: The Gemini 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 your drive's *Hardware Installation Guide*.)

Command	Damping Function	Velocity *	Default	Related Parameters *
DABSD	ABS Damping. 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	Electronic Viscosity. 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	Active Damping. 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	Damping During Acceleration. 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) or Pocket Motion Planner (see page 11). *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
Syntax	<a_><!>DACTDP<i>
Units	i = gain
Range	0-40; 0 disables active damping
Default	4
Response	DACTDP *DACTDP0
See Also	DABSD, DELVIS, DMTR, LJRAT

Product	Rev
GT	1.02
GV	n/a
GT6	1.50
GV6	n/a

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 GT/GT6 damping features is provided in the DABSD command description; see page 65.

DAUTOS Auto Current Standby

		Product	Rev
Type	Drive Configuration	GT	1.02
Syntax	<a_><!>DAUTOS<r>	GV	n/a
Units	r = % reduction of motor current	GT6	1.50
Range	0.00-99.99 : ±0.01	GV6	n/a
Default	0.00 (no current reduction)		
Response	DAUTOS *DAUTOS0		
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 received a step pulse for one second. Full commanded current is restored upon the first step pulse that the drive receives.

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.

DCLRLR Clear the Latched Status Register Bits

		Product	Rev
Type	Drive Configuration	GT	1.02
Syntax	<a_><!>DCLRLR	GV	1.00
Units	n/a	GT6	1.50
Range	n/a	GV6	1.50
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 command, cycle power, or activate the Reset input.

TASX bit 18 Commanded Velocity Exceeds DMVLIM Limit (GV/GV6 and GT/GT6):
Bit 18 is set when the velocity demand from a controller or the internal Gemini control loops exceeds the limits set by the DMVLIM command. The GV/GV6 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 GV or GV6 drive in all operating modes (DMODE).

TASX bit 19 Bridge is in Foldback Mode (GV/GV6 only):
When a Gemini 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 (GV/GV6 and GT/GT6):
When a Gemini 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 GT drives, the GV-L3, and the GV-H20, 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 GV-U3, GV-U6, and GV-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 (GV/GV6):
When the Gemini's commanded torque/force reaches the limit set by DMTLIM (TTRQ = DMTLIM), TAS bit #31 is set. This is not considered a fault condition.

DCMDZ Zero the Drive Command Offset

Type	Drive Configuration
Syntax	<a_><!>DCMDZ<r>
Units	volts
Range	-10.00 – 10.00
Default	0.00
Response	n/a
See Also	ANICDB, DMODE

Product	Rev
GT	1.02
GV	1.00
GT6	n/a
GV6	n/a

The DCMDZ command sets the zero point for the command input. When in velocity mode (DMODE4) or torque/force mode (DMODE2), this will minimize motor drift.

Executing the DCMDZ command without an argument sets the zero reference point to the last voltage read at the command input. For this command to be executed correctly, the Command + and Command - inputs must be shorted together, or zero volts must be commanded from the servo controller.

DCMDZ can also be used to set the zero point to an arbitrary voltage by entering that value. For example, DCMDZ0.5 will make 0.5 volts equal to a commanded velocity of zero rps. Note that this value is the internal level and will not take into account any offsets in the incoming command signal.

DDAMPA Damping During Acceleration/Deceleration

Type	Drive Configuration
Syntax	<a_><!>DDAMPA<b>
Units	b = enable bit
Range	0 (disable damping) or 1 (enable damping)
Default	0 disabled)
Response	DDAMPA: *DDAMPA1
See Also	DABSD, DELVIS

Product	Rev
GT	1.02
GV	n/a
GT6	1.50
GV6	n/a

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/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 GT's damping features is provided on page 66.

DEF PROF Begin Profile Definition

Type	Compiled Motion
Syntax	<a_><!>DEF PROF<i>
Units	i = profile ID number
Range	1-16
Default	n/a
Response	n/a
See Also	DEF PROG, DEL PROF, END, GOBUF, PRUN PROF, TDIR, TMEM, TSS (see *Compiled Motion* section on page 49)

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

The DEF PROF command marks the beginning of a profile definition. For example, the DEF PROF6 command begins definition of profile #6. Up to 16 profiles may be defined.

As soon as the Gemini drive receives a subsequent END command, the profile is compiled and stored in the "profile" partition of the Gemini's non-volatile memory. Profiles remain stored until you deleted them with the DEL PROF command or issue an RFS command. To check the status of stored profiles, use the TMEM command. To report the names of all stored profiles, use the TDIR command.

NOTE

When defining a profile and the memory limitation is reached, the drive will respond with the ERRBAD prompt (default prompt is "?"), and the profile will be stored up to the point where the memory limitation was reached.

To execute a specific profile, issue the PRUN PROF command or the PROF command (e.g., you can use either PRUN PROF6 or PROF6 to execute compiled profile #6).

NOTE: The profile must be deleted (e.g., DEL PROF6) before it can be redefined.

Example:

```
DEL PROF1      ; Delete profile #1
DEF PROF1      ; Begin definition of profile #1
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
D40000         ; Distance is 40000
V4             ; Velocity is 4 revs/sec
GOBUF1         ; 3rd motion segment
END            ; End program definition
```

DEF PROG Begin Program Definition

Type	Program Definition
Syntax	<a_><!>DEF PROG<i>
Units	i = program ID number
Range	1-32
Default	n/a
Response	n/a
See Also	DEF PROF, DEL PROG, END, GOSUB, JUMP, RUN PROG, STARTP, TDIR, TMEM, TPROG, TSS

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

The DEF PROG command marks the beginning of a program definition. For example, the DEF PROG3 command begins definition of program #3. Up to 32 programs may be defined.

All programs are stored in a binary fashion within the Gemini product. When you display a stored program (TPROG), or upload it to the Motion Planner or Pocket Motion Planner editor, it may not look identical to the originally defined program. However, the program is functionally identical. Programs are stored in the Gemini drive's memory, and remain stored until you deleted them with the DEL PROG command or issue an RFS command. To check the status of stored programs, use the TMEM command. To report the names of all stored profiles, use the TDIR command.

NOTE

When defining a program and the memory limitation is reached, the drive will respond with the ERRBAD prompt (default prompt is "?"), and the program will be stored up to the point where the memory limitation was reached.

Stored programs may be executed in different ways:

- Issue the RUN PROG command to start executing a program (e.g., RUN PROG3 executes program #3).
- Execute a specific program number by activating the corresponding "BCD Program Select" input (see INFNC and INSELP command descriptions).
- Branch to ("call") the program from within another program. Use one of these options:
  - Call as a subroutine with RUN PROG, PROG, or GOSUB PROG (e.g., RUN PROG3, PROG3, or GOSUB PROG3). These three commands are identical in function – they cause program flow to branch to the called program. After the called program is executed, processing returns to the

calling program at the next command after the branch command. Up to 16 nested subroutines are allowed.

  - `JUMP PROG` (e.g., `JUMP PROG3`). The `JUMP PROG` command branches to the specified program. All nested If conditions (`IF`), loops (`L`), and subroutines are cleared. Thus the program that the `JUMP PROG` command initiates will not return control to the calling program; instead, the called program will end.

- Assign the program as the "Startup Program" with the `STARTP` command (e.g., `STARTP PROG3` assigns program #3 as the startup program). When the Gemini drive is reset or powered up, the assigned `STARTP` program is automatically executed.

NOTE: The program must be deleted (e.g., `DEL PROG3`) before it can be redefined.

Example:
```
DEL PROG3     ; Delete program number 3
DEF PROG3     ; Begin definition of program number 3
GO1           ; Initiate motion
END           ; End program definition
RUN PROG3     ; Execute program number 3
```

DEL PROF Delete Profile

Type	Compiled Motion
Syntax	<a_><!>DEF PROF<i>
Units	i = profile ID number
Range	1-16
Default	n/a
Response	n/a
See Also	DEF PROF, END, PRUN PROF, TDIR, TMEM, TSS, (Compiled Motion overview on page 49)

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

The `DEL PROF` command deletes the specified profile from the Gemini drive's memory. For example, the `DEL PROF3` command deletes profile #3. The `DEL PROF` command cannot be placed inside a program.

NOTE: To edit an existing profile, you must first delete with the `DEL PROF` command.

Example:
```
DEL PROF1     ; Delete profile #1
DEF PROF1     ; Begin definition of profile #1
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
D40000        ; Distance is 40000
V4            ; Velocity is 4 revs/sec
GOBUF1        ; 3rd motion segment
END           ; End program definition
```

DEL PROG Delete Program

Type	Program Definition
Syntax	<a_><!>DEF PROG<i>
Units	i = program ID number
Range	1-32
Default	n/a
Response	n/a
See Also	DEF PROG, END, RUN PROG, TDIR, TMEM, TPROG, TSS

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

The `DEL PROG` command deletes the specified program from the Gemini drive's memory. For example, the `DEL PROG3` command deletes program #3. The `DEL PROG` command cannot be placed inside a program.

NOTE: To edit an existing program, you must first delete with the `DEL PROG` command.

Example: (see `DEF PROG` example)

DELVIS — Electronic Viscosity

Type	Drive Configuration
Syntax	<a_><!>DELVIS<i>
Units	i = gain
Range	0-7 (0 disables Electronic Viscosity)
Default	0 (disabled)
Response	DELVIS *DELVIS0
See Also	DACTDP, DABSD, DDAMPA, DMTR, LJRAT

Product	Rev
GT	1.02
GV	n/a
GT6	1.50
GV6	n/a

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 GT's damping features is provided on page 66.

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 Pocket Motion Planner or 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.

DIBW — Current Loop Bandwidth

Type	Tuning
Syntax	<a_><!>DIBW<i>
Units	i = Hz
Range	0-5000 (motor dependent)
Default	0 (DIBW of 0 results in motor configuration error)
Response	DIBW: *DIBW1000
See Also	DMTLIM, DMTSCL, DMVLIM, DNOTAF, DNOTAQ, DNOTLD, DNOTLG, DPBW, DVBW, SGIRAT, TASX, TCS, TGAIN, TSGSET

Product	Rev
GT	n/a
GV	1.00
GT6	n/a
GV6	1.50

AUTO-SETUP: This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6) or Pocket Motion Planner (see page 11). 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 your Gemini drive's *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
Syntax	<a_><!>DIFOLD<b>
Units	b = enable bit
Range	0 (disable) or 1 (enable)
Default	0
Response	DIFOLD: *DIFOLD0
See Also	

Product	Rev
GT	n/a
GV	1.00
GT6	n/a
GV6	1.50

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.

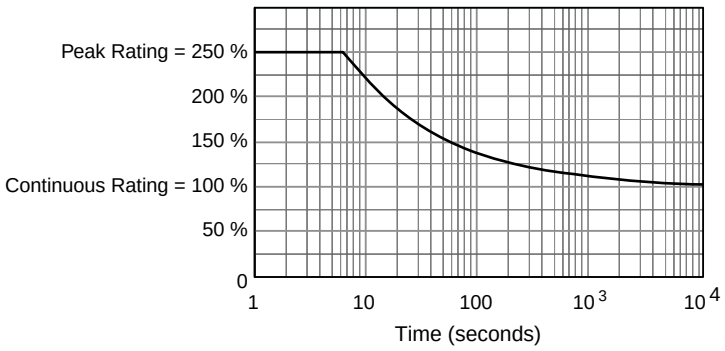
Each drive has the following specifications. Note that current ratings are for the *drive*, not for the *motor*.

	Units	GV-L3	GV-U3	GV-U6	GV-U12	GV-H20
Drive Continuous Current Rating (100%)	Amps *	3	3	6	12	20
Maximum Current Rating	Amps *	7.5	7.5	15	30	50
Maximum Time at Peak Current Rating	Seconds	6	6	6	6	6

* peak of the sinewave

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.

Drive Current Rating vs. Time



DIGN — Current Loop Gain

Type	Tuning
Syntax	<a_><!>DIGNc<r>
Units	c = gain identifier letter (required); r = gain value
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
Default	r = 0.000 (DIGNc of 0 results in motor config. warning)
Response	DIGNA: *DIGNA2.306
See Also	DMTR, TASX, TCS

Product	Rev
GT	1.02
GV	n/a
GT6	1.50
GV6	n/a

AUTO-SETUP: This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6) or Pocket Motion Planner (see page 11). 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).

The DIGN command sets the values of the gain terms for the stepper current loop. This allows drive performance to be optimized 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 * : $\text{DIGNA} = \frac{\text{Inductance (in mH)}}{2}$

DIGNB = DIGNA * 0.0896
DIGNC = DIGNA * 0.3578
DIGND = 0.98 (*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.

DMEPIT Motor Electrical Pitch

		Product	Rev
Type	Motor (Linear only)	GT	n/a
Syntax	<a_><!>DMEPIT<r> (does not take effect until RESET or cycle power)	GV	1.01
Units	r = millimeters	GT6	n/a
Range	0 to 327.68 : ±0.01	GV6	1.50
Default	0		
Response	DMEPIT: *DMEPIT40.00		
See Also	ERES, D, DRES		

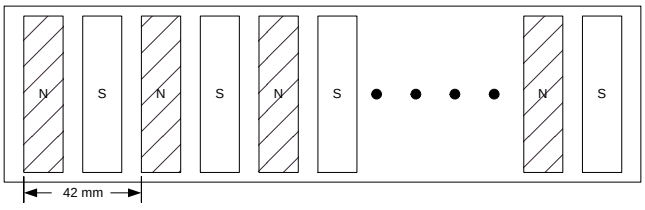
NOTE: This command does not take effect until you cycle power to the drive, or issue a RESET command.

AUTO-SETUP: This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6) or Pocket Motion Planner (see page 11). 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.

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 be equated 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 (left) shows an example of an electrical pitch of 42mm (DMEPIT42).

For all rotary motors, DMEPIT is set to zero.

Linear motor track



Definition of DMEPIT (Electrical Pitch)

Converting Between Rotary and Linear Values

The Gemini 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) and the configuration tool in Pocket Motion Planner (page 11) make it easy to perform the setup in linear units. The setup/configuration tool automatically converts your setup parameters (in linear units) to the appropriate Gemini code in rotary units. You then download the generated code/file to the drive. If you are communicating to the Gemini drive 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) DMVLIM (velocity limit) DMVSCL (velocity scaling) SMVER (max. allowable velocity error) TACC (display commanded accel) TACCA (display actual accel) TVEL (display commanded velocity) TVELA (display actual velocity) TVE (display velocity error) Velocity & Accel/Decel (e.g., V, A, AD)	$\frac{DMEPIT}{1000}$	$\frac{1000}{DMEPIT}$
DMTD (motor damping) DMTJ (motor/forcer mass) LDAMP (load damping)	$\frac{(2 \bullet \pi \bullet 1000)^2}{(DMEPIT)^2}$	$\frac{(DMEPIT)^2}{(2 \bullet \pi \bullet 1000)^2}$
DMTLIM (force limit) DMTSCL (force scaling)	$\frac{2 \bullet \pi \bullet 1000}{DMEPIT}$	$\frac{DMEPIT}{2 \bullet \pi \bullet 1000}$

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

DMODE Drive Control Mode

		Product	Rev
Type	Drive Configuration	GT	1.02
Syntax	<a_><!>DMODE<i>	GV	1.00
Units	i = control mode setting	GT6	1.50
Range	1-17 (see table below)	GV6	1.50
Default	GT: 6 GV: 2 GT6 & GV6: 12		
Response	DMODE: *DMODE6		
See Also	ANICDB, DCMDZ, DRES, SRSET		

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

DMODE	Mode	Description	GT	GV	GT6	GV6
1	RESERVED	----------	--	--	--	--
2**	Torque/Force Control (±10V) — **default for GV drive**	Allows direct command of rotary motor torque or linear motor force.	--	**X**	--	--
3	RESERVED	----------	--	--	--	--
4**	Velocity Control (±10V)	Allows direct command of the motor velocity.	**X**	**X**	--	--
5	RESERVED	----------	--	--	--	--
6	Position Control (step & direction) — **default for GT drive**	Allows direct command of the motor position.	**X**	**X**	--	--
7	Position Control (step & direction, with the direction command inverted)	See option 6 above. Allows for changing the sense of the direction signal.	**X**	**X**	--	--
8	Position Control (positive/negative)	Same as option 6, except separate step signals are supplied based on the desired direction.	**X**	**X**	--	--
9	Encoder Tracking	Allows a quadrature encoder signal to be used as the position command. The resulting position command is scaled by this ratio: ERES / DRES.	**X**	**X**	--	--
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 SRSET command.	--	**X**	--	**X**
12	Controller/Drive — **default for GT6 & GV6 drives**	Programmed motion using internal trajectory generator.	--	--	**X**	**X**
13	Autorun	Rotates the motor at 1 rps/mps. Current is reduced by 10%. Refer also to your drive's *Installation Guide*.	**X**	**X**	**X**	**X**
14	RESERVED	----------	--	--	--	--
15	Torque/Force Tuning Mode	10 Hz step input of 25% rated motor current for torque/force mode tuning. *	--	**X**	--	**X**
16	Velocity Tuning Mode	1 Hz ±2 rps/mps step velocity command for velocity mode tuning. *	--	**X**	--	**X**
17	Position Tuning Mode	1 Hz ± 1/4 rev/epitch step position command for position mode tuning. *	--	**X**	--	**X**

* Refer to the servo tuning procedures in the drive's *Hardware Installation Guide* for details.

** For ±10V operation (modes 2 or 4), you may need to zero the drive command offset to keep the motor from drifting initially. See the ANICDB and DCMDZ commands.

DMONAS Analog Monitor Output A — Scaling

		Product	Rev
Type	Outputs	GT	1.02
Syntax	<a_><!>DMONAS<i>	GV	1.00
Units	i = scale percentage (%)	GT6	1.50
Range	-2000 to 2000	GV6	1.50
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 ±10V.

Monitor waveform clipping will occur if DMONAS scaling results in an output greater than ±10V.

DMONAV Analog Monitor Output A — Variable

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

Product	Rev
GT	1.02
GV	1.00
GT6	1.50
GV6	1.50

The DMONAV command selects the function (or signal) that will be presented at the 8-bit DAC "Analog Out A" terminal (pin 21 on DRIVE I/O connector), referenced to analog ground (pin 25 on DRIVE I/O connector).

Number	Name	Description	GT	GV	GT6	GV6
0	Unused / Turn off output	No output function selected	X	X	X	X
1	Motor Temperature	(TMTEMP) Estimated motor winding temperature based on a second-order thermal model (see *Hardware Installation Guide* for details). Normalized to ±10 volts equals ±250 °C.		X		X
2	Drive Temperature	(TDTEMP) Measured internal drive temperature. Normalized to ±10 volts equals ±250 °C.	X	X	X	X
3	Position Error	This value is normalized as follows (value clips at ±1 rev or ±1 electrical pitch, regardless of the DMONAS setting): • Rotary motors: ±10V = ±1 rev (based on TPER ÷ ERES) • Linear motors: ±10V = ±1 epitch (based on TPER ÷ DMEPIT)		X		X
4	Velocity Setpoint	(TVEL) User commanded velocity. Normalized to ±10 volts equals ±200 revs/sec (rotary) or ±8.4 meters/sec (linear).	X	X	X	X
5	Estimated Actual Velocity	(TVELA) Output of velocity estimator, based on encoder data. Normalized to ±10 volts equals ±200 revs/sec (rotary) or ±8.4 meters/sec (linear).		X		X
6	Acceleration Setpoint	(TACC) User commanded acceleration.			X	X
7	Torque/Force Setpoint	(TTRQ) User commanded torque/force.		X		X
8	Actual Torque/Force	(TTRQA) Calculated torque/force based on measured motor current and motor Ke (DMTKE). Scaled as a % of DMTSCL.		X		X
9	Velocity Error	(TVE) Normalized to ±10 volts equals ±200 revs/sec (rotary) or ±8.4 meters/sec (linear).		X		X
10	Phase A Commanded Current	Instantaneous commanded current for phase A (Volts per Amp). *	X		X	
11	Phase A Actual Current	Instantaneous measured current for phase A (Volts per Amp). *	X	X	X	X
12	Phase B Commanded Current	Instantaneous commanded current for phase B (Volts per Amp). *	X		X	
13	Phase B Actual Current	Instantaneous measured current for phase B (Volts per Amp). *	X	X	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		X
17	d-axis Actual Current	Estimated direct-axis current from coordinate-conversion software. Based on measured phase currents. *		X		X
18	q-axis Commanded Current	Commanded quadrature-axis current. This current is proportional to commanded torque/force. *		X		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		X
20	Phase A Applied Voltage Duty Cycle	A voltage representation of the PWM duty cycle for Phase A.	X		X	
21	Phase B Applied Voltage Duty Cycle	A voltage representation of the PWM duty cycle for Phase B.	X		X	
22	RESERVED					
23	Position Setpoint	This value is normalized as follows (value clips at ±1 rev or ±1 electrical pitch, regardless of the DMONAS setting): • Rotary motors: ±10V = ±1 rev (based on TPC ÷ ERES) • Linear motors: ±10V = ±1 epitch (based on TPC ÷ DMEPIT)		X		X
24	Actual Position	This value is normalized as follows (value clips at ±1 rev or ±1 electrical pitch, regardless of the DMONAS setting): • Rotary motors: ±10V = ±1 rev (based on TPE ÷ ERES) • Linear motors: ±10V = ±1 epitch (based on TPE ÷ DMEPIT)		X		X

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

Stepper Drive	Nominal (V/A) Current Scaling
GTU5	1.423
GTU8	0.882
GTL5	1.432
GTL8	0.882

Servo Drive	Nominal (V/A) Current Scaling
GVL3	0.615
GVU3	0.615
GVU6	0.308
GVU12	0.205
GVH20	0.123

DMONBS Analog Monitor Output B — Scaling

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

Product	Rev
GT	1.02
GV	1.00
GT6	1.50
GV6	1.50

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

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

DMONBV Analog Monitor Output B — Variable

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

Product	Rev
GT	1.02
GV	1.00
GT6	1.50
GV6	1.50

The DMONBV command selects the function (or signal) that will be presented at the "Analog Out B" terminal (pin 22 on the DRIVE I/O connector), referenced to analog ground (pin 25 on the DRIVE I/O connector).

Number	Name	Description	GT	GV	GT6	GV6
0	Unused / Turn off output	No output function selected	X	X	X	X
1	Motor Temperature	(TMTEMP) Estimated motor winding temperature based on a second-order thermal model (see *Hardware Installation Guide* for details). Normalized to ±10 volts equals ±250 °C.		X		X
2	Drive Temperature	(TDTEMP) Measured internal drive temperature. Normalized to ±10 volts equals ±250 °C.	X	X	X	X
3	Position Error	This value is normalized as follows (value clips at ±1 rev or ±1 electrical pitch, regardless of the DMONAS setting): • Rotary motors: ±10V = ±1 rev (based on TPER ÷ ERES) • Linear motors: ±10V = ±1 epitch (based on TPER ÷ DMEPIT)		X		X
4	Velocity Setpoint	(TVEL) User commanded velocity. Normalized to ±10 volts equals ±200 revs/sec (rotary) or ±8.4 meters/sec (linear).	X	X	X	X
5	Estimated Actual Velocity	(TVELA) Output of velocity estimator, based on encoder data. Normalized to ±10 volts equals ±200 revs/sec (rotary) or ±8.4 meters/sec (linear).		X		X
6	Acceleration Setpoint	(TACC) User commanded acceleration.			X	X
7	Torque/Force Setpoint	(TTRQ) User commanded torque/force.		X		X
8	Actual Torque/Force	(TTRQA) Calculated torque/force based on measured motor current and motor Ke (DMTKE). Scaled as a % of DMTSCL.		X		X

Continued …

Number	Name	Description	GT	GV	GT6	GV6
9	Velocity Error	(TVE) Normalized to ±10 volts equals ±200 revs/sec (rotary) or ±8.4 meters/sec (linear).		X		X
10	Phase A Commanded Current	Instantaneous commanded current for phase A (Volts per Amp). *	X		X	
11	Phase A Actual Current	Instantaneous measured current for phase A (Volts per Amp). *	X	X	X	X
12	Phase B Commanded Current	Instantaneous commanded current for phase B (Volts per Amp). *	X		X	
13	Phase B Actual Current	Instantaneous measured current for phase B (Volts per Amp). *	X	X	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		X
17	d-axis Actual Current	Estimated direct-axis current from coordinate-conversion software. Based on measured phase currents. *		X		X
18	q-axis Commanded Current	Commanded quadrature-axis current. This current is proportional to commanded torque/force. *		X		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		X
20	Phase A Applied Voltage Duty Cycle	A voltage representation of the PWM duty cycle for Phase A.	X		X	
21	Phase B Applied Voltage Duty Cycle	A voltage representation of the PWM duty cycle for Phase B.	X		X	
22	RESERVED					
23	Position Setpoint	This value is normalized as follows (value clips at ±1 rev or ±1 electrical pitch, regardless of the DMONAS setting): • Rotary motors: ±10V = ±1 rev (based on TPC ÷ ERES) • Linear motors: ±10V = ±1 epitch (based on TPC ÷ DMEPIT)		X		X
24	Actual Position	This value is normalized as follows (value clips at ±1 rev or ±1 electrical pitch, regardless of the DMONAS setting): • Rotary motors: ±10V = ±1 rev (based on TPE ÷ ERES) • Linear motors: ±10V = ±1 epitch (based on TPE ÷ DMEPIT)		X		X

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

Stepper Drive	Nominal (V/A) Current Scaling
GTU5	1.423
GTU8	0.882
GTL5	1.432
GTL8	0.882

Servo Drive	Nominal (V/A) Current Scaling
GVL3	0.615
GVU3	0.615
GVU6	0.308
GVU12	0.205
GVH20	0.123

DMTAMB Motor Ambient Temperature

Type	Motor
Syntax	<a_><!>DMTAMB<r>
Units	r = Degrees Celsius
Range	-50.0 to 250.0 : ±0.1
Default	40.0
Response	DMTAMB: *DMTAMB40.0
See Also	DMTMAX, DMTRWC, DMTTCM, DMTTCW, TASX

Product	Rev
GT	n/a
GV	1.01
GT6	n/a
GV6	1.50

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 (DMTTCM), the motor winding time constant (DMTTCW), 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.

DMTD — Motor Damping

Type	Motor
Syntax	<a_><!>DMTD<r> (does not take effect until RESET or cycle power)
Units	Rotary motor: r = Nm/rad/sec Linear motor: r = N/meter/sec
Range	Rotary motor: 0.000000 to 0.010000 : ±0.000001 Linear motor: DMEPIT (electrical pitch) dependent
Default	0.000000
Response	DMTD: *DMTD0.002000
See Also	DMTR, LDAMP

Product	Rev
GT	n/a
GV	1.00
GT6	n/a
GV6	1.50

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

AUTO-SETUP: This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6) or Pocket Motion Planner (see page 11). 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
Syntax	<a_><!>DMTIC<r>
Units	r = Amps-RMS
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

Product	Rev
GT	1.02
GV	1.00
GT6	1.50
GV6	1.50

GV/GV6 Only: This command does not take effect until you cycle power to the drive, or issue a RESET.
GT/GT6 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) or Pocket Motion Planner (see page 11). 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.

GV: If DMTIC is set higher than the full-scale value calculated by DMTSCL (torque/force scale) the new DMTIC value will be ignored (but not overwritten), the status warning bit 28 in TASX will be set, a value of 400 will be written to the TCS register, and the full-scale value calculated from DMTSCL will be used internally.

GT: If DMTIC is set higher than the drive maximum current (TDICNT), the new DMTIC value will be ignored (but not overwritten), the status warning bit 28 in TASX will be set, a value of 400 will be written to the TCS register, and the maximum drive current will be used internally.

Example:
```
DMTIC5       ; Set the motor current to 5 amps RMS (equates to 7.07 amps peak)
```

DMTICD Continuous Current Derating

		Product	Rev
Type	Motor		
Syntax	<a_><!>DMTICD<i> (does not take effect until RESET or cycle power)	GT	n/a
Units	i = % derating at rated speed	GV	1.00
Range	0.00 to 100.00 : ±0.01	GT6	n/a
Default	0.00 (DMTICD of 0 results in no current derating)	GV6	1.50
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 command.

AUTO-SETUP: This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6) or Pocket Motion Planner (see page 11). 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%.

DMTIND Motor Inductance

		Product	Rev
Type	Motor		
Syntax	<a_><!>DMTIND<r> (does not take effect until RESET or cycle power)	GT	1.02
Units	r = mH	GV	n/a
Range	0.0 to 200.0 : ±0.1	GT6	1.50
Default	0.0 (DMTIND of 0 results in motor config. error)	GV6	n/a
Response	DMTIND *DMTIND10		
See Also	DMTLMN, DMTLMX, DMTR, TASX, TCS		

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

AUTO-SETUP: This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6) or Pocket Motion Planner (see page 11). 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 inductance for stepper motors only (servo motor inductance is set with DMTLMN and DMTLMX). The motor inductance entered is the motor inductance you measure across a 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 Configuration chapter of your drive's *Hardware Installation Guide*.

DMTIP — Peak Current

Type	Motor
Syntax	<a_><!>DMTIP<r>
Units	r = Amps-RMS
Range	0.00 to 128.00 : ±0.01
Default	0.00 (DMTIP of 0 results in motor config. warning)
Response	DMTIP: *DMTIP7.50
See Also	DMTIC, DMTICD, DMTLIM, DMTR, TASX, TCS, TDIMAX

Product	Rev
GT	n/a
GV	1.00
GT6	n/a
GV6	1.50

AUTO-SETUP: This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6) or Pocket Motion Planner (see page 11). 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 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	GV-L3	GV-U3	GV-U6	GV-U12	GV-H20
Maximum Current Rating	Amps *	7.5	7.5	15	30	50

* 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
Syntax	<a_><!>DMTJ<r> (does not take effect until RESET or cycle power)
Units	Rotary motor: r = $kgm^2 * 10^{-6}$ Linear motor: r = kg
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

Product	Rev
GT	1.02
GV	1.00
GT6	1.50
GV6	1.50

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

AUTO-SETUP: This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6) or Pocket Motion Planner (see page 11). 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

		Product	Rev
Type	Motor	GT	n/a
Syntax	<a_><!>DMTKE<r> (does not take effect until RESET or cycle power)	GV	1.00
Units	Rotary motor: r = volts (0-peak) / krpm Linear motor: r = volts/meter/sec	GT6	n/a
Range	Rotary motor: 0.0 to 200.0 : ±0.1 Linear motor: DMEPIT (electrical pitch) dependent	GV6	1.50
Default	0.0 (DMTKE of 0 results in motor config. error)		
Response	DMTKE: *DMTKE15.0		
See Also	DMONAV, DMONBV, DMTSCL, DMTR, TASX, TCS		

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

AUTO-SETUP: This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6) or Pocket Motion Planner (see page 11). 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

DMTLIM Torque/Force Limit

		Product	Rev
Type	System	GT	n/a
Syntax	<a_><!>DMTLIM<r>	GV	1.00
Units	Rotary motor: r = Nm Linear motor: r = N	GT6	n/a
Range	Rotary motor: 0.0 to 500.0 (motor/drive dependent) : ±0.1 Linear motor: DMEPIT (electrical pitch) dependent	GV6	1.50
Default	500.0		
Response	DMTLIM: *DMTLIM10.5		
See Also	DCLRLR, 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) or Pocket Motion Planner (see page 11). 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 (page 84) 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 Gemini 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, reducing the rate of motor heating and allowing more reaction time for the person tuning the system, and reducing the chances of damage to the mechanical system.

When the Gemini'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. 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 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):

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

Working with servo gains.

- Servo tuning process: refer to your Gemini drive's *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

		Product	Rev
Type	Motor		
Syntax	<a_><!>DMTLMN<r> (does not take effect until RESET or cycle power)	GT	n/a
Units	r = mH	GV	1.00
Range	0.1 to 200.0 (motor dependent) : ±0.1	GT6	n/a
Default	0.0 (DMTLMN of 0 results in motor config. error)	GV6	1.50
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 command.

AUTO-SETUP: This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6) or Pocket Motion Planner (see page 11). 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.

DMTLMX Maximum Motor Inductance

		Product	Rev
Type	Motor	GT	n/a
Syntax	<a_><!>DMTLMX<r> (does not take effect until RESET or cycle power)	GV	1.00
Units	r = mH	GT6	n/a
Range	0.1 to 200.0 (motor dependent) : ±0.1	GV6	1.50
Default	0.0 (DMTLMX of 0 results in motor config. error)		
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 command.

AUTO-SETUP: This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6) or Pocket Motion Planner (see page 11). 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

		Product	Rev
Type	Motor	GT	n/a
Syntax	<a_><!>DMTMAX<r>	GV	1.01
Units	r = Degrees Celsius	GT6	n/a
Range	-50.0 to 250.0 : ±0.1	GV6	1.50
Default	125.0		
Response	DMTMAX: *DMTMAX125.0		
See Also	DMTAMB, DMTIC, DMTRWC, DMTTCM, DMTTCW, TASX		

AUTO-SETUP: This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6) or Pocket Motion Planner (see page 11). 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 (DMTTCM), the motor winding time constant (DMTTCW), 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.

DMTR — Identify (and Load) Motor

		Product	Rev
Type	Drive Configuration	GT	1.02
Syntax	<a_><!>DMTR<i>	GV	1.00
Units	i = Parker motor identification number	GT6	1.50
Range	0-2000	GV6	1.50
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) or Pocket Motion Planner (see page 11).

When you select a specific Parker motor using the Motion Planner or Pocket 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 Gemini drive, you must cycle drive power or issue a RESET 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.

Stepper Motor Data Parameters	
DIGNA	Current Loop Gain A
DIGNB	Current Loop Gain B
DIGNC	Current Loop Gain C
DIGND	Current Loop Gain D
DMTIC	Continuous Current
DMTIND	Motor Inductance
DMTJ	Motor Rotor Inertia
DMTRES	Motor Winding Resistance
DMTSTT	Motor Static Torque
DMVLIM	Velocity Limit
DMVSCL	Velocity Scaling
DPOLE	Number of Motor Pole Pairs

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

Although these command values are auto-configured when you select a Parker motor (using the setup tool in Motion Planner or Pocket 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 Gemini 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 Pocket Motion Planner (or configure each motor parameter command with a value other than zero – using a Gemini terminal emulator), download the resulting configuration information and then cycle power or issue the RESET 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 and Pocket Motion Planner to configure the Gemini 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 an updated 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). If you are using Pocket Motion Planner, transfer the updated file to your hand-held PC and copy it to the \My Documents\Gemini directory.

The DMTR values for all available Parker motors are listed in the GEM_motors.mtr file.

DMTRES Motor Winding Resistance

		Product	Rev
Type	Motor		
Syntax	<a_><!>DMTRES<r> (does not take effect until RESET or cycle power)	GT	1.02
Units	r = Ohm	GV	1.00
Range	0.00 to 50.00 : ±0.01	GT6	1.50
Default	0.00 (DMTRES of 0 results in motor config. error)	GV6	1.50
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 command.

AUTO-SETUP: This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6) or Pocket Motion Planner (see page 11). 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. The resistance value is the resistance of one motor phase as measured at 25 °C at the drive end of the motor cable (motor cable included). For steppers, this would be A+ to A-; for servos, U to V.

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

		Product	Rev
Type	Motor		
Syntax	<a_><!>DMTRWC<r> (does not take effect until RESET or cycle power)	GT	n/a
Units	r = Degrees Celsius/Watt (°C/W)	GV	1.00
Range	0.00 to 50.00 : ±0.01	GT6	n/a
Default	0.00	GV6	1.50
Response	DMTRWC: *DMTRWC23.60		
See Also	DMTR, DMTTCM, DMTTCW, TASX, TCS		

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

AUTO-SETUP: This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6) or Pocket Motion Planner (see page 11). 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.

DMTSCL Torque/Force Scaling

		Product	Rev
Type	Drive Configuration	GT	n/a
Syntax	<a_><!>DMTSCL<r>	GV	1.00
Units	Rotary motor: r = Nm Linear motor: r = N	GT6	n/a
Range	Rotary motor: 0.0 to 500.0 (motor/drive dependent) : ±0.1 Linear motor: DMEPIT (electrical pitch) dependent	GV6	1.50
Default	0		
Response	DMTSCL: *DMTSCL20.0		
See Also	DMEPIT, DMODE, 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) or Pocket Motion Planner (see page 11). 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.)

GV and GV6: The DMTSCL command scales the torque/force (TTRQ and TTRQA).

GV only (n/a to GV6): DMTSCL is also applied when the drive is set to DMODE2 (±10V torque/force).

The DMTSCL command scales the torque/force command input. It sets the full-scale torque/force that will be produced from a 10-volt input command. It controls the gain applied to the input. This can be used to scale the input to match application needs. For example, if a torque/force sensor produces 2 volts per Newton-meter, the drive could be scaled to match this by using DMTSCL5 — this sets 10V = 5 Nm (0.5 Nm/Volt).

DMTLIM may limit torque/force to less than this full-scale value. Note that changing this value affects many gain terms including position loop tuning values; tuning may need to be repeated after changing DMTSCL. It should ideally be set before tuning the system.

NOTE: To configure the drive in torque/force mode so that a 10-volt torque/force command will produce the rated peak current of the drive (without reference to motor parameters) enter for DMTSCL the result of the following calculation:

Rotary motors: $DMTSCL = TDIMAX * \frac{3\sqrt{3}}{200\pi} * DMTKE$ (for V*/KRPM)

Linear motors: $DMTSCL = TDIMAX * \frac{\sqrt{6}}{8\pi} * \frac{DMEPIT}{1000} * DMTKE$ (for V*/meter/sec)

* peak value

DMTSTT Motor Static Torque

		Product	Rev
Type	Drive Configuration	GT	1.02
Syntax	<a_><!>DMTSTT<r> (does not take effect until RESET or cycle power)	GV	n/a
Units	r = oz-in	GT6	1.50
Range	0.0 to 5000.0 : ±0.02	GV6	n/a
Default	0.0 (DMTSTT of 0 results in motor config. error)		
Response	DMTSTT *DMTSTT410.0		
See Also	DMTR, TASX, TCS		

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

AUTO-SETUP: This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6) or Pocket Motion Planner (see page 11). 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

		Product	Rev
Type	Motor	GT	n/a
Syntax	<a_><!>DMTTCM<r> (does not take effect until RESET or cycle power)	GV	1.00
Units	r = minutes	GT6	n/a
Range	0.0 to 300.0 : ±0.1	GV6	1.50
Default	0.0		
Response	DMTTCM: *DMTTCM30.4		
See Also	DMTR, DMTTCW		

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

AUTO-SETUP: This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6) or Pocket Motion Planner (see page 11). 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 ¼" 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

		Product	Rev
Type	Motor	GT	n/a
Syntax	<a_><!>DMTTCW<r> (does not take effect until RESET or cycle power)	GV	1.00
Units	r = minutes	GT6	n/a
Range	0.00 to 100.00 : ±0.01	GV6	1.50
Default	0.00		
Response	DMTTCW: *DMTTCW28.40		
See Also	DMTR, DMTTCM, DMTRWC		

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

AUTO-SETUP: This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6) or Pocket Motion Planner (see page 11). 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
Syntax	<a_><!>DMTW<r> (does not take effect until RESET or cycle power)
Units	Rotary motor: r = revs/sec Linear motor: r = meters/sec
Range	Rotary motor: 0.0 to 200.0 : ±0.1 Linear motor: DMEPIT (electrical pitch) dependent
Default	0.0 (DMTW of 0 results in motor config. error)
Response	DMTW: *DMTW150.0
See Also	DMEPIT, DMTICD, DMTR, TASX, TCS

Product	Rev
GT	n/a
GV	1.00
GT6	n/a
GV6	1.50

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

AUTO-SETUP: This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6) or Pocket Motion Planner (see page 11). 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/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.

DMVLIM — Velocity Limit

Type	System
Syntax	<a_><!>DMVLIM<r>
Units	Rotary motor: r = revs/sec Linear motor: r = meters/sec
Range	Rotary motor: GT/GT6: 0.000000 to 60.000000 : ±0.000001 GV/GV6: 0.000000 to 200.000000 : ±0.000001 Linear motor: DMEPIT (electrical pitch) dependent
Default	GT/GT6: 50.000000 GV/GV6: 200.000000
Response	DMVLIM: *DMVLIM50.000000
See Also	DCLRLR, DMODE, DMTR, DMVSCL, DVBW, SGVRAT, TASX, TGAIN, TSGSET

Product	Rev
GT	1.02
GV	1.00
GT6	1.50
GV6	1.50

AUTO-SETUP: This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6) or Pocket Motion Planner (see page 11). 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 (page 84) 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 velocity mode (DMODE4), position modes (DMODE6, DMODE7, DMODE8 or DMODE9), or controller/drive mode (DMODE12), DMVLIM clamps the command to the specified value.

GV only: In torque/force mode (DMODE2), this command can be used to set a velocity override limit. This controls system velocity in the event that load friction and commanded torque/force combined would result in excessive velocity.

If the velocity demand from a controller or the internal Gemini control loops exceeds the limits set by DMVLIM, then TASX bit 18 will be set. The GV/GV6 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. Bit 18 stays latched until a DCLRLR is issued or a drive reset occurs.

GT6 & GV6 only: Changes to the DMVLIM command are not allowed while motion is in progress.

Working with servo gains.

- Servo tuning process: refer to your Gemini drive's *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.

DMVSCL Velocity Scaling

Type	Drive Configuration
Syntax	<a_><!>DMVSCL<r>
Units	Rotary motor: r = revs/sec Linear motor: r = meters/sec
Range	Rotary motor: GT: 0.000000 to 60.000000 : ±0.000001 GV: 0.000000 to 200.000000 : ±0.000001 Linear motor: DMEPIT (electrical pitch) dependent
Default	GT: 50.000000 GV: 0.000000
Response	DMVSCL: *DMVSCL40.000000
See Also	DMODE, DMVLIM, DVBW, LDAMP, SGVRAT

Product	Rev
GT	1.02
GV	1.00
GT6	n/a
GV6	n/a

AUTO-SETUP: This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6) or Pocket Motion Planner (see page 11). 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 DMVSCL command scales the velocity command input signal only in the ±10V velocity mode (DMODE4). This sets the full-scale value of the input by changing the gain applied to it. This functions like a "tach gain" control, changing the velocity that will result from a given input voltage — for example to 50 rps per 10 volts (5rps/volt, or 300RPM/volt).

NOTE

The input Analog-to-Digital converter is 12 bits; therefore, the "effective" velocity resolution is calculated as follows:

$$"effective"\ resolution = \frac{DMVSCL}{2048}$$

Example: If DMVSCL is set to 50.000000 revs/sec, the effective internal resolution is ± ~.024 rps (1.46 rpm).

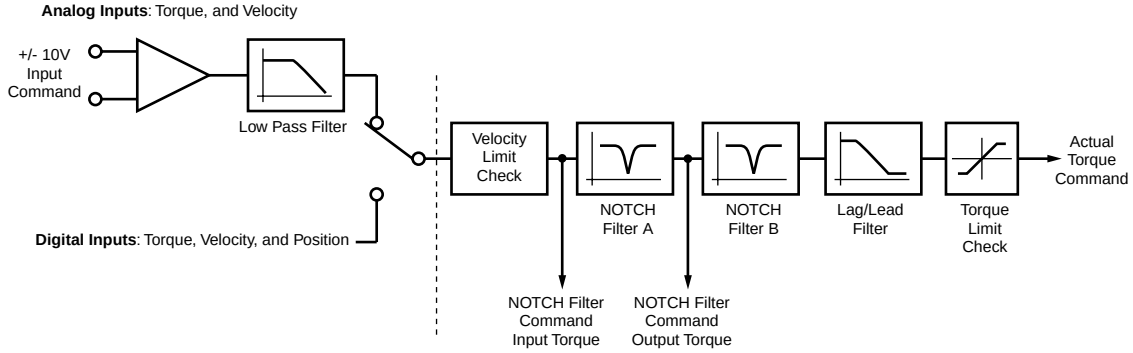
GV: Changing the DMVSCL value will affect overall position loop response when used with the drive operating in the ±10V Velocity Mode (DMODE4), so it should ideally be set before tuning.

DNOTAD Notch Filter A Depth

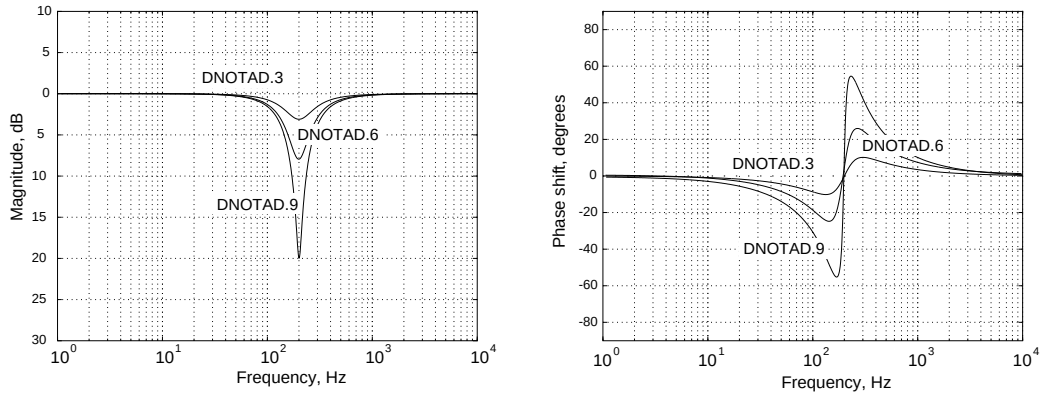
		Product	Rev
Type	Tuning	GT	n/a
Syntax	<a_><!>DNOTAD<i>	GV	1.01
Units	n/a	GT6	n/a
Range	0.0000 - 1.0000	GV6	1.50
Default	0.0000 (depth is zero)		
Response	DNOTAD: *DNOTAD.5		
See Also	DMODE, DNOTAF, DNOTAQ, DNOTBD, DNOTBF, DNOTBQ, DNOTLD, DNOTLG, DPBW, DVBW, TGAIN, TSGSET		

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 Torque/Force Tuning mode (DMODE15).

CAUTION: In velocity mode (DMODE4) and the position modes (DMODE5,6,7,8 or 9), setting the notch filter’s center frequency (DNOTAF) lower than either the velocity loop bandwidth (DVBW) or position loop bandwidth (DPBW), respectively, may cause unstable behavior.

Working with servo gains.

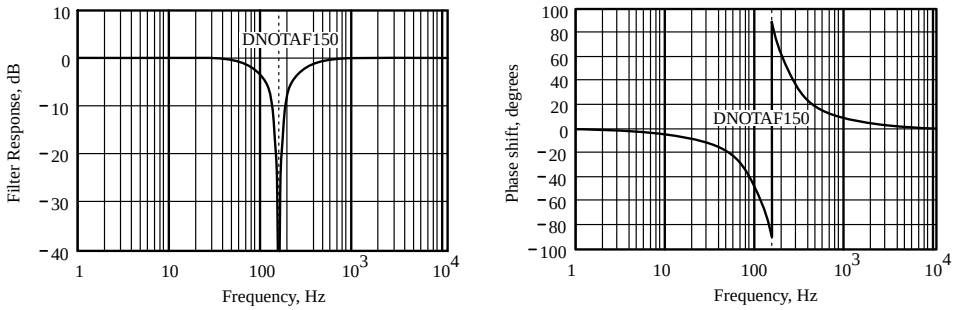
- Servo tuning process: refer to your Gemini drive’s *Hardware Installation Guide*.
- Check the values of all active gains (DNOTAD is one of many servo gains): use TGAIN.
- Creating and invoking gain sets: see SGSET, SGENB, TGAIN, TSGSET.

DNOTAF Notch Filter A Frequency

		Product	Rev
Type	Tuning	GT	n/a
Syntax	<a_><!>DNOTAF<i>	GV	1.00
Units	i = Hz	GT6	n/a
Range	0 (disable), or 60-1000	GV6	1.50
Default	0 (filter is disabled)		
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 internal commanded torque/force vs. the user 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).

Caution: In velocity mode (DMODE4) and the position modes (DMODE5,6,7,8 or 9), setting the notch filter's center frequency (DNOTAF) lower than either the velocity loop bandwidth (DVBW) or position loop bandwidth (DPBW), respectively, will cause unpredictable results.

Working with servo gains.

- Servo tuning process: refer to your Gemini drive's *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

		Product	Rev
Type	Tuning	GT	n/a
Syntax	<a_><!>DNOTAQ<r>	GV	1.00
Units	r = quality factor	GT6	n/a
Range	0.5 to 2.5	GV6	1.50
Default	1		
Response	DNOTAQ: *DNOTAQ1.5		
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 your Gemini drive's *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

		Product	Rev
Type	Tuning		
Syntax	<a_><!>DNOTBD<i>	GT	n/a
Units	n/a	GV	1.01
Range	0.0000 - 1.0000	GT6	n/a
Default	0.0000 (depth is zero)	GV6	1.50
Response	DNOTBD: *DNOTBD.5		
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 your Gemini drive's *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

		Product	Rev
Type	Tuning		
Syntax	<a_><!>DNOTBF<i>	GT	n/a
Units	i = Hz	GV	1.00
Range	0 (disable), or 60-1000	GT6	n/a
Default	0 (filter is disabled)	GV6	1.50
Response	DNOTBF: *DNOTBF200		
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 your Gemini drive's *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

		Product	Rev
Type	Tuning		
Syntax	<a_><!>DNOTBQ<r>	GT	n/a
Units	r = quality factor	GV	1.00
Range	0.5 to 2.5	GT6	n/a
Default	1	GV6	1.50
Response	DNOTBQ: *DNOTBQ1.5		
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 your Gemini drive's *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

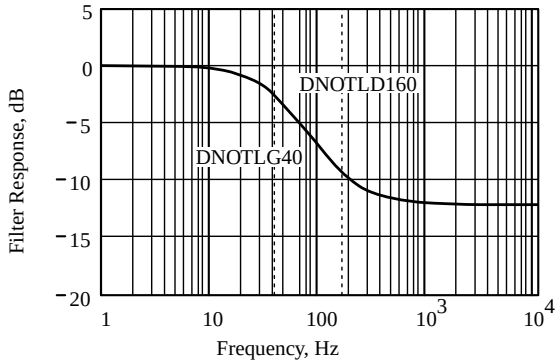
		Product	Rev
Type	Tuning	GT	n/a
Syntax	<a_><!>DNOTLD<i>	GV	1.00
Units	i = Hz	GT6	n/a
Range	0 (disable), or 80-1000	GV6	1.50
Default	0 (filter is disabled)		
Response	DNOTLD: *DNOTLD200		
See Also	DNOTAD, DNOTAF, DNOTAQ, DNOTBD, DNOTBF, DNOTLG, TASX, TGAIN, TSGSET		

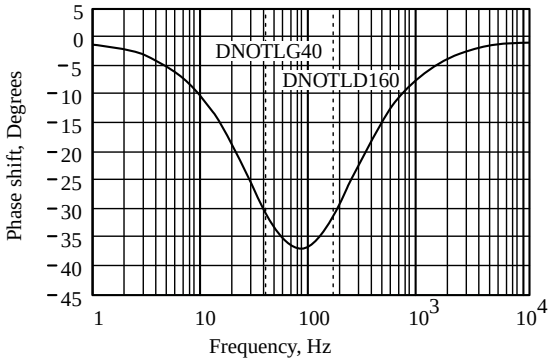
The DNOTLD command sets the break frequency of the lead filter. This filter cannot be used alone, but must be used in conjunction with the DNOTLG lag filter. The DNOTLG lag filter must be configured before the DNOTLD lead filter is configured.

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 your Gemini drive's *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

		Product	Rev
Type	Tuning	GT	n/a
Syntax	<a_><!>DNOTLG<i>	GV	1.00
Units	i = Hz	GT6	n/a
Range	0 (disable), or 20-1000	GV6	1.50
Default	0 (filter is disabled)		
Response	DNOTLG: *DNOTLG400		
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 your Gemini drive's *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.

DPBW Position Loop Bandwidth

		Product	Rev
Type	Tuning	GT	n/a
Syntax	<a_><!>DPBW<r>	GV	1.00
Units	r = Hz	GT6	n/a
Range	1.00 to 100.00 : ±0.01	GV6	1.50
Default	5.00		
Response	DPBW: *DPBW10.00		
See Also	DIBW, DMTR, DVBW, 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 5) or Pocket Motion Planner (see page 9). 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 (page 84) 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 your Gemini drive's *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.

DPHBAL Phase Balance

Type	Drive Configuration
Syntax	<a_><!>DPHBAL<r>
Units	r = % of (peak) nominal commanded current
Range	90.0 to 110.0 : ±0.1
Default	100.0
Response	DPHBAL: *DPHBAL100.0
See Also	DPHOFA, DPHOFB, DWAVEF

Product	Rev
GT	1.02
GV	n/a
GT6	1.50
GV6	n/a

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 your drive's *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
Syntax	<a_><!>DPHOFA<r>
Units	r = no units
Range	-10.000 to 10.000 : ±0.001
Default	0.000
Response	DPHOFA: *DPHOFA5.000
See Also	DPHBAL, DPHOFB, DWAVEF

Product	Rev
GT	1.02
GV	n/a
GT6	1.50
GV6	n/a

The DPHOFA command adjusts the current offset of phase A.

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

DPHOFB Phase B Current Offset

Type	Drive Configuration
Syntax	<a_><!>DPHOFB<r>
Units	r = no units
Range	-10.000 to 10.000 : ±0.001
Default	0.000
Response	DPHOFB: *DPHOFB5.000
See Also	DPHBAL, DPHOFA, DWAVEF

Product	Rev
GT	1.02
GV	n/a
GT6	1.50
GV6	n/a

The DPHOFB command adjusts the current offset of phase B.

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

DPOLE — Number or Motor Pole Pairs

		Product	Rev
Type	Motor	GT	1.02
Syntax	<a_><!>DPOLE<i> (does not take effect until RESET or cycle power)	GV	1.00
Units	i = pole pairs	GT6	1.50
Range	1-200	GV6	1.50
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 command.

AUTO-SETUP: This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6) or Pocket Motion Planner (see page 11). 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 (ω_e) is related to the mechanical speed (ω_m) of the motor by the pole pairs (P/2). 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.

DPWM — Drive PWM Frequency

		Product	Rev
Type	Drive Configuration	GT	n/a
Syntax	<a_>DPWM <i> (does not take effect until RESET or cycle power)	GV	1.01
Units	KHz	GT6	n/a
Range	0, 8, 16, 20 or 40	GV6	1.50
Default	0 (use the drive's default frequency): GV-U3, GV-U6, GV-U12 and GV-H20: 8 KHz GV-L3: 40 KHz		
Response	DPWM8		
See Also			

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

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

Note: All currents are peak of the sine wave values.

DRES — Drive Resolution

		Product	Rev
Type	Drive Configuration	GT	1.02
Syntax	<a_><!>DRES<i> (does not take effect until RESET or cycle power)	GV	1.00
Units	Rotary motor: i = counts/rev Linear motor: i = counts/electrical pitch	GT6	1.50
Range	GT/GT6: 200-128000; GV: 200-1024000	GV6	n/a
Default	GT/GT6: 25000; GV: 4000		
Response	DRES: *DRES25000		
See Also	DMEPIT, DMODE, ERES		

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

GT/GT6: The drive resolution represents counts/rev.

GV: The DRES command is used by the GV drive only when it is configured for step and direction input (DMODE6, DMODE7, DMODE8, or DMODE9). The drive resolution for GV rotary motors is represented as counts/rev and for GV linear motors as counts per electrical pitch. For example, a GV drive configured for a rotary motor, operating in DMODE6 and connected to a step and direction source with a resolution of 50,000 counts/rev, would have the drive resolution set to DRES50000. **AUTO-SETUP**: DRES is automatically configured to the same value as ERES, according to the Parker motor selected with the configuration utility in Pocket Motion Planner or Motion Planner.

If the drive is operating in the Encoder Tracking Mode (DMODE9), the resulting position command is scaled by this ratio: encoder resolution (ERES) / drive resolution (DRES).

DRIVE — Drive Enable

		Product	Rev
Type	Drive Configuration	GT	1.02
Syntax	<a_><!>DRIVE<b>	GV	1.00
Units	b = enable bit	GT6	1.50
Range	0 (shutdown the drive) or 1 (enable the drive)	GV6	1.50
Default	1 (enabled - if hardware enable interlock is closed)		
Response	DRIVE *DRIVE1		
See Also	ESK, FLTDSB, FLTSTP, TAS, TASX, TER, TPC, TPE, TPER		

The DRIVE command allows you to enable or disable (shut down) the drive.

If the hardware enable input interlock (pin 1 to pin 2 on the DRIVE I/O connector) is closed on power up, the drive is automatically enabled (generates a DRIVE1 command). To disable the drive, either issue the DRIVE0 command or open the hardware enable interlock.

Conversely, if the hardware enable input interlock is open on power up, the drive is disabled (DRIVE0). To enable the drive, close the hardware enable interlock.

GV: Issuing a DRIVE1 command from a DRIVE0 condition will set the position error to zero (TPER = 0).

All of these "Fault Conditions" automatically cause a shut down (DRIVE0), as well as activate the "fault" output (output #2) 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.
- (GT & GV only) If operating in the FLTSTP1 mode and the drive received incoming indexer pulses during power up or drive enable (DRIVE1).
- If operating in the FLTDSB1 mode and the drive received a DRIVE0 command or the hardware enable input interlock was opened.
- (GT only) If operating in the ESK1 mode and stall is detected.

DSTALL — Stall Detect Sensitivity

Type	Drive Configuration
Syntax	<a_><!>DSTALL<i>
Units	i = sensitivity level selection
Range	0-50 (0 disables the stall detect function)
Default	0 (disabled)
Response	DSTALL: *DSTALL20
See Also	ERROR, ESK, KDRIVE, TAS, TASX, TER

Product	Rev
GT	1.02
GV	n/a
GT6	1.50
GV6	n/a

The DSTALL command sets the sensitivity of the stall detection function. A setting of 0 disables the 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. Stalls are reported in TASX bit #17. GT6: If error-checking bit #1 is enabled (ERROR1), a stall causes the Gemini to branch to the ERRORP program.

If the Fault on Stall mode is enabled (ESK1), when the Gemini drive detects a stall, it will:

- Immediately stop pulses from being sent to the motor
- Set axis status (TAS) bit #12 and error status (TER) bit #1
- GT only: Fault the drive (disable the drive with DRIVE0, and activate output #2 and the relay output)
- GT6 only: Execute a Kill (!K). If Disable Drive on Kill is enabled (KDRIVE1) it disables the drive.

DVBW — Velocity Loop Bandwidth

Type	Tuning
Syntax	<a_><!>DVBW<i>
Units	i = Hz
Range	0-500
Default	0
Response	DVBW: *DVBW200
See Also	DIBW, DMTR, SGVRAT, TGAIN, TSGSET

Product	Rev
GT	n/a
GV	1.00
GT6	n/a
GV6	1.50

AUTO-SETUP: This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6) or Pocket Motion Planner (see page 11). 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 DVBW command sets the bandwidth of the velocity loop. If the velocity loop mode (DMODE4) is used, its bandwidth must be well below that of the current loop (DIBW) for non-interacting performance. If the bandwidth of the current loop is less than 10 times that of the velocity loop, current loop performance may limit velocity loop performance.

Working with servo gains.

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

DWAVEF — Waveform

Type	Drive Configuration
Syntax	<a_><!>DWAVEF<r>
Units	r = % 3rd harmonic
Range	-20.00 to 10.00 : ±0.01
Default	-4.00 (-4% 3rd harmonic injection)
Response	DWAVEF *DWAVEF-4
See Also	DPHBAL, DPHOFA, DPHOFB

Product	Rev
GT	1.02
GV	n/a
GT6	1.50
GV6	n/a

The DWAVEF command sets the percentage of 3rd harmonic included in the commanded motor current waveform. A procedure for configuring the Waveform is provided in your drive's *Hardware Installation Guide*; refer to the section entitled "Motor Control Settings".

E — Enable Communication

Type	Communication Interface
Syntax	<a_><!>E<b>
Units	b = enable bit
Range	0 (serial communication off) or 1 (serial communication on)
Default	1
Response	n/a
See Also	ADDR, ECHO, XONOFF

Product	Rev
GT	1.02
GV	1.00
GT6	1.50
GV6	1.50

The E command allows you to enable and disable serial communication on the Gemini 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.

ECHO — Communication Echo Enable

Type	Communication Interface
Syntax	<a_><!>ECHO<b>
Units	b = enable bit
Range	0 (disable) or 1 (enable)
Default	1
Response	ECHO: *ECHO1
See Also	EOL, EOT, ERRLVL, TSS

Product	Rev
GT	1.02
GV	1.00
GT6	1.50
GV6	1.50

The ECHO command enables/disables command echo.

- If using an RS-232 daisy-chain, ENABLE echo.
- If using an RS-485 multi-drop, DISABLE echo.

Consult the *Hardware Installation Guide* for RS-232 and RS-485 wiring instructions.

NOTE: The ECHO command has no obvious effect in the Motion Planner and Pocket Motion Planner terminal programs. In these programs, you will always see the characters that you type echoed on the screen. Only after you send a command delimiter (colon, carriage return or line feed) will a command line be sent to the Gemini drive.

ELSE — Else Condition of IF Statement

Type	Program Flow Control
Syntax	<a_><!>ELSE
Units	n/a
Range	n/a
Default	n/a
Response	n/a
See Also	IF, NIF

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

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 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
```

END — End Program/Profile Definition

		Product	Rev
Type	Program Definition	GT	n/a
Syntax	<a_><!>END	GV	n/a
Units	n/a	GT6	1.50
Range	n/a	GV6	1.50
Default	n/a		
Response	n/a		
See Also	DEF PROF, DEF PROG, DEL PROF, DEF PROG, GOSUB, PRUN, RUN		

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

Example (for programs):

```
DEL PROG3     ; Delete program number 3
DEF PROG3     ; Begin definition of program number 3
GO1           ; Initiate motion
END           ; End program definition
RUN PROG3     ; Execute program number 3
```

EOL — End of Line Terminating Characters

		Product	Rev
Type	Communication Interface	GT	1.02
Syntax	<!>EOL<i>,<i>,<i>	GV	1.00
Units	n/a	GT6	1.50
Range	i = 0-255	GV6	1.50
Default	13,10,0		
Response	EOL: *EOL13,10,0		
See Also	BOT, EOT, ERRLVL, XONOFF		

The End of Line Terminating Characters (EOL) command designates the characters to be placed at the end of each line, but not the last line, in a multi-line response. The last line of a multi-line response has the EOT characters. Up to 3 characters can be placed at the end of each line. The characters are designated with their ASCII equivalent (no character that has a value of zero [Ø] will be output). For example, a carriage return is ASCII 13, a line feed is ASCII 10, and no terminating character is designated with a zero. (For example, EOL13,0,0 places a carriage return after each line of a response.) If the first field is a zero, the drive will only accept zeros from the other two fields.

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
Ctrl-Z	26

For a more complete list of ASCII Equivalents, refer to the ASCII Table on page 193.

NOTE: This command is intended to be used only during live terminal communication with the drive. Do not download this command to the drive, or place it in a program.

EOT — End of Transmission Characters

		Product	Rev
Type	Communication Interface	GT	1.02
Syntax	<!>EOT<i>,<i>,<i>	GV	1.00
Units	n/a	GT6	1.50
Range	i = 0-255	GV6	1.50
Default	13,0,0		
Response	EOT: *EOT13,0,0		
See Also	BOT, EOL, ERRLVL		

The End of Transmission Terminating Characters (EOT) command designates the characters to be placed at the end of every response. Up to 3 characters can be placed after the last line of a multi-line response, or after all single-line response. The characters are designated with their ASCII equivalent (no character that has a value of zero [Ø] will be output). For example, a carriage return is ASCII 13, a line feed is ASCII 10,

a Ctrl-Z is ASCII 26, and no terminating character is designated with a zero. If the first field is a zero, the drive will only accept zeros from the other two fields.

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
Ctrl-Z	26

For a more complete list of ASCII Equivalents, refer to the ASCII Table on page 193.

NOTE: This command is intended to be used only during live terminal communication with the drive. Do not download this command to the drive, or place it in a program.

Example:

```
EOT13,10,26   ; Place a carriage return, line feed, and Ctrl-Z after the last line
              ; of a multi-line response, and after all single-line responses
```

ERASE — Erase All Programs and Profiles

Type	Subroutine or Program Definition
Syntax	<a_><!>ERASE
Units	n/a
Range	n/a
Default	n/a
Response	n/a
See Also	DEF PROF, DEF PROG, DEL PROF, DEL PROG, RFS

Product	Rev
GT	n/a
GV	n/a
GT6	1.70
GV6	1.70

The Erase All Programs and Profiles (ERASE) command deletes all programs and profiles created with the DEF command. If you do not want to erase all of the programs and all of the profiles, you can use the DEL command to selectively delete programs or profiles. The RFS command will erase all programs and reset all values to factory defaults.

ERES — Encoder/Resolver Resolution

Type	Encoder Configuration
Syntax	<a_><!>ERES<i> (does not take effect until RESET or cycle power)
Units	Rotary motor: i = counts/rev Linear motor: i = counts/electrical pitch
Range	200-1024000
Default	4000 (If SFB4, the default is 4096 counts/rev)
Response	ERES: *ERES4000
See Also	DMEPIT, DMTR, DRES, ORES, SFB, SMPER, TPE, TPER

Product	Rev
GT	n/a
GV	1.00
GT6	n/a
GV6	1.50

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

AUTO-SETUP for GV Drives only: This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6) or Pocket Motion Planner (see page 11). 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 (page 84) for a list of auto-configured commands.

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).

If you are using resolver feedback (SFB4), the ERES value is automatically set to 4096 counts/rev (post quad).

Resolutions for Parker-supplied encoders – SFB1 (refer also to the *Gemini Motor Reference Manual*):

- BE Series Servo Motors ..... BExxxxJ-xxxx: ERES8000
- G Series Servo Motors ..... GxxxxK-xxxx: ERES8192
- SM, N or J Series Servo Motors ..... SM/N/JxxxxD-xxxx: ERES2000
  SM/N/JxxxxE-xxxx: ERES4000

- 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).

$$ERES = \frac{DMEPIT\ \ (mm)}{Encoder\_resolution\ \ (mm/count)}$$

Example: Linear encoder resolution (post quad) is 1 µm and the electrical pitch is 42mm (DMEPIT42). ERES is calculated as:

$$ERES = \frac{42\ \ (mm)}{1 \cdot 10^{-3}\ (mm/count)} = 42000$$

Resolutions for Parker-supplied resolvers – SFB4: ERES4096 (BE, SM, N, J and G Series motors)

ERRBAD Error Prompt

		Product	Rev
Type	Communication Interface	GT	1.02
Syntax	<!>ERRBAD<i>,<i>,<i>,<i>	GV	1.00
Units	n/a	GT6	1.50
Range	i = 0-255	GV6	1.50
Default	13,10,63,32		
Response	ERRBAD: *ERRBAD13,10,63,32		
See Also	BOT, EOT, ERRDEF, ERRLVL, ERROK		

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. If the first field is a zero, the drive will only accept zeros from the other two fields. For a more complete list of ASCII equivalents, refer to the ASCII Table on page 193.

NOTE: This command is intended to be used only during live terminal communication with the drive. Do not download this command to the drive, or place it in a program.

ERRDEF Program Definition Prompt

		Product	Rev
Type	Communication Interface	GT	1.02
Syntax	<!>ERRDEF<i>,<i>,<i>,<i>	GV	1.00
Units	n/a	GT6	1.50
Range	i = 0-255	GV6	1.50
Default	13,10,45,32		
Response	ERRDEF: *ERRDEF13,10,45,32		
See Also	BOT, DEF PROG, DEF PROF, END, EOT, ERRBAD, ERRLVL, ERROK		

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. If the first field is a zero, the drive will only accept zeros from the other two fields. For a more complete list of ASCII equivalents, refer to the ASCII Table on page 193.

NOTE: This command is intended to be used only during live terminal communication with the drive. Do not download this command to the drive, or place it in a program.

ERRLVL Error Detection Level

Type	Error Handling
Syntax	<a_><!>ERRLVL<i>
Units	i = error level setting
Range	i = 0, 2, 3 or 4
Default	4
Response	ERRLVL: *ERRLVL4
See Also	BOT, EOL, EOT, ERRBAD, ERROK

Product	Rev
GT	1.02
GV	1.00
GT6	1.50
GV6	1.50

In each command description, there is a "Response" field that identifies the potential response(s) given when the command is executed. For example, configuration 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 (error detection level 4, selected with the ERRLVL4 command), the response characters are transmitted in the following order:

1. BOT (beginning of transmission) characters. The default setting is for no characters to be transmitted.
2. Asterisk (*) immediately preceding the response text.
3. Response text (root text of the response).
4. EOT (end of transmission) characters. The default EOT transmission is a carriage return.
   If the response comprises more than one line (e.g., TERRLG 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. If certain error conditions are detected, an error message is transmitted. Refer to page 15 for a list of possible error messages and their causes.
6. If an error is detected, the ERRBAD characters are transmitted. The default ERRBAD transmission characters are (in order of transmission): carriage return, line feed, "?" prompt, and space.
   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.

Although the root text identified in each command's "Response" field is always transmitted, the ERRLVL command allows you to include or exclude other 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).
- Asterisk (*), transmitted at the beginning of each line of the Response (e.g., *ERRLVL4).
- 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.
- Error message (see page 15).

	Characters when no error is detected			Characters when error is detected		
	BOT/EOT/EOL	*	ERROK	BOT/EOT/EOL	ERRBAD	Error Msg.
ERRLVL4	√	√	√	√	√	√
ERRLVL3	√	√	√		√	
ERRLVL2	√	√				
ERRLVL0 *	√					

* ERRLVL0 is recommended when uploading parameters from the drive for use in program or text file, as only the command name and set value will be reported. ERRLVL0 can not be used with RS-232 daisy chains.

NOTE: This command is intended to be used only during live terminal communication with the drive. Do not download this command to the drive, or place it in a program.

If you are using RS-485 Multi-Drop …

Using ERRLVL4 (factory default setting), you will have to address each command to its respective unit in the multi-drop (e.g., send 2_TASX to unit #2 in the chain). As an alternative, ERRLVL2 eliminates the need to address each command to the specific unit, but be aware that when you use ERRLVL2 there will be no ERROK or ERRBAD prompts and no error messages.

ERROK — Good Prompt

Type	Communication Interface
Syntax	<!>ERROK<i>,<i>,<i>,<i>
Units	n/a
Range	i = 0-255
Default	13,10,62,32
Response	ERROK: *ERROK13,10,62,32
See Also	ERRBAD, ERRLVL

Product	Rev
GT	1.02
GV	1.00
GT6	1.50
GV6	1.50

The Good Prompt (ERROK) command designates the characters to be placed into the output buffer after a command has been entered correctly. Up to 4 characters can be placed in the output buffer. These characters serve as a prompt for the next command. The characters are designated with their ASCII equivalent. For example, a carriage return is ASCII 13, a line feed is ASCII 10, a greater than symbol is ASCII 62, a space is ASCII 32, and no terminating character is designated with a zero. If the first field is a zero, the drive will only accept zeros from the other three fields. For a more complete list of ASCII equivalents, refer to the ASCII Table on page 193.

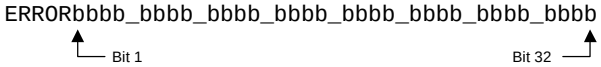
GT6/GV6 only: The ERROK characters are not transmitted if you send an immediate command (e.g., !TAS) to the product while it is executing a program.

NOTE: This command is intended to be used only during live terminal communication with the drive. Do not download this command to the drive, or place it in a program.

ERROR Error-Checking Enable

		Product	Rev
Type	Error Handling	GT	n/a
Syntax	<a_><!>ERROR<b><b>...<b><b> (32 bits)	GV	n/a
Units	n/a	GT6	1.50
Range	b = 0 (disable), 1 (enable), or X (don't change)	GV6	1.50
Default	0		
Response	ERROR: *ERROR0000_0000_0000_0000_0000_0000_0000_0000		
See Also	[ASX], ERRORP, GOWHEN, INFNC, K, LH, LS, S, TASX, TER, TRGFN		

When an error-checking bit is enabled (set to 1), the operating system will respond to a specific execution error by branching to the error program assigned with the ERRORP command (see table below). Each bit corresponds to a different error condition. For example, to enable checking for hard limit (bit #2) and soft limit (bit #3) error conditions, you would use the ERRORx11 command.



Bit #	**Function** (Error bits 13-18 and 20-32 are reserved.)	Branch Type
1 *	Stall Detected. (GT6 only)	GOSUB
2	Hard Limit Hit: When an input is defined as an end-of-travel input (INFNCi-R or INFNCi-S), and that input becomes active. Hard limits must be enabled (LH3).	JUMP if COMEXL0; GOSUB if COMEXL1
3	Soft Limit Hit: Functions when soft limits are enabled (LS3).	JUMP if COMEXL0; GOSUB if COMEXL1
4	Drive Fault: Detected only if the drive is enabled (DRIVE1).	JUMP
5	Commanded Kill or Commanded Stop (a K, !K, S, or !S command is sent).	!K = JUMP; !S = JUMP if COMEXS0; !S = GOSUB if COMEXS1, but need !C
6	Input Kill: When an input is defined as a KILL input (INFNCi-C), and that input becomes active.	JUMP
7	User Fault Input: When an input is defined as a user fault input (INFNCi-F), and that input becomes active.	JUMP
8	Stop Input: When an input is defined as a stop input (INFNCi-D), and that input becomes active.	JUMP
9	Enable input is activated (not grounded). See also TINO bit #6.	JUMP
10	Pre-emptive (on-the-fly) GO or registration move profile not possible at the time of attempted execution.	GOSUB
11 **	Target Zone Settling Timeout Period (set with the STRGTT command) is exceeded.	GOSUB
12 **	Maximum Position Error (set with the SMPER command) is exceeded.	GOSUB
19	Fieldbus Error	JUMP

* GT6 drive only; ** GV6 drive only

Example:

```
DEF PROG8     ; Define program 8 (which will be the error program)
IF(ER=bXX1)   ; If error is soft limit (bit #3 set to one), 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 PROG8  ; Assign PROG8 as error program. Branch to PROG8 upon hitting a
              ; soft limit.
ERRORXX1      ; Set error condition bit #3 to check for a soft limit
```

ERRORP Error Program Assignment

		Product	Rev
Type	Error Handling	GT	n/a
Syntax	<a_><!>ERRORP PROG<i>	GV	n/a
Units	i = number of the program created with DEF PROG	GT6	1.50
Range	0, 1-32 (0 clears the existing program assignment)	GV6	1.50
Default	0 (no program is assigned)		
Response	ERRORP: *ERRORP PROG8		
See Also	DEF PROG, DEL PROG, ERRLVL, ERROR, TER		

Using the ERRORP PROG command, you can assign any previously defined program (DEF PROG) as the error program. For example, to assign a previously defined program #8 as the error program, enter the ERRORP PROG8 command. If you later decide not to have an error program, issue the ERRORP PROG0 command; after the ERRORP PROG0 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=b1ØX)). An error program set-up example is provided on page 26.

When an error condition occurs and the associated error-checking bit has been enabled with the ERROR command, the Gemini drive will branch to the error program. Depending on the error condition, the branch will be either a JUMP 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 JUMP command to go to a different program. If the error condition calls for a JUMP, there is no way to return to the point at which the error occurred.

The ERRORP assignment is saved in EEPROM memory. If the program that is identified as the ERRORP program is deleted with the DEL PROG command, the ERRORP assignment is automatically cleared.

NOTES

- **Where to include the** ERROR **and** ERRORP **commands:** (we recommend one of these options)
  - Add these commands to the setup/configuration file generated when using the configuration wizard. If using Motion Planner, see step 6 on page 7 to modify the setup file. If using Pocket Motion Planner (see page 11), you can edit the configuration file in a text editor (it has a ".pmp" extension and is located in the directory with GEMINI.EXE).
  - Add these commands as the first two commands in the program that is assigned as the start-up program 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.
- **Using "Keep Alive"**: If the drive is being powered from AC power alone, when AC power is removed, the ERRORP program is only partially executed. To ensure the ERRORP program is fully executed when AC power is removed, always have +24VDC power present on the "Keep Alive" input (the **+24V DC** and **24V RTN** pins).

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 command. For example, to disable error checking for the kill input activation (bit #6), issue the ERRORxxxxx0 command. To re-enable the error-checking bit, issue the ERRORxxxxx1 command.

- Delete the program assigned as the ERRORP program (DEL PROGn).
- Satisfy the *How to Remedy the Error* requirement identified in the table below.

NOTE

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

ERROR **Bit #**	**Cause of the Error**	**Branch Type to** ERRORP	**How to Remedy the Error**
1	GT6 only. Stall detected.	Gosub	Issue a GO command.
2	Hard Limit Hit. An input is defined as an end-of-travel input (INFNCi-R or INFNCi-S), and that input became active. Hard limits must be enabled (see LH).	If COMEXL0, then Jump; If COMEXL1, then Gosub	Change direction & issue GO command; or issue LH0.
3	Soft Limit Hit (soft limits must be enabled first—see LS).	If COMEXL0, then Jump; If COMEXL1, then Gosub	Change direction & issue GO command; or issue LS0.
4	Drive Fault (Detected only if drive enabled – DRIVE1).	Jump	Check the cause with TASX, remedy the fault condition, then issue a DRIVE1 command.
5	Commanded Stop or Kill (whenever a K, !K, S, or !S command is sent).	If !K, then Jump; If !S & COMEXS0, then Jump; If !S & COMEXS1, then Gosub, but need !C	No fault condition is present—there is no error to clear. If you want the program to stop, you must open the Enable Interlock.
6	Kill Input Activated (see INFNCi-C).	Jump	Deactivate the kill input.
7	User Fault Input Activated (see INFNCi-F).	Jump	Deactivate the user fault input, or disable it by assigning it a different INFNC function.
8	Stop Input Activated (see INFNCi-D).	Jump	Deactivate the stop input, or disable it by assigning it a different INFNC function.
9	Enable input not grounded.	Jump	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	GV6 only. Target Zone Timeout (STRGTT value has been exceeded).	Gosub	Issue these commands in this order: STRGTEØ, DØ, GO, STRGTE1
12	GV6 only. Exceeded Max. Allowable Position Error (set with the SMPER command).	Gosub	Issue a DRIVE1 command. Verify that the feedback device is working properly.
19	Fieldbus Error Detected	Jump	RESET the drive or cycle power.

Reserved Bits: Bits 13 – 18, 20 – 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 K command to end program execution or you can use the JUMP command to go to a different program. If the error condition calls for a JUMP, there is no way to return to the point at which the error occurred.

Example:

```
DEF PROG8      ; Define program 8 (which will be the error program)
IF(ER=bXX1)    ; If error is soft limit (bit #3 set to one), 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 PROG8   ; Set program #8 to be the error program.
               ; Branch to program #8 upon hitting a soft limit.
ERRORXX1       ; Set error condition bit #3 to check for a soft limit
```

ESK — Fault on Stall

Type	Drive Configuration
Syntax	<a_><!>ESK<b>
Units	b = enable bit
Range	0 (disable) or 1 (enable)
Default	1
Response	ESK: *ESK0
See Also	DRIVE, DSTALL, TAS, TASX, TER

Product	Rev
GT	1.02
GV	n/a
GT6	1.50
GV6	n/a

Use ESK to enable/disable the Fault on Stall mode. If the Fault on Stall mode is enabled (ESK1), the occurrence of a stall will immediately stop pulses from being sent to the motor and will disable the drive (DRIVE0).

Factory default settings for stall detection:

- Status Reporting: Stalls are reported in TAS bit #12 and TER bit #1if ESK1 mode is enabled. Stalls are always reported with TASX bit #17, regardless of the ESK setting. All three of these status bit are cleared with a DRIVE1 command.
- DSTALL0: Stall detection is disabled.
- ESK1: When a stall occurs, kill pulses to the motor. The GT responds with a "fault" (disables the drive with DRIVE0, and activates output #2 and the relay output). The GT6 responds with a kill (!K), and if Disable Drive on Kill is enabled (KDRIVE1) it disables the drive (DRIVE0).

FLTDSB — Fault on Drive Disable (DRIVE0)

Type	Drive Configuration
Syntax	<a_><!>FLTDSB<b>
Units	b = enable bit
Range	0 (disable) or 1 (enable)
Default	1
Response	FLTDSB: *FLTDSB1
See Also	DRIVE, FLTSTP, TAS, TASX, TER

Product	Rev
GT	1.02
GV	1.00
GT6	1.50
GV6	1.50

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), output #2 (pin 43 on the DRIVE I/O connector) is activated and the dry contact relay (labeled "RELAY COM" and "RELAY N.O." on the 4-pin removable connector) is opened.

FLTSTP — Fault on Startup Incoming Indexer Pulses

Type	Drive Configuration
Syntax	<a_><!>FLTSTP<b> (does not take effect until RESET or cycle power)
Units	b = enable bit
Range	0 (disable) or 1 (enable)
Default	1
Response	FLTSTP: *FLTSTP1
See Also	DMODE, DRIVE, FLTDSB, TAS, TASX, TER

Product	Rev
GT	1.02
GV	1.00
GT6	n/a
GV6	n/a

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

Use the FLTSTP command to enable/disable the Fault on Incoming Indexer Pulses mode. If this mode is enabled (default setting), a drive fault will occur if the drive is receiving step pulses in excess of 20 pulses per second during power up or drive enable (DRIVE1). The drive fault comprises the following :

- Set these status bits: TAS bits #13 and #14
  TER bit #4
  TASX bit #8
- Activate output #2 (pin 43 on the DRIVE I/O connector), and opens the dry contact relay (labeled "RELAY COM" and "RELAY N.O.") on the 4-pin removable connector.
- Disable the drive (DRIVE0).

The FLTSTP command is only applicable to DMODE6, DMODE7, DMODE8, and DMODE9.

GO — Initiate Motion

Type	Motion
Syntax	<a_><!>GO<b>
Units	n/a
Range	b = 0 (don't go) or 1 (go)
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, TAS, V

Product	Rev
GT	1.02
GV	1.00
GT6	1.50
GV6	1.50

The Initiate Motion (GO) command instructs the motor to make a move using motion parameters that have been previously entered. Several commands affect the motion that will occur when a GO is received: 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 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)

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 page 44.

Example:

```
DEL PROG1     ; Delete program #1
DEF PROG1     ; Begin definition of program #1
MA0           ; Incremental positioning mode
MC0           ; Preset positioning mode
A10           ; Set the acceleration to 10 revs/sec/sec
V1            ; Set the velocity to 1 revs/sec
D100000       ; Set the distance to 100000 counts
GO1           ; Initiate motion
END           ; End definition of program #1
```

GOBUF Store a Motion Segment in Compiled Memory

		Product	Rev
Type	Compiled Motion	GT	n/a
Syntax	<a_><!>GOBUF<b>	GV	n/a
Units	n/a	GT6	1.50
Range	b = 0 (don't go) or 1 (go)	GV6	1.50
Default	1		
Response	n/a		
See Also	DEF PROF, END, MA, MC, POUTA, PRUN PROF, TAS, TER, TRGFN, TSS, VF, (Compiled Motion overview on page 49)		

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 profiles defined with the DEF PROF command.

Each GOBUF motion segment may have its own distance to travel, velocity, acceleration and deceleration. The end of a preset motion (MCØ) segment 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 velocity specified; a compiled MC1 GOBUF motion segment is finished when the velocity 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).

Triggered GOBUF segments with an input: If you wish a GOBUF segment to be initiated with a trigger interrupt input, use the TRGFNc1 command. For more information, refer to the TRGFN command description.

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.

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.

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.

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.

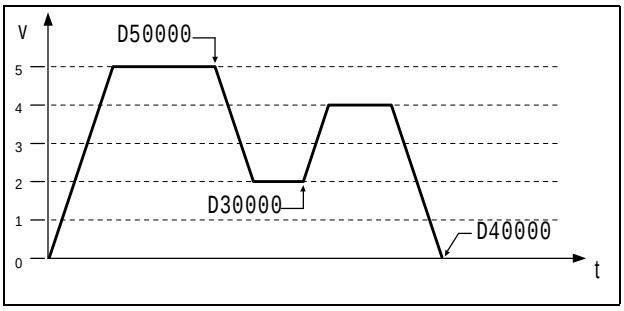
You may use a mode continuous (MC1) non-zero velocity segment as the last motion segment in a profile (no error will result). The axis will just continue traveling at the goal velocity.

NOTE: Each GOBUF motion segment can consume 12 bytes of compiled memory. If there is no more space left in compiled memory, the drive will respond with the ERRBAD prompt (default prompt is "?").

Example:

```
DEL PROF1   ; Delete profile #1
DEF PROF1   ; Begin definition, profile #1
MC0         ; Preset positioning mode
D50000      ; Distance is 50000
A10         ; Acceleration is 10 rps/s
AD10        ; Deceleration is 10 rps/s
V5          ; Velocity is 5 rps
GOBUF1      ; 1st motion segment
D30000      ; Distance is 30000
V2          ; Velocity is 2 rps
GOBUF1      ; 2nd motion segment
D40000      ; Distance is 40000
V4          ; Velocity is 4 rps
GOBUF1      ; 3rd motion segment
END         ; End program definition

PRUN PROF1 ; Run profile #1
```

The resulting profile from this program:



GOSUB — Call a Subroutine

Type	Program Flow Control
Syntax	<a_><!>GOSUB PROG<i>
Units	i = number of the program created with DEF PROG
Range	1-32
Default	n/a
Response	n/a
See Also	DEF PROG, END, ERROR, ERRORP, JUMP, RUN PROG

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

The GOSUB PROG command, executed within a program, branches to ("calls") the specified program as a subroutine. After the "called" program/subroutine is completed, program control returns to the "calling" program at the command immediately following GOSUB PROG. If an invalid program number is entered, no branch will occur and processing will continue with the line after GOSUB PROG.

Up to 16 levels of subroutine calls can be made without receiving an error.

An alternative to using the GOSUB PROG command is to use the RUN PROG command or the PROG command. These two additional methods function identically to GOSUB PROG.

If you wish to create a branch to a program and not return to the calling program, use the JUMP PROG command.

Example:

```
DEF PROG1     ; Begin definition of program #1 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 #2 as a subroutine
LN            ; End loop
END           ; End subroutine definition of program #1
DEF PROG2     ; Begin definition of program #2
OUT1          ; Turn on output #1
T1            ; Time delay of 1 second
OUT0          ; Turn off output #1
END           ; End subroutine definition of program #2
RUN PROG1     ; Execute program #1
; After program #1 is initiated, and the first move of 4000 counts is complete, a
; GOSUB will occur causing the execution of program #2. After program #2 finishes,
; control will be passed back to program #1, to the command immediately following
; the GOSUB.
```

GOWHEN Conditional GOBUF

Type	Compiled Motion
Syntax	<a_><!>GOWHEN(T = <i>)
Units	i = time in milliseconds
Range	1 to 999999
Default	n/a
Response	n/a
See Also	COMEXC, GOBUF, T, TAS, TER, TRGFN, WAIT

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

In compiled motion profiles, you can use the GOWHEN command to delay execution of the subsequent GOBUF statement until the specified time delay (in milliseconds) has been satisfied. During the time delay, the profile in progress continues at constant velocity. For example, when progress through the profile reaches the GOWHEN(T=500) command, execution of the subsequent GOBUF is paused for ½ second.

A preset GOBUF command that is already in motion can start a new profile using the GOWHEN and GOBUF 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 GOBUF sequence.

While a subsequent GOBUF command is suspended (waiting for the GOWHEN conditional expression to be true), axis status (TAS) bit #26 is set. This bit is cleared when the GOWHEN condition evaluates true, or if a stop (!S) or a kill (!K) is executed.

NOTE: If you wish motion to be triggered with a trigger input, use the TRGFNc1 command.

HOM Go Home

Type	Homing
Syntax	<a_><!>HOM<b>
Units	n/a
Range	b = 0 (home in positive direction), or 1 (home in negative direction)
Default	n/a
Response	n/a
See Also	HOMA, HOMBAC, HOMDF, HOMEDG, HOMV, HOMVF, HOMZ, INLVL, PSET, TAS, TIN

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

The HOM command instructs the Gemini to search for the home position in the specified direction.

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 each axis status register (refer to the TAS command). *When the homing operation is successfully completed, the absolute position register is set to zero (equivalent to PSETØ).*

NOTE

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 & deceleration (HOMA)
- Home velocity (HOMV)
- Final home velocity (HOMVF)
- Home reference edge (HOMEDG)
- Backup to home (HOMBAC)
- Final home direction (HOMDF)
- Active state of home input (INLVL)
- Home to encoder Z-channel (HOMZ) – GV6 only

For more information on homing, including sample scenarios, refer to the *Homing* section (page 31).

Example:

```
DEL PROG20    ; Delete program #20
DEF PROG20    ; Begin definition of program #20
MA0           ; Select incremental positioning mode
MC0           ; Select preset positioning mode
HOMA10        ; Set home acceleration/deceleration to 10 revs/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 (GV6 only)
INFNC4-T      ; Assign input #4 to function as the home input
INLVLxxx0     ; Set home input active level to low
HOMV1         ; Set home velocity to 1 revs/sec
HOMVF.1       ; Set home final velocity to 0.1 revs/sec
HOM0          ; Execute go home in positive-direction
END           ; End definition of program #20
```

HOMA Home Acceleration

Type	Homing
Syntax	<a_><!>HOMA<r>
Units	r = revs/sec/sec (linear motors: see page 44 for linear/rotary conversion)
Range	0.0001 – 9999.9999
Default	10.0000
Response	HOMA: *HOMA10.0000
See Also	DMEPIT, HOM, HOMBAC, HOMDF, HOMEDG, HOMV, HOMVF, HOMZ

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

The Home Acceleration (HOMA) command specifies the acceleration and deceleration rate to be used upon executing the next go home (HOM) command. The motion will be trapezoidal (s-curve accel/decel is not available for homing profiles).

Example: Refer to the go home (HOM) command example.

HOMBAC Home Backup Enable

Type	Homing
Syntax	<a_><!>HOMBAC<b>
Units	n/a
Range	b = 0 (disable) or 1 (enable)
Default	0
Response	HOMBAC: *HOMBAC0
See Also	HOM, HOMA, HOMDF, HOMEDG, HOMV, HOMVF, HOMZ

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

The Home Backup Enable (HOMBAC) command enables or disables the backup to home switch function. When this function is enabled, the motor will decelerate to a stop after encountering the active edge of the home region, and then move the motor in the opposite direction at the home final velocity (HOMVF) until the active edge of the home region is encountered. This motion will occur regardless of whether or not the home input is active at the end of the deceleration of the initial go home move.

Example: Refer to the go home (HOM) command example.

HOMDF Home Final Direction

		Product	Rev
Type	Homing	GT	n/a
Syntax	<a_><!>HOMDF<b>	GV	n/a
Units	n/a	GT6	1.50
Range	b = 0 (positive-direction) or 1 (negative-direction)	GV6	1.50
Default	0		
Response	HOMDF: *HOMDF0		
See Also	HOM, HOMA, HOMBAC, HOMEDG, HOMV, HOMVF, HOMZ, INFNC, INLVL		

The Home Final Direction (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).

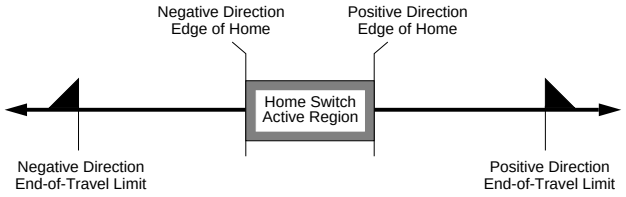
Example: Refer to the go home (HOM) command example.

HOMEDG Home Reference Edge

		Product	Rev
Type	Homing	GT	n/a
Syntax	<a_><!>HOMEDG<b>	GV	n/a
Units	n/a	GT6	1.50
Range	b = 0 (positive-direction edge) or 1 (negative-direction edge)	GV6	1.50
Default	0		
Response	HOMEDG: *HOMEDG0		
See Also	HOM, HOMA, HOMBAC, HOMDF, HOMV, HOMVF, HOMZ, INFNC, INLVL		

The Home Reference Edge (HOMEDG) command specifies which edge of the home switch the homing operation will consider as its final destination.

As illustrated below, the positive-direction edge of the home switch is defined as the first switch transition seen by the Gemini drive 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 drive 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

		Product	Rev
Type	Homing	GT	n/a
Syntax	<a_><!>HOMV<r>	GV	n/a
Units	r = revs/sec (linear motors: see page 44 for linear/rotary conversion)	GT6	1.50
Range	0.0000-200.0000	GV6	1.50
Default	1.0000		
Response	HOMV: *HOMV1.0000		
See Also	DMEPIT, HOM, HOMA, HOMBAC, HOMDF, HOMEDG, HOMVF, HOMZ		

The Home Velocity (HOMV) command specifies the velocity to use when the home algorithm begins its initial go home (HOM) move. The velocity remains set until you change it with a subsequent home velocity command.

Example: Refer to the go home (HOM) command example.

HOMVF — Home Final Velocity

		Product	Rev
Type	Homing	GT	n/a
Syntax	<a_><!>HOMVF<r>	GV	n/a
Units	r = revs/sec (linear motors: see page 44 for linear/rotary conversion)	GT6	1.50
Range	0.0000-200.0000	GV6	1.50
Default	0.1000		
Response	HOMVF: *HOMVF0.1000		
See Also	DMEPIT, HOM, HOMA, HOMBAC, HOMDF, HOMEDG, HOMV, HOMZ		

The Home Final Velocity (HOMVF) command specifies the velocity to use when the home algorithm does its final approach. This command is only operational when backup to home (HOMBAC) is enabled, or when homing to an encoder Z channel (HOMZ).

Example: Refer to the go home (HOM) command example.

HOMZ — Home to Encoder Z-channel Enable

		Product	Rev
Type	Homing	GT	n/a
Syntax	<a_><!>HOMZ<b>	GV	n/a
Units	n/a	GT6	n/a
Range	b = 0 (disable) or 1 (enable)	GV6	1.50
Default	0		
Response	HOMZ: *HOMZ0		
See Also	HOM, HOMA, HOMBAC, HOMDF, HOMEDG, HOMV, HOMVF, INFNC, INLVL		

This 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.

Example: Refer to the go home (HOM) command example.

IF — IF Statement

		Product	Rev
Type	Program Flow Control; Conditional Branching	GT	n/a
Syntax	<a_><!>IF(*expression*)	GV	n/a
Units	(see syntax examples below)	GT6	1.50
Range	(see syntax examples below)	GV6	1.50
Default	n/a		
Response	n/a		
See Also	A, AD, D, ELSE, NIF, TAS, TASX, TER, TIN, TPC, TPE, TPER, TSS, V, VARI		

The IF command is used in conjunction with the ELSE and NIF commands to provide conditional branching. If the IF expression 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.

IF statement programming order:
```
IF(expression)...commands...NIF
or
IF(expression)...commands...ELSE...commands...NIF
```

Multiple parentheses may not be used within the IF command.

Syntax Example: Binary Data

In this example, the IF condition evaluates true when bit #12 of the Axis Status register is set (binary value is "1").

```
IF (AS = bxxxxxxxxxxx1)
```

Operand for the selected status register.
Options are:
- AS ..... Axis status (see TAS)
- ASX ... Extended axis status (see TASX)
- ER ..... Error status (see TER)
- IN ..... Input status (see TIN)
- FBS ... Fieldbus status (see TFBS)
- SS ..... System status (see TSS)

Binary state.
- 0 = bit is false, or not set
- 1 = bit is true, or set
- X = ignore bit (mask)

The bit pattern is numbered 1-n, left to right. This example evaluates true if bit #12 is set.

"b" is required to prefix the binary state.

"=" is required.

Syntax Example, using the binary bit-select shortcut:

```
IF (AS.12 = b0)
```

AS = Axis Status register

Bit select operator (.) is required

Bit #12 is selected

Bit state (0 = false, 1 = true)

"b" is required to prefix the binary state

"=" is required

Syntax Example: Integer/Variable Data Comparison

(This capability is available as of OS rev 1.60)

In this example, the IF condition compares the present encoder position (PE) with the value of integer variable #5 (VARI5). The condition evaluates true when the integer value of the encoder position is less than the value of VAR5.

```
IF (PE < VARI5)
```

Operand options (left & right):
- Numeric constant
- Integer variable (VARI*n*)
- System variables options (e.g., VARI5=PC):
  - A ..... Programmed acceleration (see A)
  - AD ..... Programmed deceleration (see AD)
  - V ..... Programmed velocity (see V)
  - D ..... Programmed distance (see D)
  - ANI ..... Analog input(see TANI)
  - PC ..... Commanded position (see TPC)
  - PE ..... Encoder/resolver position (see TPE)
  - PER ..... Position error (see TPER)

Operator options:
- = ... Equals
- <> ..... Not equal to
- > ..... Greater than
- < ..... Less than
- >= ..... Greater than or equal to
- <= ..... Less than or equal to

NOTE

System variables A, AD, and V, are real numbers, with a resolution of 0.0001. When comparing the value of these variables, the IF condition uses the integer representation (removes the decimal point). For example, if the commanded velocity is 5.0000 units/sec, the integer observed by an IF evaluation would be 50000. Thus, if you want the IF condition to evaluate true when the commanded velocity (V) is <= 5.0000 units/sec, use this syntax: IF(V<=50000).

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
```

INDEB — Input Debounce Time

Type	Inputs
Syntax	<a_><!>INDEB<i>
Units	i = time in milliseconds (ms)
Range	i = 2-250
Default	50
Response	INDEB: *INDEB50
See Also	INFNC, INLVL, TIN

Product	Rev
GT	1.02
GV	1.00
GT6	1.50
GV6	1.50

The INDEB command governs the debounce time for the digital inputs on the DRIVE I/O connector:

Input #	Pin #	GT & GV Function (fixed)	GT6 & GV6 Function (INFNC default)
1	28	Positive end-of-travel limit	INFNC1-R (Positive end-of-travel limit)
2	29	Negative end-of-travel limit	INFNC2-S (Negative end-of-travel limit)
3	31	User fault	INFNC3-T (Home limit)
4	34	*(input not available)*	INFNC4-H (Trigger interrupt)
5	35	*(input not available)*	INFNC5-A (General-purpose)
6	37	*(input not available)*	INFNC6-A (General-purpose)
7	38	*(input not available)*	INFNC7-A (General-purpose)
8	39	*(input not available)*	INFNC8-A (General-purpose)

GT/**GV**: Only input #3 (User Fault input) is debounced by INDEB (INDEB is n/a to inputs #1 and #2).

GT6/**GV6**: Inputs defined as limit inputs (INFNCi-R, INFNCi-S, or INFNCi-T) will not be debounced.

The debounce is the period of time that the input must be held in a fixed state before the drive recognizes it and reports the state with the TIN command. The default setting is 50 ms.

INFNC — Input Function

Type	Input
Syntax	<a_><!>INFNC<i>-<c>
Units	i = general-purpose input #; c = function identifier letter
Range	i = 1-8; c = A, B, C, D, E, F, H, R, S, T
Default	Input 1 = R (positive direction end-of-travel limit) Input 2 = S (negative direction end-of-travel limit) Input 3 = T (home limit) Input 4 = H (trigger interrupt) All other inputs set to A (general purpose)
Response	INFNC6: *INFNC6-A NO FUNCTION - STATUS OFF *(NOTE: input # must be specified for report back)*
See Also	COMEXR, COMEXS, ERROR, INDEB, INLVL, INSELP, INUFD, K, LH, PSET, TER, TIN, TRGFN, TRGLOT, TSS

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

The Input Function (INFNC) command defines the function of each of the 8 general-purpose inputs located on the DRIVE I/O connector. For example, the INFNC6-D command defines input #6 as a stop input (identified by function "D").

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 Gemini 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 Scan Rate. The programmable inputs are scanned once per millisecond.

Input Active Level. The active level for each input is set with the INLVL command.

Status. Check the binary status report with the TIN command.

Default Function Assignments. *(pin numbers refer to the* DRIVE I/O *connector)*

Input #	Pin #	Factory Default Function
1	28	INFNC1-R (positive direction end-of-travel limit)
2	29	INFNC2-S (negative direction end-of-travel limit)
3	31	INFNC3-T (home limit)
4	34	INFNC4-H (trigger interrupt)
5	35	INFNC5-A (general-purpose input)
6	37	INFNC6-A (general-purpose input)
7	38	INFNC7-A (general-purpose input)
8	39	INFNC8-A (general-purpose input)

INFNCi-A General Purpose Input

INFNCi-B BCD Program Select

The *BCD program select* function allows you to execute defined programs by activating the program select inputs. BCD program select inputs are assigned BCD weights, with the least weight (1) on the smallest numbered input. The next BCD weight is assigned to the next input defined as a BCD input. To execute a particular program, you activate the combination of inputs to achieve the BCD weight that corresponds to the *number of the program*. The program number is determined by the DEF PROGn command, where "n" is the number of the program (e.g., DEF PROG16 begins the definition of program #16).

To execute the maximum possible number of stored programs (32), you would need six inputs configured as BCD program select inputs. For example, the table below shows the BCD weights if inputs 3-8 are configured as BCD program select inputs.

Input #	BCD Weight	
Input 3	1	← least significant bit value
Input 4	2	
Input 5	4	
Input 6	8	
Input 7	10	
Input 8	20	← most significant bit value

Examples: Activating inputs 7 and 4 would execute program #12, activating inputs 4, 7 and 8 would execute program #32.

Before you can execute programs using the BCD program select inputs, you must first enable scanning with the INSELP1 command. Once enabled, the Gemini will continuously scan the BCD inputs and execute the program (by number) according to the weight of the currently active BCD inputs. After executing and completing the selected program, the Gemini will scan the inputs again. NOTE: To disable scanning, execute the !INSELP0 command, or place the INSELP0 command in a program that can be selected with BCD inputs.

The INSELP command also determines how long the BCD program select input level must be maintained before the Gemini executes the program. This delay is referred to as debounce time (but is not affected by the INDEB setting).

INFNCi-C Kill

When a *Kill* input goes active:

- Motion stops (using the LHAD and LHADA decel rate). This is an edge detection function and is not intended to inhibit motion; to inhibit motion, use the Pause/Resume function (INFNCi-E).
- The program currently in progress is terminated.
- Commands currently in the command buffer are eliminated.
- The drive is left in the enabled state (DRIVE1), unless the "disable drive on kill" function is enabled with the KDRIVE1 command.
- Error bit #6 in the error status register (see TER) is set.
- If error-checking bit #6 is enabled (e.g., ERRORxxxxx1), the error program (assigned with the ERRORP command) will be executed to respond to the error condition.

INFNCi-D Stop

A *Stop* input stops motion. Motion deceleration during the stop is controlled by the AD & ADA commands. Activating a Stop input sets error bit #8 (see TER). If error-checking bit #8 is enabled (ERRORxxxxxxx1), activating a Stop input will cause a branch to the ERRORP program. The stop input function is effected by the COMEXS setting, as follows:

- COMEXS0 (factory default setting): Motion stops. The motion profile cannot be resumed. Program execution is terminated and commands in the command buffer are discarded. Program execution cannot be resumed.
- COMEXS1: Motion stops. The motion profile can be resumed with the !C command or a resume input (INFNCi-E). Program execution stops, but the commands in the command buffer are saved. Program execution can be resumed with !C or a resume input (INFNCi-E).
- COMEXS2: The drive responds as it does in the COMEXS0 mode, with the exception that you can still use the BCD inputs to select programs (INSELP value is retained). For more details on BCD program selection, refer to the INFNCi-B function (above) and INSELP.

INFNCi-E Pause/Continue

A *Pause/Continue* input will affect motion and program execution depending on the COMEXR command setting, as described below. In both cases, when the input is activated, the command being processed will be allowed to finish executing before the program is paused.

- COMEXR0: Only program execution is paused; motion in progress continues to its programmed destination. Release the pause input or execute !C to resume program execution.
- COMEXR1: Both motion and program execution are paused; the motion stop function is used to halt motion. *After motion stops*, release the pause input or execute !C to resume motion and program execution.

INFNCi-F User Fault

When a *User Fault* input goes active:

- Motion stops (using the LHAD and LHADA deceleration rate).
- The program currently in progress is terminated.
- Commands currently in the command buffer are eliminated.
- Error Status (TER) bit #7 is set, and Extended Axis Status (TASX) bit #23 is set. These status bits are cleared when the input is deactivated.
- If error-checking bit #7 is enabled (e.g., ERRORxxxxxx1), the error program (assigned with the ERRORP command) will be executed to respond to the error condition.
- The drive will continue to send current to the motor windings for the time specified by INUFD.
- The drive is disabled (DRIVE0). CAUTION: This allows the load to freewheel to a stop.

INFNCi-H Trigger Interrupt

A *Trigger Interrupt* input can be used for these purposes:

- Registration. (see RE description or page 49 for details)
- Trigger a pending GOBUF in compiled motion. (see TRGFN description for details)

Trigger Interrupt Debounce: The "Trigger Interrupt" input debounce is governed by the TRGLOT command setting (default is 24 ms). The TRGLOT setting overrides the existing INDEB setting for only the general-purpose inputs that are assigned the "Trigger Interrupt" function.

Restriction: Multiple trigger interrupt inputs may be used for TRGFN functions; however, only one of the inputs may be used for triggering a registration move (the lowest number trigger interrupt input is used). (e.g., if INFNC4-H and INFNC5-H, only input #4 may be used for registration)

INFNCi-R End-of-Travel Limit, Positive Direction

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. REMEMBER: Once an input is assigned a limit function, it is no longer debounced (INDEB has no effect), and it must be enabled with an LH command before being active.

INFNCi-S End-of-Travel Limit, Negative Direction

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. REMEMBER: Once an input is assigned a limit function, it is no longer debounced (INDEB has no effect), and it must be enabled with an LH command before being active.

INFNCi-T Home Limit

This assigns the home limit input function. For example, INFNC3-T assigns the "Home limit" function to general-purpose input #3. REMEMBER: Once an input is assigned a limit function, it is no longer debounced (INDEB has no effect). For more information homing, refer to the HOM command and to the homing feature overview on page 31.

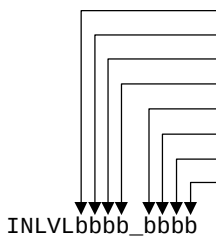
INLVL Input Active Level

Type	Input
Syntax	<a_><!>INLVL<bbbbbbbb>
Units	n/a
Range	b = 0 (active low) or 1 (active high)
Default	11000000
Response	INLVL: *INLVL1100_0000
See Also	INDEB, INFNC, TIN

Product	Rev
GT	1.02
GV	1.00
GT6	1.50
GV6	1.50

The INLVL command defines the active state of the general-purpose inputs on the DRIVE I/O connector. Example: INLVL11000000 sets the active level for inputs 1 & 2 to active high, all other inputs are set to active low.

INLVL bit assignments (bits are numbered 1-8 from left to right):



Input #	Pin #	GT & GV Function (fixed)	GT6 & GV6 Function (INFNC default)
1	28	Positive end-of-travel limit	INFNC1-R (Positive end-of-travel limit)
2	29	Negative end-of-travel limit	INFNC2-S (Negative end-of-travel limit)
3	31	User fault	INFNC3-T (Home limit)
4	34	*(input not available)*	INFNC4-H (Trigger interrupt)
5	35	*(input not available)*	INFNC5-A (General-purpose)
6	37	*(input not available)*	INFNC6-A (General-purpose)
7	38	*(input not available)*	INFNC7-A (General-purpose)
8	39	*(input not available)*	INFNC8-A (General-purpose)

NOTE: If you do not address all of the available inputs with the INLVL command (e.g., INLVL01 addresses only inputs 1 and 2), the remaining input levels will be set to active low. GV & GT only: If you attempt to change an unused bit (bits 4-8) to "1", the drive will respond with the "?" error prompt and will not implement the INLVL command.

Active Level Relationships

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 your drive's *Hardware Installation Guide* for wiring instructions). Pin 27 is the pull up for inputs 1-3, and pin 33 is the pull up for inputs 4-8.

INSELP Select Program Enable

Type	Input
Syntax	<a_><!>INSELP<b>,<i>
Units	b = strobe enable/disable bit i = strobe time in milliseconds
Range	b = 0 (disable) or 1 (enable) i = 0-5000
Default	0,0
Response	INSELP: *INSELP1,40
See Also	COMEXS, DEF PROG, INFNC, INLVL, TSS

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

The INSELP command enables program selection with BCD inputs (that is, inputs configured as BCD program select inputs with the INFNCi-B command). In addition, the INSELP command establishes the strobe time for the BCD inputs. 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 (1-32).

The BCD 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* (1 millisecond).

The Kill (!K) command releases this mode, in addition to INSELP0. 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 (COMEXS0). This mode will always be disabled after a reset

Example:

```
DEL PROG1     ; Delete program #1
DEF PROG1     ; Begin definition of program #1
TAS           ; Transfer axis status
END           ; End program
DEL PROG2     ; Delete program #2
DEF PROG2     ; Begin definition of program #2
TREV          ; Transfer operating system revision
END           ; End program
DEL PROG3     ; Delete program #3
DEF PROG3     ; Begin definition of program #3
TPC           ; Transfer commanded position
END           ; End program
INFNC1-B      ; Assign input 1 as a BCD program select input
INFNC2-B      ; Assign input 2 as a BCD program select input
INSELP1,50    ; Enable scanning BCD inputs, levels must be maintained for 50ms
; You can now execute the programs by activating the correct
; combination of inputs:
;  * Activate only input 1 (BCD weight of 1) to execute program #1
;  * Activate only input 2 (BCD weight of 2) to execute program #2
;  * Activate inputs 1 & 2 (BCD weight of 3) to execute program #3
```

INUFD — User Fault Input Delay

Type	Input
Syntax	<a_><!>INUFD<i>
Units	milliseconds
Range	0 – 1000
Default	0
Response	INUFD: *INUFD100
See Also	INFNC, OUTBD, OUTFNC

Product	Rev
GT	1.61
GV	1.61
GT6	1.70
GV6	1.70

The INUFD command specifies the amount of time that current will remain in the motor windings after a user fault input (INFNCn-F) is seen. The fault output (OUTFNCn-F) will still be immediately asserted. This command is intended to be used in vertical applications, where the brake must be enabled while the motor still has torque so that the load is always supported. This is the complement to the OUTBD command.

JUMP — Jump to a Program (and do not return)

Type	Program Definition; Program Flow Control
Syntax	<a_><!>JUMP PROG<i>
Units	i = program number
Range	1-32
Default	n/a
Response	n/a
See Also	DEF PROG, DEL PROG, END, ERROR, ERRORP, GOSUB, IF, L, RUN PROG

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

The JUMP PROG command, executed within a program, branches to ("calls") the specified program.

All nested IFs, loops (L), and GOSUB subroutines are cleared; thus, the program that the JUMP PROG initiates will **not** return control to the line after JUMP PROG, when the program completes operation. Instead, the program will end.

If an invalid program number is entered, JUMP PROG will be ignored, and processing will continue with the line after JUMP PROG.

If you wish to create a branch to a program and return to the calling program, use the GOSUB PROG command.

Example

```
; ********************************************************************************
; * In this example, program #23 is executed and calls program #21 as a subroutine. *
; * Program #21 then initiates motion (GO1) and jumps to program #22 to turn on     *
; * output #2. Then, because the JUMP PROG command cleared the Program 21           *
; * subroutine, program execution is terminated instead of returning to program #23.*
; ********************************************************************************
DEL PROG21    ; Delete program #21
DEF PROG21    ; Begin definition program #21
GO1           ; Initiate motion
JUMP PROG22   ; Jump to program #22
END           ; End subroutine definition
DEL PROG22    ; Delete program #22
DEF PROG22    ; Begin definition of program #22
OUTx1         ; Turn on output #2
END           ; End program definition
DEL PROG23    ; Delete program #23
DEF PROG23    ; Begin definition of program #23
GOSUB PROG21  ; Gosub to program #21
GO1           ; Initiate motion
END           ; End subroutine definition
RUN PROG23    ; Execute program #23
```

K — Kill Motion

Type	Motion
Syntax	<a_><!>K<b>
Units	n/a
Range	b = 0 (don't kill) or 1 (kill)
Default	n/a
Response	n/a
See Also	ERROR, ERRORP, GO, KDRIVE, LHAD, LHADA, S, TAS, TER

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

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).
  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, and error status (TER) bit #5 is set. If error-checking bit #5 is enabled (e.g., ERRORxxxx1), executing a Kill command will cause a branch to the ERRORP program.

If you want the drive to be disabled upon executing a Kill command, enable the *Disable Dive on Kill* mode with the KDRIVE1 command. **CAUTION**: In the KDRIVE1 mode, a Kill command or Kill input immediately shuts down the drive (DRIVE0) and allows the load to *free wheel* to a stop.

Example:

```
A2        ; Set acceleration to 2 revs/sec/sec
AD2       ; Set deceleration to 2 revs/sec/sec
V1        ; Set velocity to 1 rev/sec
D100000   ; Set distance to 100,000 counts
GO1       ; Initiate motion -- motion begins.
          ; After a short period the Kill command is sent.
!K        ; Kill motion (stop at the LHADA/LHAD decel)
```

KDRIVE — Disable Drive on Kill

Type	Drive Configuration
Syntax	<a_><!>KDRIVE<b>
Units	b = enable bit
Range	0 (disable) or 1 (enable)
Default	0
Response	KDRIVE: *KDRIVE0
See Also	DRIVE, INFNC, K

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

If you enable the Disable Drive on Kill function (KDRIVE1), then when a kill command (K or !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 (although stepper motors have some detent torque).

To re-enable the drive, issue the DRIVE1 command.

If you leave the KDRIVE command in its default state (Ø, disabled), the kill function behaves in its normal manner, leaving the drive enabled.

Example:

```
KDRIVE1       ; De-energize the drive when a kill occurs
K             ; Kill is performed and drive is de-energized
```

L — Loop

Type	Loops; Program Flow Control
Syntax	<a_><!>L<i>
Units	i = number of times to loop
Range	0-999,999,999 (o = infinite loop)
Default	0
Response	L: No response; instead, this has the same function as L0
See Also	C, COMEXS, GOSUB, LN, PLOOP, PS, S, VARI

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

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> = Ø, or if no argument is specified, all the commands between L and LN will be repeated indefinitely. The loop can be stopped by issuing an immediate Kill (!K) 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.

VARI variables may be substituted for the L command value (e.g., L(VARI5)). For details, see page 24.

Example:

```
DEL PROG18     ; Delete program #18
DEF PROG18     ; Begin definition of program #18
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
LN             ; End loop
END            ; End definition
```

LDAMP — Load Damping

Type	System
Syntax	<a_><!>LDAMP<r>
Units	Rotary motor: r = Nm/rad/sec Linear motor: r = N/meter/sec
Range	Rotary motor: 0.0000 to 1.0000 : ±0.0001 Linear motor: DMEPIT (electrical pitch) dependent
Default	0.0000
Response	LDAMP: *LDAMP0.2000
See Also	DMTD, SGPRAT, SGVRAT, TGAIN, TSGSET

Product	Rev
GT	n/a
GV	1.00
GT6	n/a
GV6	1.50

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 your Gemini drive's *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

		Product	Rev
Type	EOT Limit; Drive Configuration	GT	1.02
Syntax	<a_><!>LH<i>	GV	1.00
Units	n/a	GT6	1.50
Range	i = 0 (disable both), 1 (disable positive-direction), 2 (disable negative-direction), or 3 (enable both)	GV6	1.50
Default	GT and GV: 0 (both are disabled) GT6 and GV6: 3 (both are enabled)		
Response	LH: *LH3		
See Also	DRIVE, INDEB, INFNC, INLVL, LHAD, LHADA, TAS, TER		

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

GT & GV Only: The end-of-travel limit inputs are located on the DRIVE I/O connector:

- Input #1 (pin 28) is the positive-direction end-of-travel limit.
- Input #2 (pin 29) is the negative-direction end-of-travel limit.

GT6 & GV6 Only: The "end-of-travel limit" functions are assigned with the INFNC command to two of the eight inputs on the DRIVE I/O connector. The factory default INFNC assignments are as follows (but any of the eight inputs could be re-assigned as end-of-travel inputs):

- Input #1 (pin 28) is the positive-direction end-of-travel limit (INFNC1-R).
- Input #2 (pin 29) is the negative-direction end-of-travel limit (INFNC2-S).

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 INLVL11). The INLVL command controls the active level of the limit inputs.

NOTES

- GT & GV only: If a hardware limit (either positive or negative) is encountered while limits are enabled, motion will stop, regardless of direction of motion. No subsequent motion is allowed. If limits are disabled (LH0), you are free to make a move in either direction. When a hardware limit is encountered, the drive faults and sets TAS bit 15 or 16, as well as TER bit 2. To clear these status bits, you can issue the DRIVE1 command or disable the active limit with an LH command. If a limit is active after a RESET or DRIVE1, the drive will not fault; instead, subsequent motion is allowed only in the opposite direction of the active limit.
- GT6 & GV6 only: 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.

LHAD — Hard Limit Deceleration

Type	EOT Limit
Syntax	<a_><!>LHAD<r>
Units	r = revs/sec/sec (linear motors: see page 44 for linear/rotary conversion)
Range	0.0001 – 9999.9999
Default	100.0000
Response	LHAD: *LHAD100.0000
See Also	AD, DMEPIT, DRES, INFNC, INLVL, K, KDRIVE, LH, LHADA, LSAD

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

The LHAD command determines the value at which to decelerate after a hardware end-of-travel limit has been hit. The hardware end-of-travel limits are:

GT & GV Only: The end-of-travel limit inputs are located on the DRIVE I/O connector:

- Input #1 (pin 28) is the positive-direction end-of-travel limit.
- Input #2 (pin 29) is the negative-direction end-of-travel limit.

GT6 & GV6 Only: The "end-of-travel limit" functions are assigned with the INFNC command to two of the eight inputs on the DRIVE I/O connector. The factory default INFNC assignments are as follows (but any of the eight inputs could be re-assigned as end-of-travel inputs):

- Input #1 (pin 28) is the positive-direction end-of-travel limit (INFNC1-R).
- Input #2 (pin 29) is the negative-direction end-of-travel limit (INFNC2-S).

When a drive fault, a Kill command (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.

LHADA — Hard Limit Average Deceleration

Type	Motion (S-Curve)
Syntax	<a_><!>LHADA<r>
Units	r = revs/sec/sec (linear motors: see page 44 for linear/rotary conversion)
Range	0.0001 – 9999.9999
Default	100.0000 (default is a constant deceleration ramp, where LHADA tracks LHAD; to restore tracking, set AA = 0)
Response	LHADA: *LHADA100.0000
See Also	AA, ADA, DMEPIT, INFNC, INLVL, K, LHAD, LSADA

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

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 hardware end-of-travel limits are:

GT & GV Only: The end-of-travel limit inputs are located on the DRIVE I/O connector:

- Input #1 (pin 28) is the positive-direction end-of-travel limit.
- Input #2 (pin 29) is the negative-direction end-of-travel limit.

GT6 & GV6 Only: The "end-of-travel limit" functions are assigned with the INFNC command to two of the eight inputs on the DRIVE I/O connector. The factory default INFNC assignments are as follows (but any of the eight inputs could be re-assigned as end-of-travel inputs):

- Input #1 (pin 28) is the positive-direction end-of-travel limit (INFNC1-R).
- Input #2 (pin 29) is the negative-direction end-of-travel limit (INFNC2-S).

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 53 for details on S-curve profiling.

Example:

```
LHAD10      ; Set the maximum deceleration to 10 revs/sec/sec
LHADA5      ; Set the average deceleration to 5 revs/sec/sec
```

LJRAT — System Load-to-Rotor/Forcer Inertia/Mass Ratio

Type	System
Syntax	<a_><!>LJRAT<r>
Units	Rotary motor: r = ratio of load inertia to motor rotor inertia Linear motor: r = ratio of load mass to forcer mass
Range	0.0 - 100.0
Default	0.0
Response	LJRAT: *LJRAT4
See Also	DACTDP, DELVIS, TGAIN, TSGSET

Product	Rev
GT	1.02
GV	1.00
GT6	1.50
GV6	1.50

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:

LJRAT = load inertia / motor rotor inertia

(Total system inertia = load inertia + motor rotor inertia)

Linear Motors:

LJRAT = load mass / forcer mass

(Total system mass = load mass + forcer mass)

GT/GT6 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 your Gemini drive's *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.

LN — End of Loop

Type	Loops; Program Flow Control
Syntax	<a_><!>LN
Units	n/a
Range	n/a
Default	n/a
Response	n/a
See Also	GOSUB, JUMP, L, PLN

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

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.

Example:

```
L5          ; Repeat the commands between L and LN five times
GO1         ; Start motion
LN          ; End loop
```

LS — Soft Limit Enable

Type	EOT Limit
Syntax	<a_><!>LS<i>
Units	n/a
Range	i = 0 (disable both), 1 (disable positive-direction), 2 (disable negative-direction) or 3 (enable both)
Default	0
Response	LS: *LS0
See Also	LH, LSAD, LSADA, LSNEG, LSPOS, PSET, TAS, TER

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

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 limits have been programmed (LSPOS and LSNEG), and the soft limits have been enabled (LS3), 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 Gemini drive maintains an absolute count, even though you may be programming in the incremental mode (MAØ). The soft limits will also function in incremental mode (MAØ) or continuous mode (MC1). 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 counts (soft limits are always absolute)
LSNEG-500000  ; Set soft limit negative-direction absolute position to
              ; be –500000 counts (soft limits are always absolute)
LS3           ; Enable both soft limits
LSAD100       ; Set soft limit decel to 100 revs/sec/sec
PSET0         ; Set absolute position to 0
A10           ; Set accel to 10 revs/sec/sec
V1            ; Set velocity to 1 rev/sec
D100000       ; Set distance to 100000 counts
GO1           ; Initiate motion
```

LSAD — Soft Limit Deceleration

Type	EOT Limit; Motion
Syntax	<a_><!>LSAD<r>
Units	r = revs/sec/sec (linear motors: see page 44 for linear/rotary conversion)
Range	0.0001 – 9999.9999
Default	100.0000
Response	LSAD: *LSAD100.0000
See Also	DMEPIT, DRES, LHAD, LS, LSADA, LSNEG, LSPOS

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

The LSAD command determines the value at which to decelerate after a programmed soft limit (LSPOS or LSNEG) has been hit.

Example: Refer to the soft limit enable (LS) command example.

LSADA — Soft Limit Average Deceleration

		Product	Rev
Type	EOT Limit; Motion (S-Curve)	GT	n/a
Syntax	<a_><!>LSADA<r>	GV	n/a
Units	r = revs/sec/sec (linear motors: see page 44 for linear/rotary conversion)	GT6	1.50
Range	0.0001 – 9999.9999	GV6	1.50
Default	100.0000 (default is a constant deceleration ramp, where LSADA tracks LSAD; to restore tracking, set AA = 0)		
Response	LSADA: *LSADA100.0000		
See Also	AA, ADA, DMEPIT, LHADA, LS, LSAD		

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 53 for details on S-curve profiling.

Example:
```
LSAD10       ; Set the maximum deceleration to 10 revs/sec/sec
LSADA5       ; Set the average deceleration to 5 revs/sec/sec
```

LSNEG — Soft Limit Negative Travel Range

		Product	Rev
Type	EOT Limit	GT	n/a
Syntax	<a_><!>LSNEG<r>	GV	n/a
Units	r = distance (counts)	GT6	1.50
Range	-2,147,483,648 to +2,147,483,647	GV6	1.50
Default	+0		
Response	LSNEG: *LSNEG+0		
See Also	LS, LSAD, LSADA, LSPOS, PSET		

The LSNEG command specifies the absolute position at which 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 after a successful homing operation, and can be reset using the PSET command. **Be sure to set the LSPOS value greater than the LSNEG value.**

All soft limit values entered are in absolute distance counts. 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

		Product	Rev
Type	EOT Limit	GT	n/a
Syntax	<a_><!>LSPOS<r>	GV	n/a
Units	r = distance (counts)	GT6	1.50
Range	-2,147,483,648 to +2,147,483,647	GV6	1.50
Default	+0		
Response	LSPOS: *LSPOS+0		
See Also	LS, LSAD, LSADA, LSNEG, PSET		

The LSPOS command specifies the absolute position at which 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 after a successful homing operation, and can be reset using the PSET command. **Be sure to set the LSPOS value greater than the LSNEG value.**

All soft limit values entered are in absolute distance counts. 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.

MA — Absolute/Incremental Mode Enable

Type	Motion
Syntax	<a_><!>MA<b>
Units	n/a
Range	b = 0 (incremental mode) or 1 (absolute mode)
Default	0
Response	MA: *MA0
See Also	COMEXC, D, GO, GOBUF, PSET

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

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 (MAØ), all moves are made with respect to the position at the beginning of the move. This mode is useful for repeating moves of the same distance.

In absolute mode (MA1), all moves are made with respect to the absolute zero position. The absolute zero position is equal to zero upon power up and after a successful homing operation, 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

Type	Motion
Syntax	<a_><!>MC<b>
Units	n/a
Range	b = 0 (preset mode) or 1 (continuous mode)
Default	0
Response	MC: *MC0
See Also	A, AD, COMEXC, COMEXS, D, GO, GOBUF, K, MA, PSET, S, V

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

The MC command causes subsequent moves to go a specified distance (MCØ), or a specified velocity (MC1).

In the Preset Mode (MCØ), all moves will go a specific distance. The actual distance traveled is specified by the D and MA commands.

In the Continuous Mode (MC1), all moves will go to a specific velocity (V) with the Distance (D) command establishing the direction (D+ or D-).

Motion will stop with an immediate Stop (!S) command, an immediate Kill (!K) command, or by specifying a velocity of zero (V0) 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 (see also page 44): 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 PROG11     ; Delete program #11
DEF PROG11     ; Begin definition of program #11
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 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 #11
```

NIF — End IF Statement

Type	Program Flow Control; Conditional Branching
Syntax	<a_><!>NIF
Units	n/a
Range	n/a
Default	n/a
Response	n/a
See Also	ELSE, IF

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

NIF 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 IF 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`

The `IF( ).. ELSE .. NIF` structure can be nested up to 16 levels deep.

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
```

ORES — Resolution of Step & Direction Output (GT) or Encoder Output (GV)

		Product	Rev
Type	Drive Configuration; Outputs	GT	1.02
Syntax	<a_><!>ORES<i> (does not take effect until RESET or cycle power)	GV	1.00
Units	Rotary motors: i = counts/rev Linear motors: i = counts/electrical pitch	GT6	1.50
Range	GT: 200-128000; 0 to disable GV: 200-1024000; 0 to disable	GV6	1.50
Default	GT: 0 GV: 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 command.

AUTO-SETUP for GV Drives: This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6) or Pocket Motion Planner (see page 11). 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 (page 84) for a list of auto-configured commands.

NOTE: ORES0 disables this output function.

GT: The ORES command sets the resolution of the auxiliary step & direction output (pins 14-17 on the DRIVE I/O connector).

GV: 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: 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 quad) is passed through directly.

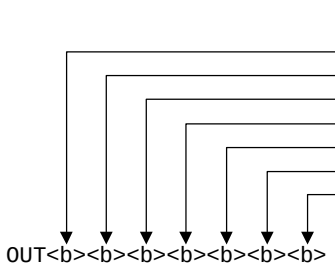
Refer to the Gemini drive's *Installation Guide* for specifications and installation instructions for the encoder output.

OUT — Output State

Type	Outputs
Syntax	<a_><!>OUT<b><b><b><b><b><b><b><b> <a_><!>OUT.<i>-<b>
Units	b = enable/disable bit i = # of the output
Range	b = 0 (off), 1 (on) or X (don't change) i = 1-7
Default	0
Response	n/a
See Also	OUTFNC, OUTLVL, POUTA, TOUT

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

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.



Output #	Pin #	GT6 & GV6 Function (OUTFNC default)
1	41	OUTFNC1-A (General-purpose)
2	43	OUTFNC2-F (Fault)
3	45	OUTFNC3-D (End-of-travel limit encountered)
4	46	OUTFNC4-E (Stall – GT6) OR OUTFNC4-G (Position error – GV6)
5	48	OUTFNC5-B (Moving/not moving)
6	49	OUTFNC6-A (General-purpose)
7	Relay	OUTFNC7-F (Fault)

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 ignore the change to that output.

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.

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).

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
```

OUTBD — Brake Output Delay

Type	Output
Syntax	<a_><!>OUTBD<i>
Units	milliseconds
Range	0 – 1000
Default	0
Response	OUTBD: *OUTBD250
See Also	DRIVE, INUFD, OUTFNC

Product	Rev
GT	1.70
GV	1.70
GT6	1.70
GV6	1.70

The OUTBD command specifies the amount of time that the fault output (OUTFNCn-F) will remain asserted after the 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 This is the complement to the INUFD command.

NOTE: When using the OUTBD command with a 6K controller, the DRFEN value for that axis should be set to 0 on the 6K. Otherwise, the 6K may not enable the drive.

OUTFNC Output Function

		Product	Rev
Type	Outputs	GT	n/a
Syntax	<a_><!>OUTFNC<i>-<c>	GV	n/a
Units	i = output #	GT6	1.50
	c = function identifier (letter)	GV6	1.50
Range	i = 1-7		
	c = A-I		
Default	(see table below)		
Response	OUTFNC1 *OUTFNC1-A PROGRAMMABLE OUTPUT - STATUS OFF		
See Also	INFNC, OUT, OUTLVL, POUTA, SMPER, TAS, TASX, TFBS, TOUT		

Use the OUTFNC command to define the function for each output. For example, the OUTFNC6-B command defines output #6 as an "Moving/Not Moving" output (identified by function "B").

Output Scan Rate: The programmable outputs are scanned once per *system update* (1 millisecond).

Status. Check the binary status report with the TOUT command.

Default Function Assignments. *(pin numbers refer to the* DRIVE I/O *connector)*

Output #	Pin #	Factory Default Function
1	41	OUTFNC1-A (general-purpose)
2	43	OUTFNC2-F (fault)
3	45	OUTFNC3-D (end-of-travel limit hit)
4	46	OUTFNC4-E (stall – GT6 only) or OUTFNC4-G (position error – GV6 only)
5	48	OUTFNC5-B (moving/not moving)
6	49	OUTFNC6-A (general-purpose)
7	Relay	OUTFNC7-F (fault)

OUTFNCi-A General Purpose Output

A *General Purpose* output is used as a standard output, where you can turn it on or off with the OUT or POUTA commands to affect processes external to the Gemini.

OUTFNCi-B Moving/Not Moving

When assigned the *Moving/Not Moving* function, the output will activate when the axis is commanded to move. As soon as the move is completed, the output will change to the opposite state.

If the target zone mode is enabled (STRGTE1), the output will not change state until the move completion criteria set with the STRGTD and STRGTV commands has been met. (For more information, refer to the *Target Zone* section on page 37.) In this manner, the Moving/Not Moving output functions as an *In Position* output.

Program example: The code example below defines outputs 1 and 2 as *General Purpose* outputs and output 3 as a *Moving/Not Moving* output. Before the motor moves 4,000 counts, output 1 turns on and output 2 turns off. These outputs remain in this state until the move is completed, then output 1 turns off and output 2 turns on. While the motor/load is moving, output 3 remains on.

```
MC0          ; Select preset positioning mode
MA0          ; Select incremental positioning mode
A10          ; Set acceleration to 10
V5           ; Set velocity to 5
D4000        ; Set distance to 4,000 counts
OUTFNC1-A    ; Set output 1 as a general purpose output
OUTFNC2-A    ; Set output 2 as a general purpose output
OUTFNC3-B    ; Set output 3 as a Moving/Not Moving output
OUT10        ; Turn output 1 on and output 2 off
GO1          ; Initiate move
OUT01        ; Turn output 1 off and output 2 on
```

OUTFNCi-C Program in Progress

A *Program in Progress* output will activate when a program is being executed. After the program is finished, the output's state is reversed. The action of executing a program is also reported with system status bit #3 (see TSS).

OUTFNCi-D End-of-Travel Limit Encountered

A *Limit Hit* output activates when a hardware or software end-of-travel limit has been encountered. For details on setting up end-of-travel limits, refer to page 29.

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 (hard limits) or the LS0 command (soft limits), but this is recommended only if the motor is not coupled to the load.

The event of encountering an end-of-travel limit is also reported with TAS bits 15-18:

Bit #15 indicates if a positive direction hardware limit was encountered.
Bit #16 indicates if a negative direction hardware limit was encountered.
Bit #17 indicates if a positive direction software limit was encountered.
Bit #18 indicates if a negative direction software limit was encountered.

OUTFNCi-E Stall Detected (GT6 only)

A *Stall* output activates when a stall is detected (also reported with TASX bit #17). For details on stall detection, refer to DSTALL.

OUTFNCi-F Fault Detected

A *Fault* output will activate 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.

OUTFNCi-G Position Error Exceeded the Max. Allowable SMPER Limit (GV6 only)

A *Position Error* 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 (TPE) 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 (DRIVE0), 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.

OUTFNCi-I Fieldbus Error

A *Fieldbus Error* output activates when the drive detects an error with the fieldbus interface. See TFBS for the specific error condition.

OUTLVL Output Active Level

Type	Outputs
Syntax	<a_><!>OUTLVL<bbbbbbb>
Units	n/a
Range	b = 0 (active low) or 1 (active high)
Default	0000000

Product	Rev
GT	1.02
GV	1.00
GT6	1.50
GV6	1.50

GT & GV: By default, the setup wizards in Motion Planner and Pocket Motion Planner change output #2 (fault output) to active high (OUTLVL0100000).

Response OUTLVL: *OUTLVL0000_000

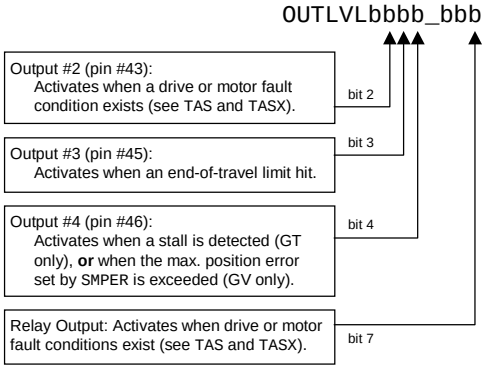
See Also OUT, OUTFNC, POUTA, TOUT

The OUTLVL command defines the active state of Gemini drive's 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. The factory default state is active low for the digital outputs, and normally open for the relay output.

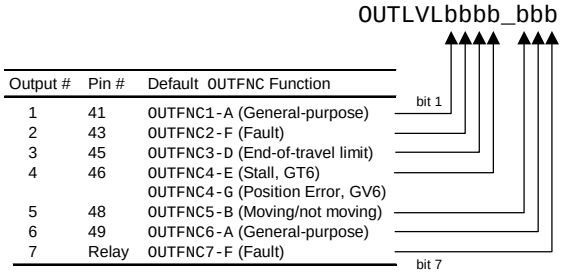
NOTE: If you do not address all of the outputs with the OUTLVL command (e.g., OUTLVL011 addresses only outputs 1-3), the remaining output levels will be set to the factory default value of "0" (active low).
GT & GV only: If you attempt to change an unused bit (bits 1, 5 & 6) to "1", the drive will respond with the "?" error prompt and will not implement the OUTLVL command.

OUTLVL bit assignments (bits are numbered 1-7 from left to right):

GT & GV:



GT6 & GV6:



Output #	Pin #	Default OUTFNC Function
1	41	OUTFNC1-A (General-purpose)
2	43	OUTFNC2-F (Fault)
3	45	OUTFNC3-D (End-of-travel limit)
4	46	OUTFNC4-E (Stall, GT6) OUTFNC4-G (Position Error, GV6)
5	48	OUTFNC5-B (Moving/not moving)
6	49	OUTFNC6-A (General-purpose)
7	Relay	OUTFNC7-F (Fault)

Bits 1, 5, & 6 are not used.

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).

PLN — End of Loop, Compiled Motion

Type	Compiled Motion
Syntax	<a_>PLN
Units	n/a
Range	n/a
Default	n/a
Response	n/a
See Also	DEF PROF, PLOOP

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

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
Syntax	<a_>PLOOP<i>
Units	i = number of times to loop
Range	0-999,999,999 (0 = infinite loop)
Default	0
Response	n/a
See Also	DEF PROF, GOBUF, PLN, PRUN PROF

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

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 cannot be nested within a profile.

WARNING: 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, after the final segment is completed, the motor will continue moving at the last segment's velocity.

The PLOOP command will consume one segment of compiled space.

An overview of Compiled Motion is provided on page 49.

Example:

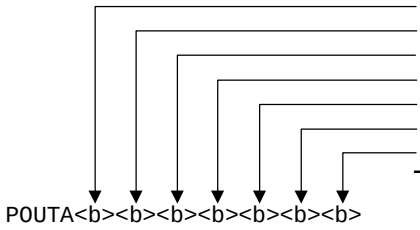
```
DEF PROF1      ; Begin definition of compiled profile #1
V1             ; Set velocity to 1 unit/sec
D1000          ; Set distance to 1000 counts
GOBUF1         ; Segment of motion sent to buffer

PLOOP3         ; Start loop of the subsequent move profile

 V10           ; Set velocity to 10 revs/sec
 D25000        ; Set distance to 25000 counts
 GOBUF1        ; First segment within loop sent to buffer

 V2            ; Set velocity to 2 revs/sec
 D1000         ; Set distance to 1000 counts
 GOBUF1        ; Second segment of motion within loop sent to buffer

 V1            ; Set velocity to 1 revs/sec
 D25000        ; Set distance to 25000 counts
 GOBUF1        ; Third segment within loop sent to buffer

PLN1           ; Close loop

V.5            ; Set velocity to 0.5 revs/sec
D100           ; Set distance to 100 counts
GOBUF1         ; Segment of motion sent to buffer (outside loop)

END            ; End definition of profile #1

PRUN PROF1     ; Execute compiled profile #1
```

POUTA Compiled Output

Type	Compiled Motion; Outputs
Syntax	<a_><!>POUTA<b><b><b><b><b><b><b> <a_><!>POUTA.<i>-<b>
Units	b = enable bit; i = output number
Range	b = 0 (off), 1 (on), or X (don't change); i = 1-7
Default	0
Response	n/a
See Also	DEF PROF, GOBUF, OUT, OUTFNC, OUTLVL, PRUN PROF

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

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, as well as the relay output (labeled "RELAY COM" and "RELAY N.O.") on the 4-pin removable connector.



Output #	Pin #	GT6 & GV6 Function (OUTFNC default)
1	41	OUTFNC1-A (General-purpose)
2	43	OUTFNC2-F (Fault)
3	45	OUTFNC3-D (End-of-travel limit encountered)
4	46	OUTFNC4-E (Stall – GT6) OR OUTFNC4-G (Position error – GV6)
5	48	OUTFNC5-B (Moving/not moving)
6	49	OUTFNC6-A (General-purpose)
7	Relay	OUTFNC7-F (Fault)

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 general-purpose output, the drive will ignore the change to that output.

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, POUTA.5-1 turns on output #5.

The POUT command consumes one segment of compiled memory.

The outputs are sampled once per "system update" (1 ms).

Example:

```
OUTFNC3-A       ; Default output function for onboard output 3
OUTFNC6-A       ; Default output function for onboard output 6
DEL PROF1       ; Delete profile #1
DEF PROF1       ; Define profile #1
D1000           ; Set distance to travel
GOBUF1          ; Motion segment
POUTA.3-1       ; Turn on output 3 when the axis travels to 1000 counts
D2000           ; New distance commanded
GOBUF1          ; Motion segment
POUTA.3-0       ; When the axis has traveled 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 profile definition

PRUN PROF1      ; Execute stored/compiled profile #1
```

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 PROF — Run a Compiled Profile

Type	Compiled Motion
Syntax	<a_><!>PRUN PROF<i> <a_><!>PROF<i>
Units	i = profile ID number
Range	1-16
Default	n/a
Response	n/a
See Also	COMEXC, DEF PROF, DEL PROF, END, GOBUF, TDIR, TMEM, TSS, (Compiled Motion overview on page 49)

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

Use the PRUN PROF command to start execution of a compiled profile previously defined/compiled with the DEF PROF command. If the axis is not ready (drive is shut down or motion is in progress), a GOBUF profile will not be executed. When execution of a pre-compiled profile begins, the axis becomes busy until motion is completed.

COMEXC1 mode must be enabled in order for command processing to continue once a motion invoking command has been initiated with PRUN PROF. If you use the PRUN PROF 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 PROF command). If in COMEXCØ mode, command processing will not continue until the motion invoking command has completed its movement.

Example:
```
DEL PROF6     ; Delete profile number 6
DEF PROF6     ; Begin definition of profile number 6
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 profile definition

PRUN PROF6    ; Execute profile number 6
```

PS — Pause Program Execution

Type	Program Flow Control
Syntax	<a_><!>PS
Units	n/a
Range	n/a
Default	n/a
Response	n/a
See Also	C, COMEXR, COMEXS, K, S, [SS], TSS

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

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 counts
GO1           ; Initiate motion
; ********************************************************************
; * NOTE:                                                            *
; * No commands after the PS command will be executed until a !C     *
; * command is received.                                             *
; ********************************************************************
```

PSET — Establish Absolute Position

		Product	Rev
Type	Motion	GT	n/a
Syntax	<a_><!>PSET<r>	GV	n/a
Units	r = counts (absolute position)	GT6	1.50
Range	-2,147,483,648 to +2,147,483,647	GV6	1.50
Default	n/a		
Response	n/a		
See Also	D, GO, HOM, INFNC, MA, MC, TPC, TPE		

Use the PSET command to offset the current absolute position to establish an *absolute position reference*. All PSET values entered are in counts.

VARI variables may be substituted for the PSET command value. For details, see page 24.

GT6: The PSET command will define the present commanded position (TPC) to be the absolute position entered.

GV6: The PSET command defines a new absolute position reference. If the drive is enabled (DRIVE1), the present commanded position (TPC) is used as the reference point. If the drive is disabled (DRIVE0), the present feedback device position (TPE) is used as the reference point.

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.
```

RE — Registration Enable

		Product	Rev
Type	Registration	GT	n/a
Syntax	<a_><!>RE<b>	GV	n/a
Units	n/a	GT6	1.50
Range	b = 0 (disable), 1 (enable)	GV6	1.50
Default	0		
Response	RE: *RE0		
See Also	COMEXC, INFNC, REG, REGLOD, TAS, TER, TRGLOT		

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) or continuous (MC1) move in progress. However, a registration (REG) move in progress cannot be interrupted by a secondary registration move.

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 Gemini is operating in the continuous command execution mode (COMEXC1).

Registration moves will not be executed while the drive is not performing a move, or while decelerating due to a stop, kill, soft limit, or hard limit.

How to Set up a Registration Move

1. Configure one of the inputs to function as a "trigger interrupt" input; this is done with the INFNCi-H command, where i is the input bit number. If more than one input is assigned the "trigger interrupt" function, only the lowest numbered input may be used to trigger a registration move (e.g., if INFNC4-H and INFNC5-H, only input #4 may be used for registration).
2. Specify the distance of the registration move with the REG command. **NOTE**: The registration move is executed using the A, AA, AD, ADA, and V values that are in effect when the REG command is entered.
3. Enable the registration function with the RE1 command. Registration is performed only when there is a non-zero distance specified with the REG command.

Registration Move Accuracy (see also Registration Move Status below)

The accuracy of the registration move distance specified with the REG command:

- GV6: ±1 encoder count
- GT6: ±50µs x velocity at the time of the registration input

RULE OF THUMB: To prevent position overshoot, make sure the REG distance is greater than 2 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 (1 ms).

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 Lockout Distance: The REGLOD command specifies what distance an axis must travel before any input assigned as a registration input will be recognized. Refer to the REGLOD 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 GO1 command.

AS.28 ..... Comparison operand for IF and WAIT conditional expressions (e.g., WAIT(AS.28=b1)).
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 Gemini drive will kill the move in progress and set axis status bit #30. This status bit is cleared with the next GO1 command.

AS.30 ..... Comparison operand for IF and WAIT conditional expressions (e.g., IF(AS.30=b1)).
TAS ..... Binary report of each status bit (bits 1-32 from left to right). See bit #30.

Input Status — Bits #1-8: 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.3=b1)).
TIN ..... Binary report of each status bit (bits 1-8 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 Gemini will branch to the error program (assigned with the ERRORP command). This status bit is cleared with the next GO1 command.

ER.10 ..... Comparison operand for IF conditional expressions (e.g., IF(ER.10=b1)).
TER ..... Binary report of each status bit (bits 1-32 from left to right). See bit #10.

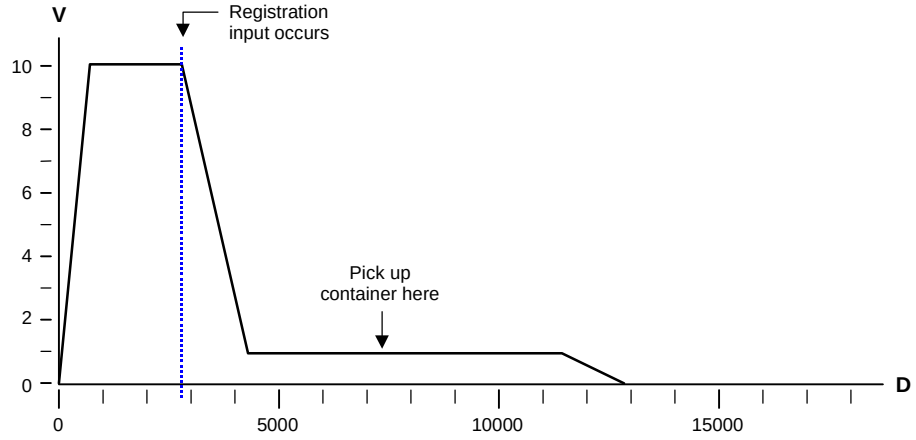
Example:

In this example, while executing a 50,000-count move, input 1 is activated and executes a registration move to slow the load's movement. An open container of volatile liquid is then placed on the conveyor belt before the conveyor stops after a registration move distance of 10,000 counts.

```
DEL PROG19    ; Delete program #19
DEF PROG19    ; Begin program definition
INFNC1-H      ; Define input #1 as a "trigger interrupt" input
A40           ; Set acceleration to 40 revs/sec/sec
AD5           ; Set deceleration to 5 revs/sec/sec
V1            ; Set velocity to 1 revs/sec
REG10000      ; Set registration distance to 10,000 counts
              ; (registration move will use the A, AD, & V values above)
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:



REG — Registration Distance

Type	Registration
Syntax	<a_><!>REG<i>
Units	i = distance (counts)
Range	0 to 2,147,483,647 (positive direction only)
Default	0 (do not make a registration move)
Response	REG: *REG0
See Also	INFNC, RE, REGLOD, TAS, TER, TIN, TRGLOT

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

The REG command specifies the distance to travel after receiving a registration input. For example, REG4000 sets up a 4000-count registration move to be initiated when the registration input is activated.

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 2 ms multiplied by the incoming velocity.

VARI variables may be substituted for the REG command value. For details, see page 24.

For additional details on Registration (including programming examples), refer to the RE command description and to the *Registration* section on page 49.

REGLOD Registration Lock-Out Distance

Type	Registration
Syntax	<a_><!>REGLOD<i>
Units	i = distance (counts)
Range	0 to +2,147,483,647
Default	0
Response	REGLOD: *REGLOD0
See Also	INFNC, RE, REG, TRGLOT

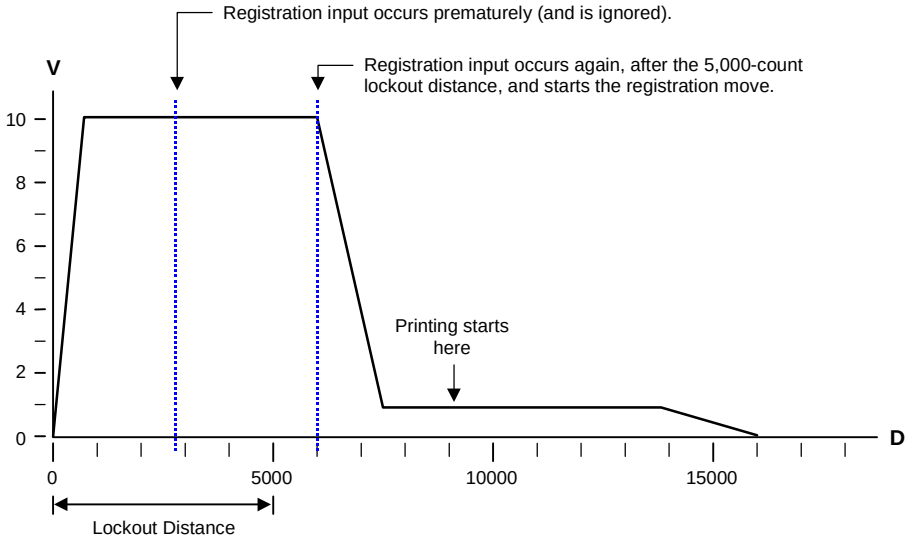
Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

The REGLOD command specifies the distance to travel before a registration input will be recognized. The lock-out distance is measured incrementally from the start of motion to the commanded position (TPC).

Example:
A print wheel uses registration to initiate each print cycle. From the beginning of motion, the Gemini should ignore all registration marks before traveling 5000 counts.

```
DEL PROG19     ; Delete program #19
DEF PROG19     ; Begin program definition
INFNC1-H       ; Define input #1 as a "trigger interrupt" input
A40            ; Set acceleration to 40 revs/sec/sec
AD5            ; Set deceleration to 5 revs/sec/sec
V1             ; Set velocity to 1 rev/sec
REG10000       ; Set registration distance to 10,000 counts
               ; (registration move will use the A, AD, & V values above)
REGLOD5000     ; 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:



RESET — Reset

		Product	Rev
Type	Communication Interface		
Syntax	<a_><!>RESET	GT	1.02
Units	n/a	GV	1.00
Range	n/a	GT6	1.50
Default	n/a	GV6	1.50
Response	RESET: (RS-232: TREV response. RS-485: No response)		
See Also	RFS, STARTP, TAS, TASX, TREV		

The RESET command affects the Gemini drive the same as cycling power, or activating the hardware Reset input (pin 3 on the DRIVE I/O connector).

GT & GV only:

All data-related commands (commands that accept binary or numeric data fields) are retained in EEPROM memory.

GT6 & GV6 only:

Programs: The drive's programs (DEF PROG) and profiles (DEF PROF) are retained in EEPROM memory. One of the programs can be assigned as the "Startup Program" (see STARTP command), which is automatically executed on power up or reset.

Commands: Some data-related commands (commands that accept binary or numeric data fields) are automatically retained in EEPROM memory, *but only if they are executed outside of a program*; these commands are denoted with "⊠" in the list on page 187. For those commands that are not automatically saved in EEPROM, they may be executed on power up or reset by placing them in a program (DEF PROG) and assigning the program as the "Startup Program" (see STARTP command). **NOTE**: Any command executed inside a program or profile is not stored in EEPROM memory.

After the reset, the drive responds with the TREV report over the RS-232 interface (if multiple drives are connected in daisy chain, the TREV report comes from unit #0 only). If RS-485 is used, no TREV is reported. Wait until you see the TREV report (or if using RS-485, wait at least 2 seconds) before communicating with the product. GT6 & GV6: If a STARTP program is assigned, the STARTP program is executed after the TREV is transmitted.

RFS — Return to Factory Settings

		Product	Rev
Type	Drive Configuration		
Syntax	<a_><!>RFS	GT	1.02
Units	n/a	GV	1.00
Range	n/a	GT6	1.50
Default	n/a	GV6	1.50
Response	n/a		
See Also	DEF PROF, DEF PROG, ERROK, RESET, TDHRS, TREV		

The RFS command returns all settings to factory default, with the exception of the ERROK prompt and the TDHRS value. GT6/GV6: All stored programs (DEF PROG) and profiles (DEF PROF) are deleted. A RESET command is automatically issued following this command; therefore, no prompt will be returned. After the reset, the drive responds with the TREV report (if multiple drives are connected, the TREV report comes from unit #0 only; if RS-485 is used, no TREV is reported).

When is the RFS event finished? The RFS process can take several seconds. During RFS, the right-hand status LED illuminates yellow. When RFS is finished, the left-hand status LED illuminates red and the drive transmits the TREV report.

Recommendation: When you complete the drive configuration procedure in Motion Planner (see page 6) or Pocket Motion Planner (see page 11), be sure to save the configuration file (GT6 & GV6 users: save your program files also) 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).

RUN PROG — Run a Program

Type	Program Definition
Syntax	<a_><!>RUN PROG<i> <a_><!>PROG<i>
Units	i = program ID number
Range	1-32
Default	n/a
Response	n/a
See Also	DEF PROG, DEL PROG, END, GOSUB, JUMP, PRUN PROF, STARTP, TDIR, TMEM

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

The RUN PROG command executes a program defined with the DEF PROG command. The RUN PROG command can be used inside a program or subroutine. The RUN PROG command functions similar to a GOSUB PROG command in that control returns to the original program (at the command immediately following RUN PROG) when the called program finishes.

Stored programs may be executed in different ways:

- Issue the RUN PROG command to start executing a program (e.g., RUN PROG3 executes program #3).
- Execute a specific program number by activating the corresponding "BCD Program Select" input (see INFNC and INSELP command descriptions).
- Branch to ("call") the program from within another program. Use one of these options:
  - Call as a subroutine with RUN PROG, PROG, or GOSUB PROG (e.g., RUN PROG3, PROG3, or GOSUB PROG3). These three commands are identical in function – they cause program flow to branch to the called program. After the called program is executed, processing returns to the calling program at the next command after the branch command. Up to 16 nested subroutines are allowed.
  - JUMP PROG (e.g., JUMP PROG3). The JUMP command branches to the specified program. All nested If conditions (IF), loops (L), and subroutines are cleared. Thus the program that the JUMP initiates will not return control to the calling program; instead, the called program will end.
- Assign the program as the "Startup Program" with the STARTP command (e.g., STARTP PROG3 assigns program #3 as the startup program). When the Gemini drive is reset or powered up, the assigned STARTP program is automatically executed.

Example:

```
DEL PROG3     ; Delete program number 3
DEF PROG3     ; Begin definition of program number 3
GO1           ; Initiate motion
END           ; End program definition
RUN PROG3     ; Execute program number 3
```

S — Stop Motion

Type	Motion
Syntax	<a_><!>S<b>
Units	n/a
Range	b = 0 (do not stop) or 1 (stop)
Default	1
Response	n/a
See Also	C, COMEXC, COMEXS, GO, K

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

The 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 Gemini drive 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.
- If S, !S or Stop input:
  - All commands in the Gemini drive's command buffer are discarded.
  - Program execution is terminated and cannot be resumed.
- If S1, or !S1:
  - All commands in the Gemini drive's command buffer are retained.
  - Program execution continues.

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

Stop Method	What Stops? Motion	Program	**Resume Motion Profile**. Allow resume with a !C command or a resume input (INFNCi-E).	**Resume Program**. Allow resume with a !C command or a resume input (INFNCi-E).	**Save Command Buffer**. Save the commands that were in the command buffer when the stop was commanded.
!S or S	Yes	Yes	Yes	Yes	Yes
!S1 or S1	Yes	No	No	No	Yes
Stop input	Yes	Yes	Yes	Yes	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 input that can be used to resume motion and program execution after a stop.

COMEXS2: Using the COMEXS2 mode, the drive responds as it does in the COMEXS0 mode, with the exception that you can still use the BCD inputs to select programs (INSELP value is retained). For more details on BCD program selection, refer to INFNC and INSELP.

SFB — Select Feedback Source

Type	Drive Configuration
Syntax	<a_><!>SFB<i>
Units	i = feedback source code
Range	1 (encoder) or 4 (resolver)
Default	1 (encoder)
Response	SFB: *SFB1
See Also	ERES, SRSET, TPE

Product	Rev
GT	n/a
GV	1.60
GT6	n/a
GV6	1.60

AUTO-SETUP: This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6) or Pocket Motion Planner (see page 11). If you did not use the configuration utility or are not using a Parker Motor, this command is set to 1 and you will have to manually set this parameter. (Refer to DMTR for a list of auto-configured commands.)

Use the SFB command to select the source for position feedback. The options are:

Options	Measurement	Setup Commands
1 (Encoder)	Encoder counts	ERES sets resolution (default is 4000 counts/rev).
4 (Resolver) *	Resolver counts	ERES sets resolution (fixed at 4096 counts/rev).

** Option 4 requires that you order the Gemini with the Resolver Option.*

SGAF — Acceleration Feedforward Gain

Type	Tuning
Syntax	<a_><!>SGAF<i>
Units	i = %
Range	0 - 500
Default	100
Response	SGAF: *SGAF100
See Also	DMTJ, LJRAT, SGENB, SGSET, SGVF, TGAIN, TSGSET

Product	Rev
GT	n/a
GV	n/a
GT6	n/a
GV6	1.50

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 GV6'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. The value of SGAF can be adjusted 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 your product's *Hardware Installation Guide*.
- 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 Gain Set

		Product	Rev
Type	Tuning	GT	n/a
Syntax	<a_><!>SGENB<i>	GV	n/a
Units	i = gain set identification number (see SGSET command)	GT6	n/a
Range	1-3	GV6	1.50
Default	n/a		
Response	n/a		
See Also	SGSET, TGAIN, TSGSET, (see list below)		

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)
- DVBW (velocity 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 your Gemini drive's *Hardware Installation Guide*.

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

		Product	Rev
Type	Tuning	GT	n/a
Syntax	<a_><!>SGINTE<b>	GV	1.00
Units	b = enable bit	GT6	n/a
Range	0 (disable), 1 (enable) or 2 (enable only at end of move)	GV6	1.50
Default	1 (enabled)		
Response	SGINTE: *SGINTE1		
See Also	DVBW, 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 your Gemini drive's *Hardware Installation Guide*.
- 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

		Product	Rev
Type	Tuning	GT	n/a
Syntax	<a_><!>SGIRAT<r>	GV	1.00
Units	r = ratio	GT6	n/a
Range	0.500 to 2.000 : ±0.001	GV6	1.50
Default	1.000		
Response	SGIRAT: *SGIRAT.775		
See Also	DIBW, DVBW, 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 your Gemini drive's *Hardware Installation Guide*.
- 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

		Product	Rev
Type	Tuning	GT	n/a
Syntax	<a_><!>SGPRAT<r>	GV	1.00
Units	r = ratio	GT6	n/a
Range	0.500 to 2.000 : ±0.001	GV6	1.50
Default	1.000		
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 your Gemini drive's *Hardware Installation Guide*.
- 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

		Product	Rev
Type	Tuning	GT	n/a
Syntax	<a_><!>SGPSIG<r>	GV	1.00
Units	r = ratio	GT6	n/a
Range	0.100 to 2.000 : ±0.001	GV6	1.50
Default	1.000		
Response	SGPSIG: *SGPSIG1.250		
See Also	DMODE, DPBW, DVBW, SGIRAT, SGPRAT, SGVRAT, TGAIN, TSGSET		

The SGPSIG command sets the ratio (frequency spread) between the velocity loop bandwidth (DVBW) 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 DMODE6, DMODE7, DMODE8, DMODE9, DMODE12, and DMODE17 (see DMODE for descriptions).

Working with servo gains.

- Servo tuning process: refer to your Gemini drive's *Hardware Installation Guide*.
- 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 Gain Set

Type	Tuning
Syntax	<a_><!>SGSET<i>
Units	i = gain set identification number
Range	1-3
Default	n/a
Response	n/a
See Also	SGENB, TGAIN, TSGSET, (see list below)

Product	Rev
GT	n/a
GV	n/a
GT6	n/a
GV6	1.50

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)
- DVBW (velocity 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 your Gemini drive's *Hardware Installation Guide*.

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

		Product	Rev
Type	Tuning	GT	n/a
Syntax	<a_><!>SGVF<i>	GV	n/a
Units	i = %	GT6	n/a
Range	0-500	GV6	1.50
Default	100		
Response	SGVF: *SGVF100		
See Also	DMTD, LDAMP, 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 GV6'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. The value of SGVF can be adjusted from zero to as high as 5 times or 500% (SGVF500) of the theoretical value.

Working with servo gains.

- Servo tuning process: refer to your Gemini drive's *Hardware Installation Guide*.
- 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
```

SGVRAT Velocity Damping Ratio

		Product	Rev
Type	Tuning	GT	n/a
Syntax	<a_><!>SGVRAT<r>	GV	1.00
Units	r = ratio	GT6	n/a
Range	0.500 to 2.000 : ±0.001	GV6	1.50
Default	1.000		
Response	SGVRAT: *SGVRAT.224		
See Also	DVBW, 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 your Gemini drive's *Hardware Installation Guide*.
- 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.

SHALL — Hall Sensor Configuration

		Product	Rev
Type	Drive Configuration		
Syntax	<a_><!>SHALL<i> (does not take effect until RESET or cycle power)	GT	n/a
Units	i = control option #	GV	1.02
Range	0 (do not invert) 1 (invert)	GT6	n/a
		GV6	1.50
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 command.

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.

SMPER — Maximum Allowable Position Error

		Product	Rev
Type	Drive Configuration		
Syntax	<a_><!>SMPER<i>	GT	n/a
Units	i = feedback device counts	GV	1.00
Range	0 to 2,147,483,647 (0 = do not monitor position error condition)	GT6	n/a
		GV6	1.50
Default	4000		
Response	SMPER: *SMPER4000		
See Also	DRES, ERES, ERROR, ERRORP, TAS, TER, TERRLG, 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 (TPC) scaled by the ratio of DRES/ERES and the actual position (TPE) 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 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.

GV only: SMPER command does not apply to GV operating in DMODE2 or DMODE4.

GV6 only: You can enable ERROR command bit #12 to continually check for the position error condition, and when it occurs to branch to the program assigned with the ERRORP program.

You can check the present position error with the TPER command.

CAUTION: If the SMPER value is set to zero (SMPER0), the position error condition is not monitored, allowing the position error to accumulate without causing a fault.

Example:

```
ERES4000      ; Set feedback resolution to 4000 counts/rev
SMPER4000     ; Set maximum allowable position error to 1 rev before
              ; a fault condition will occur.
```

SMVER — Maximum Allowable Velocity Error

Type	Drive Configuration
Syntax	<a_><!>SMVER<r>
Units	r = feedback device revs/sec (linear motors: see DMEPIT for linear/rotary conversion)
Range	0.000000 to 200.000000 : ±0.000001
Default	0.000000 (do not monitor velocity error condition)
Response	SMVER: *SMVER10.000000
See Also	DMEPIT, TASX, TVE, TVEL, TVELA

Product	Rev
GT	n/a
GV	1.00
GT6	n/a
GV6	1.50

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.

GV only: SMVER command does not apply to GV operating in DMODE2.

SRSET — Resolver Offset Angle

Type	Drive Configuration
Syntax	<a_><!>SRSET<r>
Units	r = angle in degrees
Range	-180.0 to +180.0 : ±0.1 (no value = use auto align if in DMODE11)
Default	0
Response	SRSET: (returns the calculated value only if in DMODE11)
See Also	DMODE, SFB, TSROFF

Product	Rev
GT	n/a
GV	1.60
GT6	n/a
GV6	1.60

AUTO-SETUP: This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6) or Pocket Motion Planner (see page 11). 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.

STARTP Start-Up Program

		Product	Rev
Type	Program Definition	GT	n/a
Syntax	<a_><!>STARTP PROG<i>	GV	n/a
Units	i = program ID #	GT6	1.50
Range	0, 1-32 (0 clears the existing program assignment)	GV6	1.50
Default	0 (no program is assigned)		
Response	STARTP: *STARTP PROG3		
See Also	DEF PROG, DEL PROG, RESET		

The STARTP PROG command assigns an existing program to be executed automatically when the Gemini 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 PROG12 assigns program #12 (previously defined with DEF PROG12) as the startup program.

If the program that is identified as the STARTP program is deleted with the DEL PROG command, the STARTP assignment is automatically cleared.

Example:

```
DEL PROG12    ; Delete program #12
DEF PROG12    ; Begin definition of program #12
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 PROG12 ; Assign program #12 as the startup program
```

STRGTD Target Distance Zone

		Product	Rev
Type	Target Zone	GT	n/a
Syntax	<a_><!>STRGTD<i>	GV	n/a
Units	i = distance (counts)	GT6	n/a
Range	0-999,999,999	GV6	1.50
Default	50		
Response	STRGTD: *STRGTD50		
See Also	ERROR, ERRORP, STRGTE, STRGTT, STRGTV, TAS, TER, 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.

When using the Target Zone 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 TAS) 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 Gemini drive detects an error (see TAS 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 (ERRORxxxxxxxxxx1), 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 page 37.*

Example (see STRGTE):

STRGTE Enable Target Zone Settling Mode

		Product	Rev
Type	Target Zone	GT	n/a
Syntax	<a_><!>STRGTE<b>	GV	n/a
Units	n/a	GT6	n/a
Range	b = 0 (disable) or 1 (enable)	GV6	1.50
Default	0		
Response	STRGTE: *STRGTE0		
See Also	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 distance zone defined by STRGTD and within the velocity zone defined by STRGTV) before motion can be determined complete. Axis status bit #24 (see TAS) 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 Gemini drive detects an error (see TAS 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 (ERRORxxxxxxxxxx1), 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 page 37.*

Example:

```
STRGTD5       ; Set the distance target zone to +/-5 counts
STRGTV.01     ; Set the velocity target zone to <= 0.01 revs/sec
STRGTT10      ; Set the timeout period to 10 milliseconds
STRGTE1       ; Enable the target zone criteria
; Given these target zone commands, a move with a distance of 8,000 counts
; (D8000) must end up between position 7,995 and 8,005 and settle down
; to <=0.01 revs/sec within 10 ms after the commanded profile is complete.
```

STRGTT Target Settling Timeout Period

		Product	Rev
Type	Target Zone	GT	n/a
Syntax	<a_><!>STRGTT<i>	GV	n/a
Units	i = milliseconds	GT6	n/a
Range	0-5000	GV6	1.50
Default	1000		
Response	STRGTT: *STRGTT1000		
See Also	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 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 TAS) 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 Gemini drive detects an error (see TAS 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 (ERRORxxxxxxxxxx1), 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 page 37.*

Example (see STRGTE):

STRGTV — Target Velocity Zone

		Product	Rev
Type	Target Zone	GT	n/a
Syntax	<a_><!>STRGTV<r>	GV	n/a
Units	r = revs/sec (linear motors: see page 44 for linear/rotary conversion)	GT6	n/a
Range	0.0000-200.0000	GV6	1.50
Default	1.0000		
Response	STRGTV: *STRGTV1.0000		
See Also	DMEPIT, ERROR, ERRORP, STRGTD, STRGTE, STRGTT, TAS, TER, TSTLT		

This command sets the target velocity zone for use in the Target Zone Settling Mode. The target velocity zone is a velocity range that the load must be within before motion is considered complete.

When using the Target Zone 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 less than or equal to the velocity defined by STRGTV) before motion can be determined complete. Axis status bit #24 (see TAS) 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 Gemini drive detects an error (see TAS 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 (ERRORxxxxxxxxxx1), 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 page 37.*

Example: (see STRGTE)

T — Time Delay (Dwell)

		Product	Rev
Type	Program Flow Control	GT	n/a
Syntax	<a_><!>T<r>	GV	n/a
Units	r = seconds	GT6	1.50
Range	0.001-999.999	GV6	1.50
Default	n/a		
Response	n/a		
See Also	GOWHEN, PS, TSS, VARI, 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 1 millisecond (ms).

VARI variables may be substituted for the T command value (e.g., T(VARI5)). For details, see page 24.

Example:

```
T2            ; Wait 2 seconds before executing TPE command
TPE           ; Transfer position of all encoders to the terminal
```

TACC — Transfer Commanded Acceleration

		Product	Rev
Type	Transfer	GT	n/a
Syntax	<a_><!>TACC	GV	n/a
Units	revs/sec/sec (linear motors: see DMEPIT for linear/rotary conversion)	GT6	1.50
Range	n/a	GV6	1.50
Default	n/a		
Response	TACC: *TACC100		
See Also	A, AD, DMEPIT, HOMA, LHAD, LSAD, TACCA, TVEL		

TACC reports the commanded acceleration.

TACCA — Transfer Actual Acceleration

		Product	Rev
Type	Transfer		
Syntax	<a_><!>TACCA	GT	n/a
Units	revs/sec/sec	GV	n/a
	(linear motors: see DMEPIT for linear/rotary conversion)	GT6	n/a
Range	n/a	GV6	1.50
Default	n/a		
Response	TACCA: *TACCA100		
See Also	A, AD, DMEPIT, HOMA, LHAD, LSAD, TACC, TPE, TVELA		

TACCA reports the actual acceleration as calculated by using the feedback device.

TANI — Transfer Analog Input Voltage

		Product	Rev
Type	Transfer		
Syntax	<a_><!>TANI	GT	1.02
Units	Volts	GV	1.02
Range	-10V to +10V	GT6	1.50
Default	n/a	GV6	1.50
Response	TANI: *TANI+9.8		
See Also	DCMDZ		

TANI reports the actual voltage received on pins #23 and #24 of the **DRIVE I/O** connector. The TANI value is not affected by the DCMDZ offset.

This value is also available as a system variable for use in variable assignments, conditional expression, etc. The ANI value is the actual output of the 12-bit A/D converter, ranging from –2048 to 2047. Any scaling must be done using variables.

Example:

```
VARI5 = ANI*49 ; Read the analog input value and scale it so that VARI5 = 100,000 for 10V.
```

TAS — Transfer Axis Status

		Product	Rev
Type	Transfer		
Syntax	<a_><!>TAS	GT	1.02
Units	n/a	GV	1.00
Range	n/a	GT6	1.50
Default	n/a	GV6	1.50
Response	TAS: *TAS 0000_0000_0000_0000_0000_0000_0000_0000		
See Also	TASX		

The TAS command returns the status of the axis.

*TASbbbb_bbbb_bbbb_bbbb_bbbb_bbbb_bbbb_bbbb

Bit 1 ↑ Bit 32 ↑

Bit # (left to right)	Function (1 = Yes; Ø = No)	GT	GV	GT6	GV6	To Clear This Bit:
1	Motion is commanded. (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	RESERVED	--	--	--	--	----------

(continued on next page)

Bit # (left to right)	Function (1 = Yes; Ø = No)	GT	GV	GT6	GV6	To Clear This Bit:
9	RESERVED	--	--	--	--	----------
10	RESERVED	--	--	--	--	----------
11	RESERVED	--	--	--	--	----------
12	Stall detected if ESK1 enabled. Also sets TER bit 1. GT only: This is a "Fault Condition" (see note below *). GT6 only: This is not a "fault condition"; instead, it performs a kill (!K).	X	--	X	--	GT: DRIVE1 GT6: GO
13	Drive shut down. **NOTE**: 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	X	X	DRIVE1
14 *	Drive Faults occurred. Check the TASX response to identify which fault(s) occurred.	X	X	X	X	DRIVE1
15	Positive-direction hardware limit hit. GT/GV only: This is a "Fault Condition" (see note below *). GT6/GV6 only: This is not a "fault condition".	X	X	X	X	GT/GV: DRIVE1 or LHØ. GT6/GV6: LHØ or GO in opposite direction.
16	Negative-direction hardware limit hit. GT/GV only: This is a "Fault Condition" (see note below *). GT6/GV6 only: This is not a "fault condition".	X	X	X	X	GT/GV: DRIVE1 or LHØ. GT6/GV6: LHØ or GO in opposite direction.
17	Positive-direction Software Limit (LSPOS) Hit.	--	--	X	X	GT6/GV6: LSØ or GO in opposite direction.
18	Negative-direction Software Limit (LSNEG) Hit.	--	--	X	X	GT6/GV6: LSØ or GO in opposite direction.
19	RESERVED	--	--	--	--	---------
20	RESERVED	--	--	--	--	---------
21	RESERVED	--	--	--	--	---------
22	RESERVED	--	--	--	--	---------
23 *	Position error exceeded (SMPER).	--	X	--	X	DRIVE1
24	In Target Zone (defined with STRGTD & STRGTV). This bit is set only after the *successful completion* 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 a TRGFN trigger interrupt event or a GOWHEN condition. This bit is cleared when the TRGFN trigger is activated or the GOWHEN condition evaluates true, or if a stop (!S) or a Kill (!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	RESERVED	--	--	--	--	---------
30	Pre-emptive (OTF) GO or Registration profile not possible.	--	--	X	X	GO1
31	RESERVED	--	--	--	--	---------
32	RESERVED	--	--	--	--	---------

* **FAULT CONDITIONS**: If one or more of these conditions exist, the drive automatically disables (DRIVE0), it activates output #2 (pin 43 on the DRIVE I/O connector), 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
Syntax	<a_><!>TASF
Units	n/a
Range	n/a
Default	n/a
Response	TASF: (see example below)
See Also	TAS, TASX, TASXF, TER, TERF, TSS, TSSF

Product	Rev
GT	1.70
GV	1.70
GT6	1.70
GV6	1.70

The TASF command returns a text-based status report. This is an alternative to the binary report (TAS). Example TASF response:

```
*TASF
Bit  1: Moving                          NO
Bit  2: Direction NEG                   NO
Bit  3: Accelerating                    NO
Bit  4: At Commanded Velocity           NO
Bit  5: Home Successful                 NO
Bit  6: Mode Absolute                   NO
Bit  7: Mode Continuous                 NO
Bit  8: Reserved                        NO
Bit  9: Reserved                        NO
Bit 10: Reserved                        NO
Bit 11: Reserved                        NO
Bit 12: Stall Detected                  NO
Bit 13: Drive Shutdown                  NO
Bit 14: Drive Faulted                   NO
Bit 15: POS Hardware Limit Hit          NO
Bit 16: NEG Hardware Limit Hit          NO
Bit 17: POS Software Limit Hit          NO
Bit 18: NEG Software Limit Hit          NO
Bit 19: Reserved                        NO
Bit 20: Reserved                        NO
Bit 21: Reserved                        NO
Bit 22: Reserved                        NO
Bit 23: Position Error Limit Exceeded   NO
Bit 24: In Target Zone                  YES
Bit 25: Target Zone Timeout             NO
Bit 26: GOWHEN Is Pending               NO
Bit 27: Reserved                        NO
Bit 28: Registration Move Commanded     NO
Bit 29: Reserved                        NO
Bit 30: Move Not Allowed                NO
Bit 31: Reserved                        NO
Bit 32: Reserved                        NO
```

TASX — Transfer Extended Axis Status

Type	Transfer
Syntax	<a_><!>TASX
Units	n/a
Range	n/a
Default	n/a
Response	TASX: *TASX 0000_0000_0000_0000_0000_0000_0000_0000
See Also	DCLRLR, DRIVE, RESET, TAS, TCS, TER

Product	Rev
GT	1.02
GV	1.00
GT6	1.50
GV6	1.50

TASX reports the axis status conditions. *TASXbbbb_bbbb_bbbb_bbbb_bbbb_bbbb_bbbb_bbbb

Bit #	Function (1 = Yes; Ø = No)	GT	GV	GT6	GV6	To Clear This Status Bit:
1 *	Motor temperature fault (hardware switch in the motor).	X	X	X	X	DRIVE1
2 *	Low voltage fault	X	X	X	X	DRIVE1
3 *	Drive over-temperature fault (hardware thermal sensor in drive). See TDTEMP.	X	X	X	X	DRIVE1
4	RESERVED	--	--	--	--	----------
5 *	Resolver failed (disconnected).	--	X	--	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	X	X	Resolve error condition (see TCS) and issue RESET or cycle power
8	Incoming steps during startup or drive enable (DRIVE1). **NOTE**: If operating in the FLTSTP1 mode, incoming steps will cause a "fault condition" – see * note below.	X	X	--	--	DRIVE1
9 *	Velocity error limit (SMVER value) has been exceeded.	--	X	--	X	DRIVE1
10 *	Bridge fault (hardware signal from the bridge)	X	X	X	X	RESET or cycle power
11 *	Bridge temperature fault (this is a software control).	--	X	--	X	DRIVE1
12 *	Over-voltage.	--	X	--	X	RESET or cycle power
13-16	RESERVED	--	--	--	--	----------
17	Stall detected, regardless of ESK setting.	X	--	X	--	DRIVE1
18	GV/GV6: Override mode was invoked. GT/GT6: Commanded velocity exceeds DMVLIM limit.	X	X	X	X	DCLRLR or RESET or cycle power
19	Bridge is in foldback mode.	--	X	--	X	DCLRLR or RESET or cycle power
20	Power Dissipation Circuit has been active – not applicable to GV-U3, GV-U6, and GV-U12 drives	X	X	X	X	DCLRLR or RESET or cycle power
21 *	Bad Hall state detected. (Use THALL for diagnostics.)	--	X	--	X	RESET or cycle power
22 *	Unrecognized hardware — consult factory	X	X	X	X	RESET or cycle power
23 *	User Fault input activated (GT/GV: input #3; GT6/GV6: INFNCi-F)	X	X	X	X	GT/GV: DRIVE1 GT6/GV6: Deactivate the input
24	Keep Alive active (User supplied +24VDC)	X	X	X	X	----------
25 *	Power dissipation circuit fault (excessive power dissipation)	X	X	X	X	DRIVE1
26	RESERVED	--	--	--	--	----------
27	Fieldbus Error	--	--	X	X	Issue RESET or cycle power
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	X	X	Resolve error condition (see TCS) and issue RESET or cycle power.
29 *	ORES failure (encoder output or step & direction output exceeds the maximum output frequency).	X	X	X	X	DRIVE1
30 *	Motor Thermal model fault.	--	X	--	X	Let the motor cool, then issue DRIVE1.
31	Commanded torque/force is at limit (TTRQ = DMTLIM).	--	X	--	X	DCLRLR or RESET or cycle power
32	RESERVED	--	--	--	--	---------

* **FAULT CONDITIONS**: If one or more of these conditions exist, the drive automatically disables (DRIVE0), it activates output #2 (pin 43 on the DRIVE I/O connector), 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
Syntax	<a_><!>TASXF
Units	n/a
Range	n/a
Default	n/a
Response	TASXF: (see example below)
See Also	TAS, TASF, TASX, TER, TERF, TSS, TSSF

Product	Rev
GT	1.70
GV	1.70
GT6	1.70
GV6	1.70

The TASXF command returns a text-based status report. This is an alternative to the binary report (TASX). Example TASXF response:

```
*TASX
Bit  1: Motor Temperature Switch          NO
Bit  2: Low Voltage Fault                 NO
Bit  3: Drive Termperature Fault          NO
Bit  4: Reserved                          NO
Bit  5: Feedback Failure                  NO
Bit  6: Reserved                          NO
Bit  7: Motor Configuration Error         NO
Bit  8: Incoming Steps at Enable          NO
Bit  9: Velocity Error Limit Exceeded     NO
Bit 10: Bridge Hardware Fault             NO
Bit 11: Bridge Temperature Fault          NO
Bit 12: Drive Overvoltage                 NO
Bit 13: Reserved                          NO
Bit 14: Reserved                          NO
Bit 15: Reserved                          NO
Bit 16: Reserved                          NO
Bit 17: Stall Detected                    NO
Bit 18: Velocity Override                 NO
Bit 19: Bridge Foldback                   NO
Bit 20: Power Regeneration Active         NO
Bit 21: Bad Hall State                    NO
Bit 22: Unrecognized Hardware             NO
Bit 23: User Fault                        NO
Bit 24: Keep-Alive Mode                   NO
Bit 25: Power Regeneration Fault          NO
Bit 26: Reserved                          NO
Bit 27: Fieldbus Error                    NO
Bit 28: Motor Configuration Warning       NO
Bit 29: ORES Failure                      NO
Bit 30: Motor Thermal Model Fault         NO
Bit 31: Maximum Torque Commanded          NO
Bit 32: Reserved                          NO
```

TCS — Transfer Configuration Status

Type	Transfer
Syntax	<a_><!>TCS
Units	Fault/Warning code (see table below)
Range	n/a
Default	n/a
Response	TCS: *TCS46
See Also	DMTR, TASX

Product	Rev
GT	1.02
GV	1.00
GT6	1.50
GV6	1.50

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 following table).

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
-32158	Fault	GV6-PB GT6-PB	Too many bytes in FBPIC	Redefine FBPIC	Bit 27
-32168	Fault	GV6-PB GT6-PB	Too many bytes in FBPOC	Redefine FBPOC	Bit 27
-32228	Fault	GV/GV6	ERES ≠ 4096 during SFB4	Change ERES to 4096 (ERES4096).	Bit 7
-32238	Fault	GV/GV6	Internal position loop gains ≤ 0.	Increase one or all of: DPBW, SGPRAT, SGPSIG. Recheck DMTKE, DMTJ, LJRAT, DMTD, LDAMP.	Bit 7
-32248	Fault	GV/GV6	Internal velocity loop gains ≤ 0.	Increase one or all of: DVBW, SGVRAT. Recheck DMTKE, DMTJ, LJRAT, DMTD, LDAMP.	Bit 7
-32259	Fault	GV/GV6	Internal current loop gains < 0.	Increase DIBW and/or re-check DMTRES, DMTLMN, DMTLMX and SGIRAT.	Bit 7
-32367	Fault	GT/GT6	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	All	DMTJ = 0	Enter non zero value.	Bit 7
-32714	Fault	GV/GV6	DMTLMX = 0	Enter non zero value.	Bit 7
-32715	Fault	GV/GV6	DMTLMN = 0	Enter non zero value.	Bit 7
-32718	Fault	GV/GV6	DMTW = 0	Enter non zero value.	Bit 7
-32723	Fault	All	DPOLE = 0	Enter non zero value.	Bit 7
-32725	Fault	All	DMTRES = 0	Enter non zero value.	Bit 7
-32726	Fault	GT/GT6	DMTIND = 0	Enter non zero value.	Bit 7
-32727	Fault	GV/GV6	DMTKE = 0	Enter non zero value.	Bit 7
-32729	Fault	GT/GT6	DMTSTT = 0	Enter non zero value.	Bit 7
40	Warning	All	DMTIC = 0	Enter non zero value.	Bit 28
46	Warning	GT/GT6	DIGNA = 0	Enter non zero value.	Bit 28
47	Warning	GT/GT6	DIGNB = 0	Enter non zero value.	Bit 28
48	Warning	GT/GT6	DIGNC = 0	Enter non zero value.	Bit 28
49	Warning	GT/GT6	DIGND = 0	Enter non zero value.	Bit 28
51	Warning	GV/GV6	DMTIP = 0	Enter non zero value.	Bit 28
400	Warning	All	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	GV/GV6	DMTLIM is too high for the drive's peak current rating.	Lower DMTLIM and/or recheck DMTKE.	Bit 28
503	Warning	GV/GV6	DMTIP is set too high. Drive is set to maximum value.	Lower DMTIP.	Bit 28
551	Warning	GV/GV6	Notch filter calculation error.	Check DNOTAF, DNOTBF, DNOTAQ, DNOTBQ.	Bit 28
560	Warning	GV/GV6	Lead frequency (DNOTLD) < 20 Hz. The last good value is used. (0 = disable)	Increase DNOTLD.	Bit 28
561	Warning	GV/GV6	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

* All GV6 conditions apply to GV6-PB; All GT6 conditions apply to GT6-PB.

TDHRS — Transfer Operating Hours

Type	Transfer
Syntax	<a_><!>TDHRS
Units	Lifetime operating hours (resolution is ¼ or 0.25 hours)
Range	Hour counter rolls over at 16384.00 hours
Default	n/a
Response	TDHRS: *TDHRS16.5
See Also	RESET, TERRLG

Product	Rev
GT	1.02
GV	1.00
GT6	1.50
GV6	1.50

The TDHRS command reports the lifetime number of hours (to the nearest ¼ hour) that the Gemini drive has had power applied (AC mains or 24 Volt keep-alive).

NOTE: The hour count rolls over at 16,384.00 hours.

TDICNT — Transfer Continuous Current Rating

Type	Transfer
Syntax	<a_><!>TDICNT
Units	Amps peak
Range	n/a
Default	n/a
Response	TDICNT: *TDICNT10
See Also	DMTIC, DMTIP, TDIMAX

Product	Rev
GT	1.02
GV	1.00
GT6	1.50
GV6	1.50

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
Syntax	<a_><!>TDIMAX
Units	Amps peak
Range	n/a
Default	n/a
Response	TDIMAX: *TDIMAX10
See Also	DMTIP, DMTIP, TDICNT

Product	Rev
GT	n/a
GV	1.00
GT6	n/a
GV6	1.50

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 Programs/Profiles Stored in Memory

Type	Transfer
Syntax	<a_><!>TDIR
Units	n/a
Range	n/a
Default	n/a
Response	TDIR: *PROG1
See Also	DEF PROF, DEF PROG

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

TDIR reports the names of the programs (DEF PROG) and profiles (DEF PROF) stored in the Gemini drive's EEPROM memory. **NOTE**: TDIR is not allowed inside a program.

TDTEMP Transfer Drive Temperature

Type	Transfer
Syntax	<a_><!>TDTEMP
Units	Degrees C
Range	n/a
Default	n/a
Response	TDTEMP: *TDTEMP50
See Also	DMONAV, DMONBV, TERRLG

Product	Rev
GT	1.02
GV	1.00
GT6	1.50
GV6	1.50

The Gemini drive has two internal temperature sensors. One is located on the drive's power block, the other on the DSP board. TDTEMP reports the higher temperature from the two sensors. The drive's overtemperature fault takes effect at 80°C (or 90°C for GV/GV6-H20).

TDVBUS Transfer Bus Voltage

Type	Transfer
Syntax	<a_><!>TDVBUS
Units	Volts
Range	n/a
Default	n/a
Response	TDVBUS: *TDVBUS170
See Also	

Product	Rev
GT	1.02
GV	1.00
GT6	1.50
GV6	1.50

GT/GT6: TDVBUS reports the currently measured bus voltage of the drive.
GV/GV6: TDVBUS reports the nominal DC bus voltage (see calculation below) when the drive is first powered up.

$$169\text{V } (= \sqrt{2} * 120) \quad \text{or} \quad 339\text{V } (= \sqrt{2} * 240)$$

TER — Transfer Error Status

		Product	Rev
Type	Transfer	GT	1.02
Syntax	<a_><!>TER	GV	1.00
Units	n/a	GT6	1.50
Range	n/a	GV6	1.50
Default	n/a		
Response	TER: *TER0000_0000_0000_0000_0000_0000_0000_0000		
See Also	ERROR, ERRORP, ESK, LH, SMPER, TASX, TERRLG		

The TER command returns the status of the 32 error bits. The TER status command reports a binary bit report.

```
*TERbbbb_bbbb_bbbb_bbbb_bbbb_bbbb_bbbb_bbbb
  Bit 1                                Bit 32
```

Bit # (left to right)	Function (1 = Yes; Ø = No)	GT	GV	GT6	GV6	To Clear This Bit:
1	Stall detected if ESK1 enabled.	X	--	X	--	GT: DRIVE1 GT6: GO
2	Hardware end-of-travel limit hit. This condition is detectable only when the limits are enabled (LH3).	X	X	X	X	GT/GV: DRIVE1 or LHØ. GT6/GV6: LHØ 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	GT6/GV6: LSØ or GO in opposite direction.
4	Drive fault — refer to TASX to ascertain which fault(s) occurred.	X	X	X	X	Varies with Fault – refer to TASX
5	Commanded Kill or Commanded Stop (a K, !K, S, or !S command was executed). (see also, ERROR and ERRORP command descriptions)	--	--	X	X	Cleared when the ERRORP program is executed (requires ERROR bit #5 to be set)
6	Kill input activated. An input defined as a Kill input (INFNCi-C) was activated.	--	--	X	X	Deactivate input
7	User Fault input activated. GT/GV: Input #3. GT6/GV6: Input must be assigned the user fault function (INFNCi-F).	X	X	X	X	GT/GV: DRIVE1 GT6/GV6: 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	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	--	X	DRIVE1
13 – 18	RESERVED	--	--	--	--	---
19	Fieldbus error	--	--	X	X	Reset the drive
20 – 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 Gemini 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 on page 26.

TERF — Transfer Error Status (full-text report)

Type	Transfer
Syntax	<a_><!>TERF
Units	n/a
Range	n/a
Default	n/a
Response	TERF: (see example below)
See Also	TAS, TASF, TASX, TASXF, TER, TSS, TSSF

Product	Rev
GT	1.70
GV	1.70
GT6	1.70
GV6	1.70

The TERF command returns a text-based status report. This is an alternative to the binary report (TER). Example TERF response:

```
*TERF
Bit  1: Stall Detected                 NO
Bit  2: Hardware Limit Hit             NO
Bit  3: Software Limit Hit             NO
Bit  4: Drive Fault                    NO
Bit  5: Commanded Kill or Stop         NO
Bit  6: Kill Input Active              NO
Bit  7: User Fault Input Active        NO
Bit  8: Stop Input Active              NO
Bit  9: Enable Input Open              NO
Bit 10: Move Not Possible              NO
Bit 11: Target Zone Timeout            NO
Bit 12: Position Error Limit Exceeded  NO
Bit 13: Reserved                       NO
Bit 14: Reserved                       NO
Bit 15: Reserved                       NO
Bit 16: Reserved                       NO
Bit 17: Reserved                       NO
Bit 18: Reserved                       NO
Bit 19: Fieldbus Error                 NO
Bit 21: Reserved                       NO
Bit 22: Reserved                       NO
Bit 23: Reserved                       NO
Bit 24: Reserved                       NO
Bit 25: Reserved                       NO
Bit 26: Reserved                       NO
Bit 27: Reserved                       NO
Bit 28: Reserved                       NO
Bit 29: Reserved                       NO
Bit 30: Reserved                       NO
Bit 31: Reserved                       NO
Bit 32: Reserved                       NO
```

TERRLG Transfer Error Log

		Product	Rev
Type	Transfer	GT	1.02
Syntax	<a_><!>TERRLG	GV	1.00
Units	n/a	GT6	1.50
Range	n/a	GV6	1.50
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, TAS, TASX, TDHRS, TDTEMP, TMTEMP		

The error log is updated every time an error occurs. 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 - GV only)

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.

TGAIN Transfer Active Gains

		Product	Rev
Type	Transfer	GT	n/a
Syntax	<a_><!>TGAIN	GV	1.03
Units	n/a	GT6	n/a
Range	n/a	GV6	1.50
Default	n/a		
Response	(list of all gain values)		
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)
- DVBW (velocity 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
Syntax	<a_><!>THALL
Units	n/a
Range	1-6 (0 or 7 is a fault condition – see TASX bit #21)
Default	n/a
Response	THALL: *THALL6
See Also	SHALL, TASX

Product	Rev
GT	n/a
GV	1.50
GT6	n/a
GV6	1.50

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 *GV Series Drive Hardware Installation Guide* (p/n 88-017791-01) section on using non-Compumotor motors.

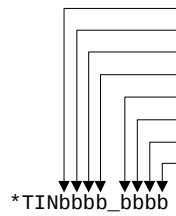
TIN — Transfer Input Status

Type	Transfer
Syntax	<a_><!>TIN
Units	n/a
Range	n/a
Default	n/a
Response	TIN: *TIN0000_0000
See Also	INDEB, INFNC, INLVL, TINO

Product	Rev
GT	1.02
GV	1.00
GT6	1.50
GV6	1.50

The TIN command returns the status (active or inactive) of the digital inputs on the DRIVE I/O connector.

TIN response (bits are numbered 1-8 from left to right):



Input #	Pin #	GT & GV Function (fixed)	GT6 & GV6 Function (INFNC default)
1	28	Positive end-of-travel limit	INFNC1-R (Positive end-of-travel limit)
2	29	Negative end-of-travel limit	INFNC2-S (Negative end-of-travel limit)
3	31	User fault	INFNC3-T (Home limit)
4	34	*(input not available)*	INFNC4-H (Trigger interrupt)
5	35	*(input not available)*	INFNC5-A (General-purpose)
6	37	*(input not available)*	INFNC6-A (General-purpose)
7	38	*(input not available)*	INFNC7-A (General-purpose)
8	39	*(input not available)*	INFNC8-A (General-purpose)

Relationships:

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 your drive's *Hardware Installation Guide* for wiring instructions). Pin 27 is the pull up for inputs 1-3, and pin 33 is the pull up for inputs 4-8.

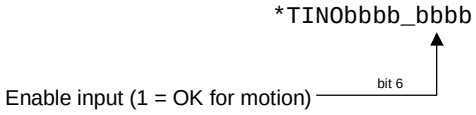
TINO — Transfer Other Input Status

Type	Transfer
Syntax	<a_><!>TINO
Units	(bit #6 = ENABLE input. 1 = OK for motion)
Range	n/a
Default	n/a
Response	TINO: *TINO0000_0100
See Also	TIN

Product	Rev
GT	1.02
GV	1.00
GT6	1.50
GV6	1.50

The Transfer Other Input Status (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):



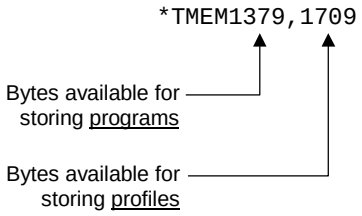
NOTE: When an over-voltage condition occurs (TASX.12 = 1), TINO will respond with TINO0000_0000, regardless of the state of the enable switch.

TMEM — Transfer Memory Usage

Type	Transfer
Syntax	<a_><!>TMEM
Units	n/a
Range	n/a
Default	n/a
Response	TMEM: *TMEM2996,1598
See Also	DEF PROG, DEF PROF, TDIR, TPROG

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

The TMEM command reports the amount of available memory for storing *programs* (programs defined with DEF PROG) and for storing *compiled profiles* (profiles defined/compiled with DEF PROF). To ascertain which programs and profiles are stored in EEPROM memory, use the TDIR command.



TMTEMP — Transfer Motor Temperature

Type	Transfer
Syntax	<a_><!>TMTEMP
Units	Degrees C
Range	n/a
Default	n/a
Response	TMTEMP: *TMTEMP45
See Also	DMONAV, DMONBV, DMTRWC, DMTTCM, DMTTCW, TASX, TERRLG

Product	Rev
GT	n/a
GV	1.00
GT6	n/a
GV6	1.50

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 DMTMAX, assuming the ambient temperature is DMTAMB.

If you are using a non-Parker motor, the TMTEMP value depends on customer-supplied values for the DMTRWC, DMTTCM and the DMTTCW parameters.

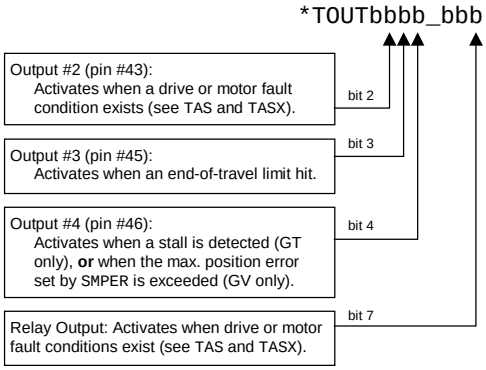
TOUT — Transfer Output Status

Type	Transfer
Syntax	<a_><!>TOUT
Units	n/a
Range	n/a
Default	n/a
Response	TOUT: *TOUT0000_000
See Also	OUTFNC, OUTLVL, SMPER, TAS, TASX, TIN, TINO

Product	Rev
GT	1.02
GV	1.00
GT6	1.50
GV6	1.50

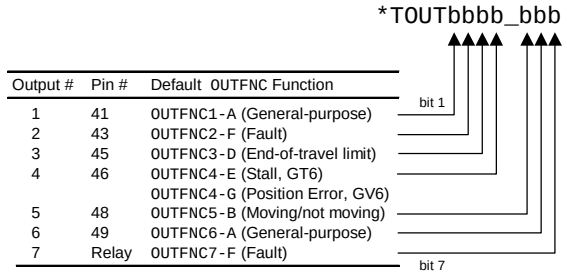
The Transfer Output Status (TOUT) command returns the present status (active or inactive) of the outputs on the DRIVE I/O connector, as well as the dry contact relay output (labeled "RELAY COM" and "RELAY N.O.") on the 4-pin removable connector. The TOUT response (bits are numbered 1-7 from left to right) is:

GT & GV:



Bits 1, 5, & 6 are not used.

GT6 & GV6:



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).

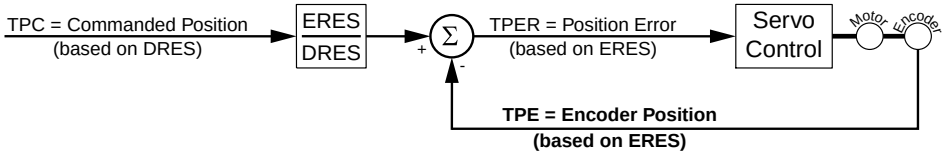
TPC — Transfer Position Commanded

		Product	Rev
Type	Transfer		
Syntax	<a_><!>TPC	GT	1.02
Units	Reported value represents distance units (counts)	GV	1.00
Range	n/a	GT6	1.50
Default	n/a	GV6	1.50
Response	TPC: *TPC+0		
See Also	DRES, ERES, IF, SMPER, TAS, TPE, TPER, VARI, WAIT		

This command allows you to display the *commanded position*. TPC is not applicable while the drive is operating in the torque/force modes (DMODE2 and DMODE15) or velocity modes (DMODE4 and DMODE16).

GT & GT6: The reported value is measured in commanded counts (AKA: "motor counts").

GV: The reported TPC value is measured in drive counts (resolution set with DRES). Commanded position is scaled by the ratio of ERES over DRES as shown below, before the servo control acts upon it.



Position error: If DRES = ERES, then TPER = TPC – TPE; otherwise, $\text{TPER} = \text{TPC} * \left(\frac{\text{ERES}}{\text{DRES}}\right) - \text{TPE}$.

GV6: The reported TPC value is measured in feedback device counts (resolution set with ERES). The position error calculation is always: TPER = TPC – TPE.

Example (assuming DRES = ERES):

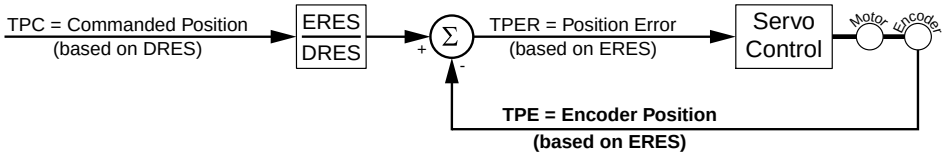
```
TPC      ; Display the commanded position. Example is: *TPC4000
TPE      ; Display the actual position.    Example is: *TPE4004
TPER     ; Display position error:         Example is: *TPER-4
```

TPE — Transfer Position of Encoder/Resolver

		Product	Rev
Type	Transfer		
Syntax	<a_><!>TPE	GT	n/a
Units	Encoder or resolver counts	GV	1.00
Range	n/a	GT6	n/a
Default	n/a	GV6	1.50
Response	TPE: *TPE+0		
See Also	ERES, IF, SFB, TPC, TPER, VARI, WAIT		

The TPE command reports the present feedback device position, based on the encoder/resolver resolution (ERES).

GV:



Position error: If DRES = ERES, then TPER = TPC – TPE; otherwise, $\text{TPER} = \text{TPC} * \left(\frac{\text{ERES}}{\text{DRES}}\right) - \text{TPE}$.

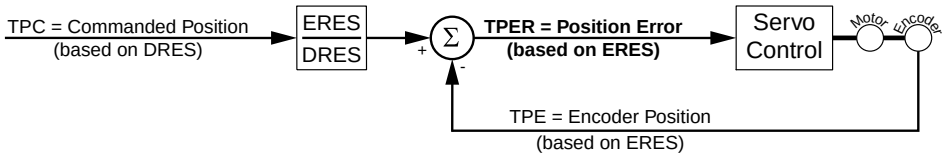
GV6: The position error calculation is always: TPER = TPC – TPE.

TPER — Transfer Position Error

		Product	Rev
Type	Transfers	GT	n/a
Syntax	<a_><!>TPER	GV	1.00
Units	Counts	GT6	n/a
Range	n/a	GV6	1.50
Default	n/a		
Response	TPER: *TPER+0		
See Also	DMODE, DMONAV, DMONBV, DRES, ERES, IF, SFB, SMPER, TAS, TPE, TPC, VARI, WAIT		

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). The position error is calculated every 250 µs.

GV: When the drive is set to DMODE6, 7, or 8, the position error is the difference between the commanded position (scaled by the ratio of ERES/DRES) and the actual position read by the feedback device. TPER does not apply in DMODE2 (torque/force control mode) and in DMODE4 (velocity control mode); TPER reports zero in these modes.



Position error calc.: If DRES = ERES, then TPER = TPC – TPE; otherwise, $\text{TPER} = \text{TPC} * \left(\frac{\text{ERES}}{\text{DRES}}\right) - \text{TPE}$.

GV6: The position error calculation is always: TPER = TPC – TPE.

Example (assuming DRES = ERES):

```
TPC     ; Display the commanded position. Example is: *TPC4000
TPE     ; Display the actual position.    Example is: *TPE4004
TPER    ; Display position error:         Example is: *TPER-4
```

TPRA — Transfer Absolute Resolver Position

		Product	Rev
Type	Transfer	GT	n/a
Syntax	<a_><!>TPRA	GV	1.61
Units	counts	GT6	n/a
Range	0 – (ERES-1)	GV6	1.61
Default	n/a		
Response	TPRA: *TPRA145		
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

		Product	Rev
Type	Transfer	GT	n/a
Syntax	<a_><!>TPROG PROG<i>	GV	n/a
Units	i = program ID number	GT6	1.50
Range	1-32	GV6	1.50
Default	n/a		
Response	(contents of specified program are displayed)		
See Also	DEF PROG, TDIR, TMEM		

The TPROG PROG command displays the contents of the program specified. For example, to display to contents of program #3, issue the TPROG PROG3 command. If there is no such program, the drive responds with the ERRBAD prompt (default is "?"). To check the memory usage for the existing program, use the TMEM command. To report the programs and profiles stored in EEPROM memory, use the TDIR command.

NOTE: TPROG cannot be used to display the contents of defined profiles (DEF PROF).

TRACE Program Trace Mode

Type	Program Debug Tool
Syntax	<a_><!>TRACE<b>
Units	n/a
Range	b = 0 (disable), 1 (enable)
Default	0
Response	TRACE: *TRACE0
See Also	TSS

Product	Rev
GT	n/a
GV	n/a
GT6	1.50
GV6	1.50

The TRACE command enables/disables Program Trace mode. When in program trace mode, all commands executed are transferred out the serial connection (RS-232 or RS-485). TSS bit #8 is set when the trace mode is enabled.

Example:

```
DEL PROG22     ; Delete program #22
DEF PROG22     ; Begin definition of program #22
L              ; Begin loop
 IF(IN.3=b1)   ; If input #3 is activated ...
  JUMP PROG6   ; Jump to program #6
  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 PROG6      ; Delete program #6
DEF PROG6      ; Begin definition of program #6
K1             ; Kill motion only
TERRLG         ; Display the error log
END            ; End definition of program

TRACE1         ; Enable trace mode
RUN PROG22     ; Run program #22
```

After enabling the trace mode, executing RUN PROG22 places the following information in the output buffer: (assume input #3 is not activated)

```
*L
*IF(IN.3=b1)
*ELSE
*MC0
*MA0
*A45
```

(continues)

TREV — Transfer Revision Level

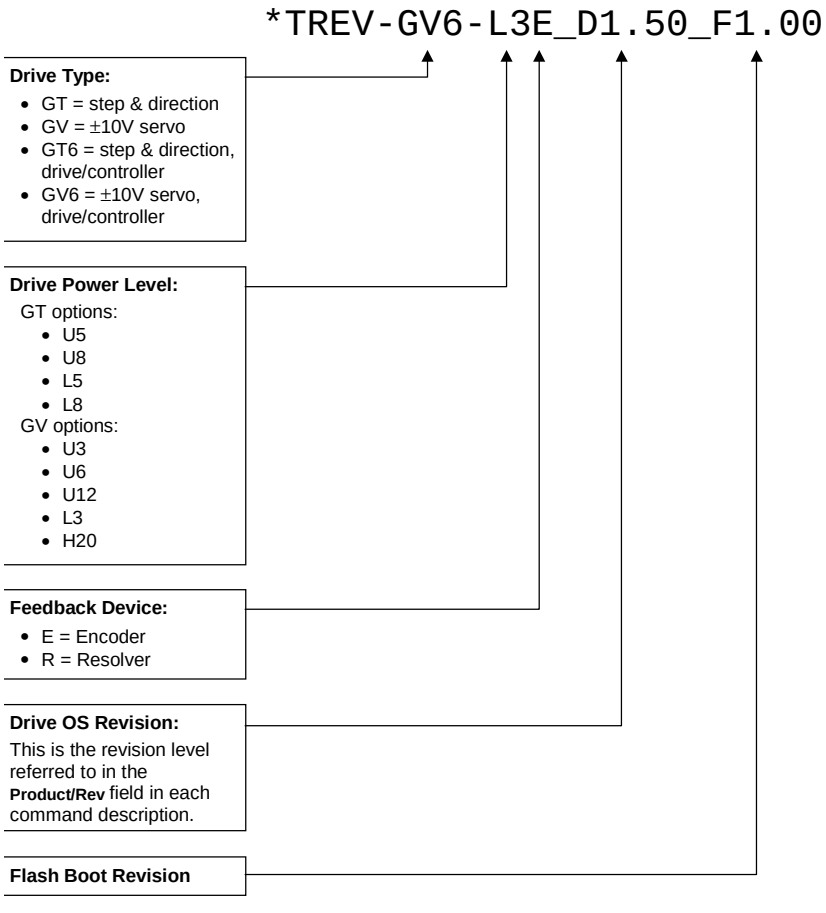
Type	Transfer
Syntax	<a_><!>TREV
Units	n/a
Range	n/a
Default	n/a
Response	TREV: *TREV-GV6-L3E_D1.50_F1.00 (response varies by product)
See Also	RESET, RFS

Product	Rev
GT	1.02
GV	1.00
GT6	1.50
GV6	1.50

The Transfer Revision Level (TREV) command reports the following information:

- Drive type
- Drive power level
- Feedback device (for GV and GV6 only)
- Drive operating system
- Flash boot revision (for hardware identification only)

Example Response:



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). The file name is in this format: GEM_*nnn*.ops (example: the operating system file for version 1.02 is called GEM_1_02.ops). Download the file to your hard drive and follow the relevant download procedure:

- Motion Planner users: refer to page 8.
- Pocket Motion Planner users: refer to page 13.
- Communications Server (COM6SRVR.EXE) users: use the SendOS method described on page 198.

TRGFN Trigger Interrupt Functions

		Product	Rev
Type	Inputs; Compiled Motion	GT	n/a
Syntax	<a_><!>TRGFN<c><b>	GV	n/a
Units	c = letter corresponding to one of the 8 inputs; b = bit to select Conditional GOBUF function	GT6	1.50
Range	c = A-H, corresponding to inputs 1-8, respectively; b = 0 (disable function) or 1 (enable function)	GV6	1.50
Default	c = D (default function for input 4 is "trigger interrupt"); b = 0		
Response	n/a		
See Also	DEF PROF, GOBUF, INFNC, TAS, TIN, TRGLOT		

Use the TRGFN command to assign the "Conditional GOBUF" function to a specific input. The Conditional GOBUF function suspends execution of the next GOBUF command (motion continues at constant velocity) until the specified trigger input goes active. The trigger function is cleared once the GOBUF is executed. To use the trigger function again, the TRGFNc1 command must be given again. If you need execution to be triggered after a time delay (dwell) use the GOWHEN command.

Axis status (TAS) bit #26 is set to one (1) when there is a pending "Conditional GOBUF" 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.

NOTE

The input used in this command must first be defined as a *Trigger Interrupt* input with the INFNCi-H command (where "i" is the input number).

TRGFN(c)1

Input #	Pin #	"c" Letter	Factory Default Input Function
1	28	A	INFNC1-R (positive direction end-of-travel limit)
2	29	B	INFNC2-S (negative direction end-of-travel limit)
3	31	C	INFNC3-T (home limit)
4	34	D	INFNC4-H **(trigger interrupt)**
5	35	E	INFNC5-A (general-purpose input)
6	37	F	INFNC6-A (general-purpose input)
7	38	G	INFNC7-A (general-purpose input)
8	39	H	INFNC8-A (general-purpose input)

Example:

```
DEL PROF1      ; Delete profile #1
DEF PROF1      ; Define profile #1
INFNC4-H       ; Define input #4 as a trigger interrupt input
TRGFND1        ; When input #4 (represented by the letter "D") is activated,
               ; execute the move commanded with the subsequent GOBUF command.
GOBUF1         ; The move is commanded, but will not execute until
               ; input #4 goes active.
END            ; End definition of profile #1
```

TRGLOT Trigger Interrupt Lockout Time

		Product	Rev
Type	Inputs	GT	n/a
Syntax	<a_><!>TRGLOT<i>	GV	n/a
Units	i = time in milliseconds	GT6	1.50
Range	0-250	GV6	1.50
Default	24		
Response	TRGLOT: *TRGLOT24		
See Also	INDEB, INFNC, RE, REG, TIN, TRGFN		

The TRGLOT command configures the amount of time in which all "trigger interrupt" inputs (all inputs configured with the INFNCi-H command) are 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 TRGFN event. The lockout time affects only those triggers configured as H (trigger interrupt) with the INFNC command during those interrupt actions (registration, TRGFN).

The TRGLOT setting overrides the existing INDEB setting for only the trigger inputs that are assigned the "Trigger Interrupt" function.

Example:

```
INFNC1-H      ; Assign input #1 as a "trigger interrupt" input
TRGLOT40      ; Set lockout time for all "trigger interrupt" inputs
              ; to be 40 milliseconds
```

TSGSET Transfer Gain Set

		Product	Rev
Type	Transfer	GT	n/a
Syntax	<a_><!>TSGSET<i>	GV	n/a
Units	i = gain set identification number (see SGSET command)	GT6	n/a
Range	1-3	GV6	1.50
Default	n/a		
Response	(reports all gains in the specified set)		
See Also	SGENB, SGSET, TGAIN, (see list below)		

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)
- DVBW (velocity 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.

TSROFF Transfer Resolver Offset Angle

Type	Transfer
Syntax	<a_><!>TSROFF
Units	angle offset, in degrees
Range	-180.0 to +180.0
Default	n/a
Response	TSROFF: *TSROFF-90.0
See Also	DMODE, SRSET

Product	Rev
GT	n/a
GV	1.60
GT6	n/a
GV6	1.60

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.

WARNING: Motion (less than 1 rev) will occur when you initiate the auto alignment mode.

TSS Transfer System Status

Type	Transfer
Syntax	<a_><!>TSS
Units	n/a
Range	n/a
Default	n/a
Response	TSS: *TSS1000_1100_0000_0000_0000_0000_0000_0000
See Also	TAS

Product	Rev
GT	1.02
GV	1.00
GT6	1.50
GV6	1.50

The TSS command provides information on the 32 system status bits. TSS reports a binary bit report.

*TSSbbbb_bbbb_bbbb_bbbb_bbbb_bbbb_bbbb_bbbb

Bit 1 Bit 32

BIT (Left to Right)	**Function** (1 = yes, Ø = no)	GT/GV	GT6/GV6
1	System Ready (powered up and ready to receive commands).	**X**	**X**
2	RESERVED	--	--
3	Executing a Program	--	**X**
4	RESERVED	--	--
5	RESERVED, always = 1	--	--
6	In Echo Mode (ECHO1).	**X**	**X**
7	Defining a program (DEF PROG)	--	**X**
8	Trace Mode is enabled (TRACE1)	--	**X**
9-12	RESERVED	--	--
13	Pause is active (PS command or pause input, INFNCi-E) or !S is active when COMEXS1 is set	--	**X**
14	Wait is in progress (WAIT)	--	**X**
15	RESERVED	--	--
16	RESERVED	--	--
17	RESERVED	--	--
18	External Program Select Mode is enabled (INSELP1)	--	**X**
19	Dwell is in progress (T)	--	**X**
20	RESERVED	--	--
21	RESERVED	--	--
22	EEPROM Memory Error. To clear this bit, issue a RESET or cycle power.	**X**	**X**
23	RESERVED	--	--
24	RESERVED	--	--
25-32	RESERVED	--	--

TSSF — Transfer System Status (full-text report)

Type	Transfer
Syntax	<a_><!>TSSF
Units	n/a
Range	n/a
Default	n/a
Response	TSSF: (see example below)
See Also	TAS, TASF, TASX, TASXF, TER, TERF, TSS

Product	Rev
GT	1.70
GV	1.70
GT6	1.70
GV6	1.70

The TSSF command returns a text-based status report. This is an alternative to the binary report (TSS).

Example TSSF response:

```
*TSSF
Bit  1: System Ready             YES
Bit  2: Reserved                 NO
Bit  3: Program Executing        NO
Bit  4: Reserved                 NO
Bit  5: ASCII Mode               YES
Bit  6: Echo Mode                YES
Bit  7: Defining a Program       NO
Bit  8: Trace Mode               NO
Bit  9: Reserved                 NO
Bit 10: Reserved                 NO
Bit 11: Reserved                 NO
Bit 12: Reserved                 NO
Bit 13: Pause Active             NO
Bit 14: Wait Active              NO
Bit 15: Reserved                 NO
Bit 16: Reserved                 NO
Bit 17: Reserved                 NO
Bit 18: External Program Select  NO
Bit 19: Dwell in Progress        NO
Bit 20: Reserved                 NO
Bit 21: Reserved                 NO
Bit 22: Memory Error             NO
Bit 23: Servo Data Transfer      NO
Bit 24: Reserved                 NO
Bit 25: Reserved                 NO
Bit 26: Reserved                 NO
Bit 27: Reserved                 NO
Bit 28: Reserved                 NO
Bit 29: Reserved                 NO
Bit 30: Reserved                 NO
Bit 31: Reserved                 NO
Bit 32: Reserved                 NO
```

TSTLT — Transfer Settling Time

Type	Transfer
Syntax	<a_><!>TSTLT
Units	Reported value represents milliseconds
Range	n/a
Default	n/a
Response	TSTLT: *TSTLT502
See Also	STRGTD, STRGTE, STRGTT, STRGTV

Product	Rev
GT	n/a
GV	n/a
GT6	n/a
GV6	1.50

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 STRGTE1 command.**

*** For more information on target zone operation, refer to page 37.

TTRQ — Transfer Commanded Torque/Force

Type	Transfer
Syntax	<a_><!>TTRQ
Units	% of the DMTSCL value
Range	-100.00% to +100.00% : ±0.01
Default	n/a
Response	TTRQ: *TTRQ50.00
See Also	DMODE, DMONAV, DMONBV, DMTLIM, DMTSCL, TTRQ

Product	Rev
GT	n/a
GV	1.00
GT6	n/a
GV6	1.50

In Torque/Force mode (DMODE2), TTRQ reports back the commanded ±10V value from the user before any internal limits are checked. 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
Syntax	<a_><!>TTRQA
Units	% of full-scale torque/force set by DMTSCL command
Range	-100.00% to +100.00% : ±0.01
Default	n/a
Response	TTRQA: *TTRQA55
See Also	DMONAV, DMONBV, DMTKE, DMTLIM, DMTSCL, TTRQ

Product	Rev
GT	n/a
GV	1.00
GT6	n/a
GV6	1.50

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.

TVE — Transfer Velocity Error

Type	Transfer
Syntax	<a_><!>TVE
Units	Revs/sec (linear motors: see DMEPIT for linear/rotary conversion)
Range	-200 to 200
Default	n/a
Response	TVE: *TVE3
See Also	DMEPIT, DMONAV, DMONBV, DMVLIM, DMVSCL, SMVER, TVEL, TVELA, V

Product	Rev
GT	n/a
GV	1.00
GT6	n/a
GV6	1.50

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). TVE is not applicable while the drive is operating in the torque/force mode (DMODE2) – TVE reports zero in this mode.

If operating in velocity mode (DMODE4), be sure your maximum allowable velocity is set higher than your setpoint. If this is not the case, the velocity error (TVE) will be internally limited to DMVLIM - TVELA.

The maximum allowable velocity error limit is established with the SMVER command.

TVEL — Transfer Commanded Velocity

		Product	Rev
Type	Transfer	GT	1.02
Syntax	<a_><!>TVEL	GV	1.00
Units	Revs/sec (linear motors: see DMEPIT for linear/rotary conversion)	GT6	1.50
Range	-200.000000 to 200.000000 : ±0.000001	GV6	1.50
Default	n/a		
Response	TVEL: *TVEL23.345000		
See Also	DMEPIT, DMONAV, DMONBV, DMVLIM, ERES, TVELA, V		

The TVEL command reports the commanded velocity. TVEL is not applicable while the drive is operating in the torque/force modes (DMODE2 and DMODE15). It is not the programmed velocity (V).

In velocity mode (DMODE4), TVEL reports back the commanded ±10V value from the user before any internal limits are checked. In velocity tuning mode (DMODE16), the commanded velocity is fixed at ±2 revs/sec (for rotary motors).

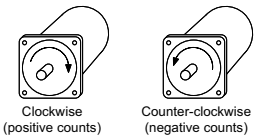
In position mode (DMODE6-9,12,17), TVEL reports the internal velocity command and is limited by DMVLIM.

TVELA — Transfer Actual Velocity

		Product	Rev
Type	Transfer	GT	n/a
Syntax	<a_><!>TVELA	GV	1.00
Units	Revs/sec (linear motors: see DMEPIT for linear/rotary conversion)	GT6	n/a
Range	-200.000000 to 200.000000 : ±0.000001	GV6	1.50
Default	n/a		
Response	TVELA: *TVELA+1.55		
See Also	DMEPIT, DMONAV, DMONBV, DMVLIM, DMVSCL, TPE, TVE, TVEL, V		

The TVELA command reports the velocity as derived from the feedback device. The sign determines the direction of motion (+ for positive-counting direction, – for negative-counting direction).

Rotary Motors:
Positive values represent clockwise motion and negative values represent counter-clockwise motion (assuming you connected the feedback device per the *Hardware Installation Guide* instructions).



You can use the TVELA command at all times; therefore, even if no motion is being commanded, TVELA could still report a non-zero value as it detects the servoing action.

V — Velocity

		Product	Rev
Type	Motion	GT	n/a
Syntax	<a_><!>V<r>	GV	n/a
Units	revs/sec (linear motors: see page 44 for linear/rotary conversion)	GT6	1.50
Range	0.0000 to 200.0000 : ±0.0001	GV6	1.50
Default	1.0000		
Response	V: *V1.0000		
See Also	DMEPIT, GO, IF, MC, TVEL, TVELA, VF, VARI, WAIT		

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 (A) rate, before reaching the velocity (V) specified.

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. For details, see page 24.

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:

```
DEL PROG2     ; Delete program #2
DEF PROG2     ; Begin definition of program #2
MA0           ; Incremental positioning mode
MC0           ; Preset positioning mode
A10           ; Set the acceleration to 10 revs/sec/sec
V1            ; Set the velocity to 1 rev/sec
D100000       ; Set the distance to 100,000 counts
GO1           ; Initiate motion
END           ; End definition of program
```

VARCLR Variable Clear

		Product	Rev
Type	Variables	GT	n/a
Syntax	<a_><!>VARCLR	GV	n/a
Units	n/a	GT6	1.70
Range	n/a	GV6	1.70
Default	n/a		
Response	n/a		
See Also	VARI		

VARCLR resets all variables (VARI) to the default value of zero.

VARI Variable (Integer)

		Product	Rev
Type	Variables	GT	n/a
Syntax	<a_><!>VARI<i><=i>	GV	n/a
Units	1st i = variable number. 2nd i = integer value (assignment).	GT6	1.60
Range	1st i = 1-99. 2nd i = -2,147,483,648 to +2,147,483,647.	GV6	1.60
Default	0 (not stored in EEPROM		
Response	VARI1: *+32		
See Also	A, AD, D, IF, L, T, TPC, TPE, TPER, V, VARCLR, WAIT		

The GT6 and GV6 drives allow you to define up to 99 user variables (integer variables) . Integer variables are represented by the syntax VARI*n*, where "*n*" is the number of the variable (range is 1-99). All VARI variables are set to zero at reset, and are not stored in EEPROM.

Integer variables may be used for:

- **Variable assignments and math operations.**

VARI*n* = <assignment>

Number of the integer variable. Range is 1-99.

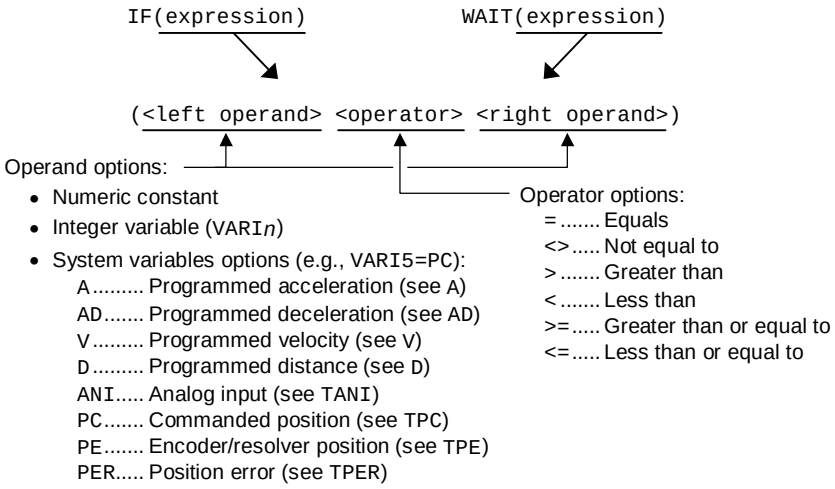
"=" is required.

Assignment options are:

- Integer constant, range is -2,147,483,648 to +2,147,483,647 (e.g., VARI8=150).
- System variables (e.g., VARI5=PC):
  - A ..... Programmed acceleration (see A)
  - AD ..... Programmed deceleration (see AD)
  - V ..... Programmed velocity (see V)
  - D ..... Programmed distance (see D)
  - PC ..... Commanded position (see TPC)
  - PE ..... Encoder/resolver position (see TPE)
  - PER ..... Position error (see TPER)
- Math operation between integers, other VARI variables, and system variables (listed above). Available math operations are:
  - + ... Add (e.g., VARI2=VARI2+1)
  - - ..... Subtract (e.g., VARI6=PE-PER)
  - * ..... Multiply (e.g., VARI4=A*VARI7)
  - / ..... Divide (e.g., VARI3=PC/2)

- **Command value substitutions.** Variable substitution allows you to substitute the value of a VARI variable as the parameter value for certain commands (applicable to T, L, D, A, AD, REG, PSET and V only). For example, if VARI9 is substituted for the L loop parameter L(VARI9), and the value of VARI9 is 7 when the L command is executed, the Gemini will initiate a 7-iteration loop. For more examples, see page 24.

- **Variable comparisons in conditional expressions.** Variable comparison allows program flow to be affected by IF and WAIT conditional expressions, based on comparisons between integer variables and system variables or integer values. Relative to variable comparisons, the options for the comparison operands (left and right) and the comparison operator are:



Examples:

IF(VARI5<PE) ..... Compares the value of integer variable #5 (VARI5) with the present encoder position (PE). The condition evaluates true when the value of VAR5 is less than the integer value of the encoder position.

WAIT(PE>=16000) ..... Command execution pauses until the encoder or resolver position is greater than or equal to 16000.

NOTE

A, AD, V and T command values are real numbers (resolution of A, AD, and V is 0.0001, resolution of T is 0.001). When substituting or comparing integer variables, the integer is applied to the full decimal range (for example, if the value of VARI5 is 136298, the substitution A(VARI5) yields an acceleration value of A13.6298). The converse is true when assigning the value of A, AD, or V to an integer variable (for example, if the value of A is 22.0000, the integer assignment VARI4=A yields a VARI4 value of 220000).

Programming Example: See page 25

VF — Final Velocity

		Product	Rev
Type	Compiled Motion	GT	n/a
Syntax	<a_><!>VF<r>	GV	n/a
Units	n/a	GT6	1.50
Range	0 (non-zero values result in error message)	GV6	1.50
Default	0		
Response	n/a		
See Also	GOBUF, V, (Compiled Motion overview on page 49)		

The Final Velocity (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.

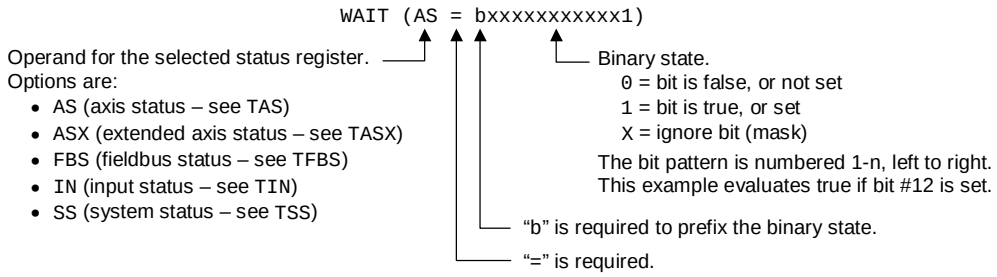
WAIT — Wait for a Specific Condition

		Product	Rev
Type	Program Flow Control	GT	n/a
Syntax	<a_><!>WAIT(*expression*)	GV	n/a
Units	(see syntax examples below)	GT6	1.50
Range	(see syntax examples below)	GV6	1.50
Default	n/a		
Response	n/a		
See Also	A, AD, D, GOWHEN, IF, T, TAS, TASX, TIN, TPC, TPE, TPER, TSS, V, VARI		

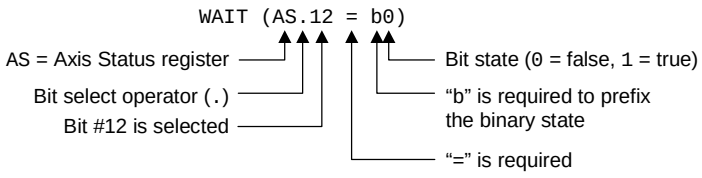
Use the WAIT command to wait for a specific expression to evaluate true. No commands, except for immediate commands, after the WAIT command will be processed until the expression contained within the parentheses of the WAIT command evaluates true. The COMEXC command has no effect on the WAIT command. When a wait condition is pending, system status (TSS) bit #14 is set.

Syntax Example: Binary Data

In this example, the WAIT condition evaluates true when bit #12 of the Axis Status register is set (binary value is "1").



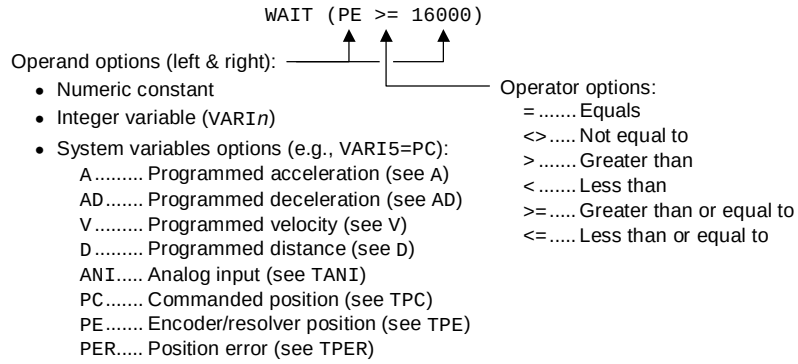
Syntax Example, using the binary bit-select shortcut:



Syntax Example:
Integer/Variable Data Comparison

(This capability is available as of OS rev 1.60)

In this example, the `IF` condition compares the present encoder/resolver position (PE) with a numeric constant (16000). The condition evaluates true when the integer value of the encoder position is greater than or equal to 16000.



NOTE

System variables A, AD, and V, are real numbers, with a resolution of 0.0001. When comparing the value of these variables, the `WAIT` condition uses the integer representation (removes the decimal point). For example, if the commanded velocity is 5.0000 units/sec, the integer observed by a `WAIT` evaluation would be 50000. Thus, if you want to wait until the commanded velocity (V) is <= 5.0000 units/sec, use this syntax: `WAIT(V<=50000)`.

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(AS.1=b0) ; Wait for no commanded motion
COMEXC0        ; Disable continuous command execution mode
```

XONOFF Enable/Disable XON / XOFF

Type	Communication Interface
Syntax	<a_><!>XONOFF<b>
Units	b = enable bit
Range	0 (disable) or 1 (enable)
Default	1
Response	XONOFF: *XONOFF1
See Also	BOT, E, EOT, ERRBAD, ERROK

Product	Rev
GT	1.02
GV	1.01
GT6	1.50
GV6	1.50

Use the `XONOFF` command to enable or disable XON/XOFF (ASCII handshaking). `XONOFF1` enables XON/XOFF, which allows the Gemini product to recognize ASCII handshaking control characters. `XONOFFØ` disables XON/XOFF.

NOTE: If you are using RS-485 multi-drop or RS-232 daisy-chain, disable XON/XOFF with `XONOFFØ`.

Appendix A:
Command Quick Reference
(commands listed alphabetically)

Command	Description	GT	GV	GT6	GV6	Type/Group	Syntax	Units	Range	Default
A	Acceleration	--	--	X	X	Motion	<a_><!>A<r>	revs/sec/sec	0.0001 - 9999.9999	10.0000
AA	Acceleration (S-Curve)	--	--	X	X	Motion (S-curve)	<a_><!>AA<r>	revs/sec/sec	0.0001 - 9999.9999	10.0000
AD	Deceleration	--	--	X	X	Motion	<a_><!>AD<r>	revs/sec/sec	0.0001 - 9999.9999	10.0000
ADA	Deceleration (S-Curve)	--	--	X	X	Motion (S-curve)	<a_><!>ADA<r>	revs/sec/sec	0.0001 - 9999.9999	10.0000
ADDR	☒ Multiple Unit Auto-Address	X	X	X	X	Comm. I/F	<a_><!>ADDR<i>	unit address	0 - 99	0
ANICDB	Command Input Deadband	X	X	--	--	Drive Config.	<a_><!>ANICDB<r>	Volts	0.00 – 10.00	0.04
BOT	Beginning Transmission Characters	X	X	X	X	Comm. I/F	<!>BOT<i>,<i>,<i>	ASCII equivalent	0 - 255	0,0,0
C	Continue Command Exec.	--	--	X	X	Prog. Flow	<a_>!C	N/A	N/A	N/A
CERRLG	Clear Error Log	X	X	X	X	Error Handlg	<a_><!>CERRLG	N/A	N/A	N/A
COMEXC	☒ Continuous Command Processing Mode	--	--	X	X	Command Buffer Contrl	<a_><!>COMEXC<b>	enable bit	0 (disable), 1 (enable)	0
COMEXL	☒ Continue Execution on Limit	--	--	X	X	Command Buffer Contrl	<a_><!>COMEXL<b>	enable bit	0 (disable), 1 (enable)	0
COMEXR	☒ Continue Motion on Pause/Continue Input	--	--	X	X	Command Buffer Contrl	<a_><!>COMEXR<b>	enable bit	0 (disable), 1 (enable)	0
COMEXS	☒ Continue Execution on Stop	--	--	X	X	Command Buffer Contrl	<a_><!>COMEXS<i>	function ID	0, 1, or 2	0
D	Distance	--	--	X	X	Motion	<a_><!>D<r>	distance (counts)	-2^{31} to $+2^{31}-1$	4000
DABSD	☒ Enable ABS Damping	X	--	X	--	Drive Config.	<a_><!>DABSD<b>	enable bit	0 (disable), 1 (enable)	0
DACTDP	☒ Active Damping Gain	X	--	X	--	Drive Config.	<a_><!>DACTDP<i>	gain value	0 - 40	4
DAUTOS	☒ Auto Standby Reduction	X	--	X	--	Drive Config.	<a_><!>DAUTOS<r>	percentage	0.00 - 99.99	0
DCLRLR	Clear Latched Status Register	X	X	X	X	Drive Config.	<a_><!>DCLRLR	N/A	N/A	N/A
DCMDZ	☒ Zero The Drive Command Offset	X	X	--	--	Drive Config.	<a_><!>DCMDZ	N/A	N/A	N/A
DDAMPA	☒ Damping During Accel	X	--	X	--	Drive Config.	<a_><!>DDAMPA<b>	enable bit	0 (disable), 1 (enable)	0
DEFPROF	☒ Define a Profile	--	--	X	X	Compiled Motion	<a_><!>DEF PROF<i>	profile ID #	1 - 16	N/A
DEFPROG	☒ Define a Program	--	--	X	X	Program Def.	<a_><!>DEF PROG<i>	program ID #	1 - 32	N/A
DELPROF	Delete Profile	--	--	X	X	Compiled Motion	<a_><!>DEL PROF<i>	profile ID #	1 - 16	N/A
DELPROG	Delete Program	--	--	X	X	Program Def.	<a_><!>DEL PROG<i>	program ID #	1 - 32	N/A
DELVIS	☒ Electronic Viscosity Enable	X	--	X	--	Drive Config.	<a_><!>DELVIS<i>	gain	0 - 7 (0 = disabled)	0
DIBW	☒ Current Loop Bandwidth	--	X	--	X	Tuning	<a_><!>DIBW<i>	Hz	0 - 5000	0 (auto)
DIFOLD	☒ Current Foldback Enable	--	X	--	X	Drive Config.	<a_><!>DIFOLD<b>	enable bit	0 (disable), 1 (enable)	0
DIGN	☒ Current Loop Gain	X	--	X	--	Tuning	<a_><!>DIGNc<r>	c: gain identifier r: value of gain	DIGNA-C: 0.000 - 15.000 DIGND: 0.000 - 1.000	0 (auto)
DMEPIT	♦☒ Motor Electrical Pitch	--	X	--	X	Motor	<a_><!>DMEPIT<r>	millimeters	0.00 – 327.68	0 (auto)
DMODE	☒ Drive Control Mode	X	X	X	X	Drive Config.	<a_><!>DMODE<i>	control mode	1 - 17	GV: 2, GT: 6, GT6/GV6: 12
DMONAS	☒ Analog Monitor Output A Scaling	X	X	X	X	Outputs	<a_><!>DMONAS<i>	%	-2000 - 2000	100
DMONAV	☒ Analog Monitor Output A Variable	X	X	X	X	Outputs	<a_><!>DMONAV<i>	variable number	0 - 24	0
DMONBS	☒ Analog Monitor Output B Scaling	X	X	X	X	Outputs	<a_><!>DMONBS<i>	%	-2000 - 2000	100
DMONBV	☒ Analog Monitor Output B Variable	X	X	X	X	Outputs	<a_><!>DMONBV<i>	variable number	0 - 24	0
DMTAMB	☒ Motor Ambient Temp.	--	X	--	X	Motor	<a_><!>DMTAMB<r>	Degrees Celsius	-50.0 - 250.0	40.0
DMTD	♦☒ Motor Damping	--	X	--	X	Motor	<a_><!>DMTD<r>	R: Nm/rad/sec L: N/meter/sec	R: 0.000000 - 0.010000 L: DMEPIT dependent	0 (auto)
DMTIC	♦(GV) ☒ Continuous Current	X	X	X	X	Motor	<a_><!>DMTIC<r>	Amps-RMS	0.00 - 100.00	0 (auto)
DMTICD	♦☒ Continuous Current Derating	--	X	--	X	Motor	<a_><!>DMTICD<r>	%	0.00 - 100.00	0 (auto)
DMTIND	♦☒ Motor Inductance	X	--	X	--	Motor	<a_><!>DMTIND<r>	mH	0.0 - 200.0	0 (auto)
DMTIP	☒ Peak Current	--	X	--	X	Motor	<a_><!>DMTIP<r>	Amps-RMS	0.00 - 128.00	0 (auto)
DMTJ	♦☒ Motor Rotor Inertia or Motor Mass	X	X	X	X	Motor	<a_><!>DMTJ<r>	R: $kgm^2 * 10^{-6}$ L: kg	R: 0.000 - 10000.000 L: DMEPIT dependent	0 (auto)

♦ = requires reset (RESET, cycle power, or reset input) to apply.
☒ = saved in EEPROM (Gn6: only if executed outside a program).
(auto) = Auto-configured based on motor selection, see DMTR.

Command	Description	GT	GV	GT6	GV6	Type/Group	Syntax	Units	Range	Default
DMTKE	♦⊠ Motor Ke	--	X	--	X	Motor	<a_><!>DMTKE<r>	R: V(0-peak)/krpm L: volts/meter/sec	R: 0.0 - 200.0 L: DMEPIT dependent	0 (auto)
DMTLIM	⊠ Torque/Force Limit	--	X	--	X	System	<a_><!>DMTLIM<r>	R: Nm L: N	R: 0.0 - 500.0 L: DMEPIT dependent	500 (auto)
DMTLMN	♦⊠ Minimum Motor Inductance	--	X	--	X	Motor	<a_><!>DMTLMN<r>	mH	0.1 - 200.0	0 (auto)
DMTLMX	♦⊠ Maximum Motor Inductance	--	X	--	X	Motor	<a_><!>DMTLMX<r>	mH	0.1 - 200.0	0 (auto)
DMTMAX	♦⊠ Maximum Motor Winding Temperature	--	X	--	X	Motor	<a_><!>DMTMAX<r>	Decrees Celsius	-50.0 - 250.0	125°C (auto)
DMTR	⊠ Identify Motor	X	X	X	X	Drive Config.	<a_><!>DMTR<i>	motor number	0 - 2000	-1
DMTRES	♦⊠ Motor Winding Resistance	X	X	X	X	Motor	<a_><!>DMTRES<r>	Ohm	0.00 - 50.00	0 (auto)
DMTRWC	♦⊠ Motor Winding Thermal Resistance	--	X	--	X	Motor	<a_><!>DMTRWC<r>	°C/Watt	0.00 - 50.00	0 (auto)
DMTSCL	⊠ Torque/Force Scaling	--	X	--	X	Drive Config.	<a_><!>DMTSCL<r>	R: Nm L: N	R: 0.0 - 500.0 L: DMEPIT dependent	0 (auto)
DMTSTT	♦⊠ Motor Static Torque	X	--	X	--	Motor	<a_><!>DMTSTT<r>	oz-in	0.0 - 5000.0	0 (auto)
DMTTCM	♦⊠ Motor Thermal Time Constant	--	X	--	X	Motor	<a_><!>DMTTCM<r>	minutes	0.0 - 300.0	0 (auto)
DMTTCW	♦⊠ Motor Winding Time Constant	--	X	--	X	Motor	<a_><!>DMTTCW<r>	minutes	0.00 - 100.00	0 (auto)
DMTW	♦⊠ Motor Rated Speed	--	X	--	X	Motor	<a_><!>DMTW<r>	R: revs/sec L: meters/sec	R: 0.0 - 200.0 L: DMEPIT dependent	0 (auto)
DMVLIM	⊠ Velocity Limit	X	X	X	X	System	<a_><!>DMVLIM<r>	R: revs/sec L: meters/sec	R (GT): 0.000000-60.000000 (GV): 0.000000-200.000000 L: DMEPIT dependent	GT: 50 (auto) GV: 200 (auto)
DMVSCL	⊠ Velocity Scaling	X	X	--	--	Drive Config.	<a_><!>DMVSCL<r>	R: revs/sec L: meters/sec	R (GT): 0.000000-60.000000 (GV): 0.000000-200.000000 L: DMEPIT dependent	GT: 50 (auto) GV: 0 (auto)
DNOTAD	⊠ Notch Filter A Depth	--	X	--	X	Tuning	<a_><!>DNOTAD<r>	N/A	0.0000 - 1.0000	0
DNOTAF	⊠ Notch Filter A Frequency	--	X	--	X	Tuning	<a_><!>DNOTAF<i>	Hz	0, 60 - 1000 (0=disable)	0
DNOTAQ	⊠ Notch Filter A Quality Factor	--	X	--	X	Tuning	<a_><!>DNOTAQ<r>	quality factor	0.5 - 2.5	1
DNOTBD	⊠ Notch Filter **B** Depth	--	X	--	X	Tuning	<a_><!>DNOT**B**D<r>	N/A	0.0000 - 1.0000	0
DNOTBF	⊠ Notch Filter B Frequency	--	X	--	X	Tuning	<a_><!>DNOTBF<i>	Hz	0, 60 - 1000 (0=disable)	0
DNOTBQ	⊠ Notch Filter A Quality Factor	--	X	--	X	Tuning	<a_><!>DNOTBQ<r>	quality factor	0.5 - 2.5	1
DNOTLD	⊠ Notch Lead Filter Break Frequency	--	X	--	X	Tuning	<a_><!>DNOTLD<i>	Hz	0, 80 - 1000 (0=disable)	0
DNOTLG	⊠ Notch Lag Filter Break Frequency	--	X	--	X	Tuning	<a_><!>DNOTLG<i>	Hz	0, 20 - 1000 (0=disable)	0
DPBW	⊠ Position Loop Bandwidth	--	X	--	X	Tuning	<a_><!>DPBW<r>	Hz	1.00 - 100.00	5 (auto)
DPHBAL	⊠ Phase Balance	X	--	X	--	Drive Config.	<a_><!>DPHBAL<r>	%	90.0 - 110.0	100
DPHOFA	⊠ Phase A Current Offset	X	--	X	--	Drive Config.	<a_><!>DPHOFA<r>	%	-10.000 - 10.000	0
DPHOFB	⊠ Phase B Current Offset	X	--	X	--	Drive Config.	<a_><!>DPHOFB<r>	%	-10.000 - 10.000	0
DPOLE	♦⊠ Motor Pole Pairs	X	X	X	X	Motor	<a_><!>DPOLE<i>	pole pairs	1 - 200	0 (auto)
DPWM	♦⊠ Drive PWM Frequency	--	X	--	X	Drive Config.	<a_><!>DPWM<i>	Hz	0, 8, 16, 20 or 40	8 (L3 is 40)
DRES	♦⊠ Drive Resolution	X	X	X	--	Drive Config.	<a_><!>DRES<i>	R: counts/rev L: counts/epitch	GT: 200 - 128000 GV: 200 - 1024000	GT: 25000 GV: 4000
DRIVE	Drive Enable	X	X	X	X	Drive Config.	<a_><!>DRIVE<b>	enable bit	0 (shutdown), 1 (enable)	1
DSTALL	⊠ Stall Detect Sensitivity	X	--	X	--	Drive Config.	<a_><!>DSTALL<i>	stall sensitivity	0 - 50 (0 = disable)	0
DVBW	⊠ Velocity Loop Bandwidth	--	X	--	X	Tuning	<a_><!>DVBW<i>	Hz	0 - 500	0 (auto)
DWAVEF	⊠ Waveform	X	--	X	--	Drive Config.	<a_><!>DWAVEF<r>	%	-20.00 - 10.00	-4
E	Enable Communication	X	X	X	X	Comm. I/F	<a_><!>E<b>	enable bit	0 (disable), 1 (enable)	1
ECHO	⊠ Communication Echo	X	X	X	X	Comm. I/F	<a_><!>ECHO<b>	enable bit	0 (disable), 1 (enable)	1
ELSE	Else Condition (IF = false)	--	--	X	X	Prog. Flow	<a_><!>ELSE	N/A	N/A	N/A
END	End Program/Profile Def(n)	--	--	X	X	Program Def.	<a_><!>END	N/A	N/A	N/A
EOL	End-of-Line Characters	X	X	X	X	Comm. I/F	<!>EOL<i>,<i>,<i>	ASCII equivalent	0 - 255	13,10,0
EOT	End-of-Transmission Characters	X	X	X	X	Comm. I/F	<!>EOT<i>,<i>,<i>	ASCII equivalent	0 - 255	13,0,0
ERASE	Delete Programs / Profiles	--	--	X	X	Program Def.	<a_><!>ERASE	N/A	N/A	N/A
ERES	♦⊠ Encoder Resolution or Resolver Resolution	--	X	--	X	Drive Config.	<a_><!>ERES<i>	R: counts/rev L: counts/epitch	200 – 1024000	4000 (auto)
ERRBAD	Error Prompt	X	X	X	X	Comm. I/F	<!>ERRBAD<i>,<i>,<i>,<i>	ASCII equivalent	0 - 255	13,10,63,32
ERRDEF	Program Definition Prompt	X	X	X	X	Comm. I/F	<!>ERRDEF<i>,<i>,<i>,<i>	ASCII equivalent	0 - 255	13,10,45,32
ERRLVL	⊠ Error Detection Level	X	X	X	X	Comm. I/F	<a_><!>ERRLVL<i>	error level setting	0, 2, or 3	3
ERROK	Good Prompt	X	X	X	X	Comm. I/F	<!>ERROK<i>,<i>,<i>,<i>	ASCII equivalent	0 - 255	13,10,62,32
ERROR	⊠ Error Checking Enable	--	--	X	X	Error Handlg	<a_><!>ERROR<b>…	enable bit	0 (disable), 1 (enable), X	0
ERRORP	⊠ Error Program Assign.	--	--	X	X	Error Handlg	<a_><!>ERROR PROG<i>	program ID #	0, 1 - 32 (0 = unassign)	0
ESK	⊠ Fault on Stall Enable	X	--	X	--	Drive Config.	<a_><!>ESK<b>	enable bit	0 (disable), 1 (enable)	1
FLTDSB	⊠ Fault on Drive Disable	X	X	X	X	Drive Config.	<a_><!>FLTDSB<b>	enable bit	0 (disable), 1 (enable)	1
FLTSTP	♦⊠ Fault on Startup Indexer Pulses	X	X	--	--	Drive Config.	<a_><!>FLTSTP<b>	enable bit	0 (disable), 1 (enable)	1
GO	Initiate Motion	--	--	X	X	Motion	<a_><!>GO<b>	N/A	0 (don't go), 1 (go)	1
GOBUF	Store Compiled Motion Segment	--	--	X	X	Compiled Motion	<a_><!>GOBUF<b>	N/A	0 (don't go), 1 (go)	1
GOSUB	Call a Subroutine	--	--	X	X	Prog. Flow	<a_><!>GOSUB PROG<i>	program ID #	1 - 32	N/A
GOWHEN	Conditional GOBUF	--	--	X	X	Comp. Motion	<a_><!>GOWHEN(T=i)	i = milliseconds	1 - 999999	N/A
HOM	Go Home	--	--	X	X	Homing	<a_><!>HOM<b>	enable bit	0 (home pos.), 1 (home neg.)	N/A
HOMA	⊠ Home Acceleration	--	--	X	X	Homing	<a_><!>HOMA<r>	revs/sec/sec	0.0001 - 9999.9999	10
HOMBAC	⊠ Home Backup	--	--	X	X	Homing	<a_><!>HOMBAC<b>	enable bit	0 (disable), 1 (enable)	0
HOMDF	⊠ Home Final Direction	--	--	X	X	Homing	<a_><!>HOMDF<b>	N/A	0 (pos. dir.), 1 (neg. dir.)	0
HOMEDG	⊠ Home Reference Edge	--	--	X	X	Homing	<a_><!>HOMEDG<b>	N/A	0 (pos. dir.), 1 (neg. dir.)	0

♦ = requires reset (RESET, cycle power, or reset input) to apply.
⊠ = saved in EEPROM (Gn6: only if executed outside a program).
(auto) = Auto-configured based on motor selection, see DMTR.

Command	Description	GT	GV	GT6	GV6	Type/Group	Syntax	Units	Range	Default
HOMV	☒ Home Velocity	--	--	X	X	Homing	<a_><!>HOMV<r>	revs/sec	0.000 – 200.0000	1
HOMVF	☒ Home Final Velocity	--	--	X	X	Homing	<a_><!>HOMVF<r>	revs/sec	0.000 - 200.0000	0.1000
HOMZ	☒ Home to Encoder Z Channel	--	--	--	X	Homing	<a_><!>HOMZ<b>	enable bit	0 (disable), 1 (enable)	0
IF	IF Conditional Statement	--	--	X	X	Prog. Flow	<a_><!>IF(<op>=b<b>) <a_><!>IF(<op>.<i>=b<b>)	<op> = operand <b> = binary state <i> = selected bit	<op>: AS, ASX, ER, IN, SS <b>: 1 or 0 <i>: 1 - 32 (varies by operand)	N/A
INDEB	☒ Input Debounce Time	X	X	X	X	Inputs	<a_><!>INDEB<i>	milliseconds	2 - 250	50
INFNC	☒ Input Function Assignment	--	--	X	X	Inputs	<a_><!>INFNC<i>-<c>	<i> = input # <c> = function ID	<i>: 1 - 8 <c>: A (general purpose), B (BCD), C (kill), E (pause), F (fault), H (trigger), R (EOT limit, pos), S (EOT limit, neg), or T (home limit).	INFNC1-R INFNC2-S INFNC3-T INFNC4-H
INLVL	☒ Input Level Sense	X	X	X	X	Inputs	<a_><!>INLVL<bb...>	N/A	0 (active low), 1 (active high)	11000000
INSELP	Program Select Mode	--	--	X	X	Inputs	<a_><!>INSELP<b>,<i>	<b>: enable bit <i>: strobe (ms)	<b>: 0 (disable), 1 (enable) <i>: 0 - 5000	<b>: 0 <i>: 0
INUFD	User Fault Input Delay	X	X	X	X	Inputs	<a_><!>INUFD<i>	Milliseconds	0 – 1000	0
JUMP	Jump to a Program	--	--	X	X	Prog. Flow	<a_><!>JUMP PROG<i>	program ID #	1 - 32	N/A
K	Kill Motion	--	--	X	X	Motion	<a_><!>K<b>	enable bit	0 (don't' kill), 1 (kill)	1
KDRIVE	☒ Disable Drive on Kill	--	--	X	X	Drive Config.	<a_><!>KDRIVE<b>	enable bit	0 (disable), 1 (enable)	0
L	Loop	--	--	X	X	Prog. Flow	<a_><!>L<i>	# of times to loop	0 – 999,999,999 (0 = infinite)	0 (infinite)
LDAMP	☒ Load Damping	--	X	--	X	System	<a_><!>LDAMP<r>	R: Nm/rad/sec L: N/meter/sec	R: 0.0000 - 1.0000 L: DMEPIT dependent	0
LH	☒ Hardware End-of-Travel Limit Enable	X	X	X	X	EOT Limit; Drive config.	<a_><!>LH<i>	N/A	0 (disable both limits), 1 (disable pos. dir. limit only), 2 (disable neg. dir. limit only), 3 (enable both limits)	GT/GV: 0 GT6/GV6: 3
LHAD	☒ Hardware EOT Limit Deceleration	--	--	X	X	EOT Limit	<a_><!>LHAD<r>	revs/sec/sec	0.0001 - 9999.9999	100.0000
LHADA	☒ Hardware EOT Limit Deceleration (S-Curve)	--	--	X	X	EOT Limit; Motion (S)	<a_><!>LHADA<r>	revs/sec/sec	0.0001 - 9999.9999	100.0000
LJRAT	☒ Load-to-Rotor Inertia Ratio or Load-to-Forcer Ratio	X	X	X	X	System	<a_><!>LJRAT<r>	ratio	0.0 - 100.0	0
LN	End of Loop	--	--	X	X	Prog. Flow	<a_><!>LN	N/A	N/A	N/A
LS	☒ Software End-of-Travel Limit Enable	--	--	X	X	EOT Limit	<a_><!>LS<i>	N/A	0 (disable both limits), 1 (disable pos. dir. limit only), 2 (disable neg. dir. limit only), 3 (enable both limits)	0
LSAD	☒ Software EOT Limit Decel	--	--	X	X	EOT Limit	<a_><!>LSAD<r>	revs/sec/sec	0.0001 - 9999.9999	100.0000
LSADA	☒ Software EOT Limit Decel (S-Curve)	--	--	X	X	EOT Limit; Motion (S)	<a_><!>LSADA<r>	revs/sec/sec	0.0001 - 9999.9999	100.0000
LSNEG	☒ Software Limit Negative Range	--	--	X	X	EOT Limit	<a_><!>LSNEG<r>	distance (counts)	-2^{31} to $+2^{31}-1$	0
LSPOS	☒ Software Limit Positive Range	--	--	X	X	EOT Limit	<a_><!>LSPOS<r>	distance (counts)	-2^{31} to $+2^{31}-1$	0
MA	Incremental/Absolute (Preset) Positioning Mode	--	--	X	X	Motion	<a_><!>MA<b>	N/A	0 (incremental positioning), 1 (absolute positioning)	0
MC	Preset/Continuous Positioning Mode	--	--	X	X	Motion	<a_><!>MC<b>	N/A	0 (preset positioning), 1 (continuous positioning)	0
NIF	End of IF Statement	--	--	X	X	Prog. Flow	<a_><!>NIF	N/A	N/A	N/A
ORES	♦☒ Resolution of Step & Direction Output (GT) or Encoder Output (GV)	X	X	X	X	Drive Config.	<a_><!>ORES<i>	R: counts/rev L: counts/epitch	GT: 0, 200 - 128000 GV: 0, 200 - 1024000 (0 = disable)	GT: 0 GV: 4000
OUT	Output State Manipulation	--	--	X	X	Outputs	<a_><!>OUT<bbbbbbb> <a_><!>OUT.<i>-<b>	<b> = enable bit <i> = output #	<b>: 0 (off), 1 (on), or X (ignore) <i>: 1 - 7	0
OUTBD	Brake Output Delay	X	X	X	X	Outputs	<a_><!>OUTBD<I>	milliseconds	0 – 1000	0
OUTFNC	☒ Output Function Assignment	--	--	X	X	Outputs	<a_><!>OUTNC<i>-<c>	<i> = output # <c> = function ID	<i>: 1 - 8 <c>: A (general purpose), B (moving/not moving), C (program in progress), D (EOT limit hit), E (stall), F (fault), or G (position error).	OUTFNC1-A OUTFNC2-F OUTFNC3-D OUTFNC4-E OUTFNC5-B OUTFNC6-A OUTFNC7-F
PLN	End of Loop, Compiled Motion	--	--	X	X	Compiled Motion	<a_>PLN	N/A	N/A	N/A
PLOOP	Beginning of Loop, Compiled Motion	--	--	X	X	Compiled Motion	<a_>PLOOP<i>	# of times to loop	0 – 999,999,999 (0 = infinite)	0 (infinite)
OUTLVL	☒ Output Level Sense	X	X	X	X	Outputs	<a_><!>OUTLVL<bb...>	N/A	0 (active low), 1 (active high)	0000000
POUTA	Compiled Output	--	--	X	X	Outputs	<a_><!>POUTA<bb...> <a_><!>POUTA.<i>-<b>	<b> = enable bit <i> = output #	<b>: 0 (off), 1 (on), or X (ignore) <i>: 1 - 7	0
PRUNPROF	Run a Compiled Profile	--	--	X	X	Compiled Motion	<a_><!>PRUN PROF<i>	profile ID #	1 - 16	N/A
PS	Pause Program Execution	--	--	X	X	Prog. Flow	<a_><!>PS	N/A	N/A	N/A
PSET	Establish Absolute Position	--	--	X	X	Motion	<a_><!>PSET<r>	Counts (abs. pos.)	-2^{31} to $+2^{31}-1$	N/A
RE	☒ Registration Enable	--	--	X	X	Registration	<a_><!>RE<b>	enable bit	0 (disable), 1 (enable)	0
REG	Registration Distance	--	--	X	X	Registration	<a_><!>REG<i>	distance (counts)	0 - 2,147,483,647	0
REGLOD	☒ Regist. Lockout Distance	--	--	X	X	Registration	<a_><!>REGLOD<i>	distance (counts)	0 - 2,147,483,647	0
RESET	Reset Drive	X	X	X	X	Comm. I/F	<a_><!>RESET	N/A	N/A	N/A
RFS	Return to Factory Settings	X	X	X	X	Drive Config.	<a_><!>RFS	N/A	N/A	N/A
RUNPROG	Run a Program	--	--	X	X	Prog. Def(n)	<a_><!>RUN PROG<i>	program ID #	1 - 32	N/A
S	Stop Motion	--	--	X	X	Motion	<a_><!>S<b>	N/A	0 (don't stop), 1 (stop)	1
SFB	☒ Select Feedback Source	--	X	--	X	Drive Config.	<a_><!>SFB<i>	feedback selector	1 (encoder), 4 (resolver)	1 (auto)
SGAF	☒ Acceleration Feedforward	--	--	--	X	Tuning (Servo)	<a_><!>SGAF<i>	%	0 - 500	100

♦ = requires reset (RESET, cycle power, or reset input) to apply.
☒ = saved in EEPROM (Gn6: only if executed outside a program).
(auto) = Auto-configured based on motor selection, see DMTR.

Command	Description	GT	GV	GT6	GV6	Type/Group	Syntax	Units	Range	Default
SGENB	Enable a Gain Set	--	--	--	X	Tuning (Servo)	<a_><!>SGENB<i>	gain set ID #	1 - 3	N/A
SGINTE	☒ Integrator Enable	--	X	--	X	Tuning (Servo)	<a_><!>SGINTE<b>	enable bit	0 (disable), 1 (enable)	1
SGIRAT	☒ Current Damping Ratio	--	X	--	X	Tuning (Servo)	<a_><!>SGIRAT<r>	ratio	0.500 - 2.000	1
SGPRAT	☒ Position Damping Ratio	--	X	--	X	Tuning (Servo)	<a_><!>SGPRAT<r>	ratio	0.500 - 2.000	1
SGPSIG	☒ Velocity/Position Bandwidth Ratio	--	X	--	X	Tuning (Servo)	<a_><!>SGPSIG<r>	ratio	0.100 - 2.000	1
SGSET	Save a Gain Set	--	--	--	X	Tuning (Servo)	<a_><!>SGSET<i>	gain set ID #	1 - 3	N/A
SGVF	☒ Velocity Feedforward Gain	--	--	--	X	Tuning (Servo)	<a_><!>SGVF<i>	%	0 - 500	100
SGVRAT	☒ Velocity Damping Ratio	--	X	--	X	Tuning (Servo)	<a_><!>SGVRAT<r>	ratio	0.500 - 2.000	1
SHALL	♦☒ Hall Sensor Inversion	--	X	--	X	Drive Config.	<a_><!>SHALL<i>	option selector	0 (do not invert), 1 (invert)	0
SMPER	☒ Maximum Position Error	--	X	--	X	Drive Config.	<a_><!>SMPER<i>	counts	0 - 2,147,483,647 (0 = disable)	4000
SMVER	☒ Maximum Velocity Error	--	X	--	X	Drive Config.	<a_><!>SMVER<r>	revs/sec	0.000000 - 200.000000 (0 = disable)	0
SRSET	☒ Resolver Offset Angle	--	X	--	X	Drive Config.	<a_><!>SRSET<r>	degrees	1.0 - 180.0 (none = auto set)	0 (auto)
STARTP	☒ Startup Program Assign.	--	--	X	X	Prog. Def(n)	<a_><!>STARTPPROG<i>	program ID #	0, 1 - 32 (0 = unassign)	N/A
STRGTD	☒ Target Zone Distance	--	--	--	X	Target Zone	<a_><!>STRGTD<i>	distance (counts)	0 - 999,999,999	50
STRGTE	☒ Target Zone Mode Enable	--	--	--	X	Target Zone	<a_><!>STRGTE<b>	enable bit	0 (disable), 1 (enable)	0
STRGTT	☒ Target Zone Timeout	--	--	--	X	Target Zone	<a_><!>STRGTT<i>	milliseconds	0 - 5000	1000
STRGTV	☒ Target Zone Velocity	--	--	--	X	Target Zone	<a_><!>STRGTV<r>	R: revs/sec L: meters/sec	R: 0.0000 - 200.0000 L: DMEPIT dependent	1.0000
T	Time Delay (Dwell)	--	--	X	X	Prog. Flow	<a_><!>T<r>	seconds	0.001 - 999.999	N/A
TACC	Transfer Commanded Accel.	--	--	X	X	Transfer	<a_><!>TACC	revs/sec/sec	N/A	N/A
TACCA	Transfer Actual Accel.	--	--	--	X	Transfer	<a_><!>TACCA	revs/sec/sec	N/A	N/A
TANI	Transfer Axis Status	X	X	X	X	Transfer	<a_><!>TANI	volts	N/A	N/A
TAS	Transfer Axis Status	X	X	X	X	Transfer	<a_><!>TAS	bit field	N/A	N/A
TASF	TAS (Full Text Report)	X	X	X	X	Transfer	<a_><!>TASF	N/A	N/A	N/A
TASX	Transfer Extended Axis Status	X	X	X	X	Transfer	<a_><!>TASX	bit field	N/A	N/A
TASXF	TASX (Full Text Report)	X	X	X	X	Transfer	<a_><!>TASXF	N/A	N/A	N/A
TCS	Transfer Config. Status	X	X	X	X	Transfer	<a_><!>TCS	error code	N/A	N/A
TDHRS	☒ Transfer Operating Hours	X	X	X	X	Transfer	<a_><!>TDHRS	hours	N/A	N/A
TDICNT	Transfer Continuous Current Rating	X	X	X	X	Transfer	<a_><!>TDICNT	Amps peak	N/A	N/A
TDIMAX	Transfer Maximum Current Rating	--	X	--	X	Transfer	<a_><!>TDIMAX	Amps peak	N/A	N/A
TDIR	Transfer Programs Stored	--	--	X	X	Transfer	<a_><!>TDIR	N/A	N/A	N/A
TDTEMP	Transfer Drive Temperature	X	X	X	X	Transfer	<a_><!>TDTEMP	degrees C	N/A	N/A
TDVBUS	Transfer Bus Voltage	X	X	X	X	Transfer	<a_><!>TDVBUS	volts	N/A	N/A
TER	Transfer Error Status	X	X	X	X	Transfer	<a_><!>TER	bit field	N/A	N/A
TERF	TER (Full Text Report)	X	X	X	X	Transfer	<a_><!>TERF	N/A	N/A	N/A
TERRLG	☒ Transfer Error Log	X	X	X	X	Transfer	<a_><!>TERRLG	N/A	N/A	N/A
TGAIN	Transfer Active Gains	--	X	--	X	Transfer	<a_><!>TGAIN	N/A	N/A	N/A
THALL	Transfer Hall Sensor Values	--	X	--	X	Transfer	<a_><!>THALL	hall sensor values	N/A	N/A
TIN	Transfer Input Status	X	X	X	X	Transfer	<a_><!>TIN	bit field	N/A	N/A
TINO	Transfer Other Input Status	X	X	X	X	Transfer	<a_><!>TINO	bit field	N/A	N/A
TMEM	Transfer Memory Usage	--	--	X	X	Transfer	<a_><!>TMEM	bytes (prog,prof)	N/A	N/A
TMTEMP	Transfer Motor Temp.	--	X	--	X	Transfer	<a_><!>TMTEMP	degrees C	N/A	N/A
TOUT	Transfer Output Status	X	X	X	X	Transfer	<a_><!>TOUT	bit field	N/A	N/A
TPC	Transfer Commanded Pos.	X	X	X	X	Transfer	<a_><!>TPC	counts	N/A	N/A
TPE	Transfer Actual Position	--	X	--	X	Transfer	<a_><!>TPE	feedback counts	N/A	N/A
TPER	Transfer Position Error	--	X	--	X	Transfer	<a_><!>TPER	counts	N/A	N/A
TPRA	Transfer Abs. Resolver Pos.	--	X	--	X	Transfer	<a_><!>TPRA	counts	0 – (ERES - 1)	N/A
TPROG	Transfer Program Contents	--	--	X	X	Transfer	<a_><!>TPROGPROG<i>	program ID #	1 - 32	N/A
TRACE	☒ Program Trace Mode	--	--	X	X	Debug Tools	<a_><!>TRACE<b>	enable bit	0 (disable), 1 (enable)	0
TREV	Transfer Revision Level	X	X	X	X	Transfer	<a_><!>TREV	N/A	N/A	N/A
TRGFN	Trigger Interrupt Functions (GOBUF on trigger input)	--	--	X	X	Inputs	<a_><!>TRGFN<c><b>	<c> = input ID <b> = enable bit	<c>: A-H (inputs 1-8) <b>: 0 (disable), 1 (enable)	N/A
TRGLOT	☒ Trigger Interrupt Lockout	--	--	X	X	Inputs	<a_><!>TRGLOT<i>	milliseconds	0 - 250	24
TSGSET	Transfer Gain Set	--	--	--	X	Transfer	<a_><!>TSGSET<i>	gain set ID #	1 - 3	N/A
TSROFF	☒ Transfer Resolver Offset Angle	--	X	--	X	Transfer	<a_><!>TSROFF	degrees	-180.0 - +180.0	N/A
TSS	Transfer System Status	X	X	X	X	Transfer	<a_><!>TSS	bit field	N/A	N/A
TSSF	TSS (Full Text Report)	X	X	X	X	Transfer	<a_><!>TSSF	N/A	N/A	N/A
TSTLT	Transfer Settling Time	--	--	--	X	Transfer	<a_><!>TSTLT	milliseconds	N/A	N/A
TTRQ	Transfer Commanded Torque/Force	--	X	--	X	Transfer	<a_><!>TTRQ	% of DMTSCL	N/A	N/A
TTRQA	Transfer Actual Torque/Force	--	X	--	X	Transfer	<a_><!>TTRQA	% of DMTSCL	N/A	N/A
TVE	Transfer Velocity Error	--	X	--	X	Transfer	<a_><!>TVE	revs/sec	-200 - 200	N/A
TVEL	Transfer Commanded Vel.	X	X	X	X	Transfer	<a_><!>TVEL	revs/sec	-200.000000 - 200.000000	N/A
TVELA	Transfer Actual Velocity	--	X	--	X	Transfer	<a_><!>TELA	revs/sec	-200.000000 - 200.000000	N/A
V	Velocity	--	--	X	X	Motion	<a_><!>V<r>	revs/sec	0.0000 - 200.0000	1.0000
VARCLR	Variable Clear	--	--	X	X	Variables	<a_><!>VARCLR	N/A	N/A	N/A
VARI	Variable (Integer)	--	--	X	X	Variables	<a_><!>VARI<i>	variable number	1-99	N/A
VF	Final Velocity	--	--	X	X	Compiled Motion	<a_><!>VF<r>	N/A	0	0
WAIT	Wait for Condition	--	--	X	X	Prog. Flow	<a_><!>WAIT(<op>=b<b>) <a_><!>WAIT(<op>.<i>=b<b>)	<op> = operand <b> = binary state <i> = selected bit	<op>: AS, ASX, IN <b>: 1 or 0 <i>: 1 - 32 (varies by operand)	N/A
XONOFF	☒ XON/XOFF Enable	X	X	X	X	Comm. I/F	<a_><!>XONOFF<b>	enable bit	0 (disable), 1 (enable)	1

♦ = requires reset (RESET, cycle power, or reset input) to apply.
☒ = saved in EEPROM (Gn6: only if executed outside a program).
(auto) = Auto-configured based on motor selection, see DMTR.

Appendix B:
Command Groups

Program Definition		GT	GV	GT6	GV6
DEF PROG	Begin Definition of Program	--	--	X	X
DELPROG	Delete Program	--	--	X	X
END	End Program/Profile Definition	--	--	X	X
ERASE	Erase All Programs and Profiles	--	--	X	X
RUN PROG	Run a Program	--	--	X	X
STARTP	Startup Program Assignment	--	--	X	X
TDIR	Transfer Stored Programs & Profiles	--	--	X	X
TMEM	Transfer Memory Usage	--	--	X	X
TPROG	Transfer Program Contents	--	--	X	X

Motion		GT	GV	GT6	GV6
A	Acceleration	--	--	X	X
AA	Acceleration (S-Curve)	--	--	X	X
AD	Deceleration	--	--	X	X
ADA	Deceleration (S-Curve)	--	--	X	X
D	Distance	--	--	X	X
GO	Initiate Motion	--	--	X	X
K	Kill Motion	--	--	X	X
LHAD	Hardware EOT Limit Decel	--	--	X	X
LHADA	Hardware EOT Limit Decel (S-Curve)	--	--	X	X
LSAD	Software EOT Limit Decel	--	--	X	X
LSADA	Software EOT Limit Decel (S-Curve)	--	--	X	X
MA	Incremental/Absolute (Preset) Positioning Mode	--	--	X	X
MC	Preset/Continuous Positioning Mode	--	--	X	X
PSET	Establish Absolute Position	--	--	X	X
S	Stop Motion	--	--	X	X
V	Velocity	--	--	X	X

Compiled Motion		GT	GV	GT6	GV6
A	Acceleration	--	--	X	X
AD	Deceleration	--	--	X	X
D	Distance	--	--	X	X
DEFPROF	Begin Definition of Profile	--	--	X	X
DELPROF	Delete Profile	--	--	X	X
END	End Program/Profile Def(n)	--	--	X	X
GOBUF	Store Compiled Motion Segment	--	--	X	X
GOWHEN	Conditional GOBUF	--	--	X	X
MC	Preset/Continuous Positioning Mode	--	--	X	X
PLN	End of Loop in Compiled Motion	--	--	X	X
PLOOP	Beginning of Loop in Compiled Motion	--	--	X	X
POUTA	Compiled Output	--	--	X	X
PRUNPROF	Run a Compiled Profile	--	--	X	X
SGENB	Enable a Gain Set	--	--	--	X
TDIR	Transfer Stored Programs & Profiles	--	--	X	X
TRGFN	Trigger Interrupt Functions (GOBUF on trigger input)	--	--	X	X
V	Velocity	--	--	X	X
VF	Final Velocity	--	--	X	X

Communication Interface		GT	GV	GT6	GV6
ADDR	Multiple Unit Auto-Address	X	X	X	X
BOT	Beginning Transmission Characters	X	X	X	X
E	Enable Communication	X	X	X	X
ECHO	Communication Echo	X	X	X	X
EOL	End-of-Line Characters	X	X	X	X
EOT	End-of-Transmission Characters	X	X	X	X
ERRBAD	Error Prompt	X	X	X	X
ERRDEF	Program Definition Prompt	X	X	X	X
ERRLVL	Error Detection Level	X	X	X	X
ERROK	Good Prompt	X	X	X	X
RESET	Reset Drive	X	X	X	X
XONOFF	XON/XOFF	X	X	X	X

Command Buffer Control		GT	GV	GT6	GV6
COMEXC	Continuous Command Processing Mode	--	--	X	X
COMEXL	Continue Execution on Limit	--	--	X	X
COMEXR	Continue Motion on Pause/Cont. Input	--	--	X	X
COMEXS	Continue Execution on Stop	--	--	X	X

Program Flow		GT	GV	GT6	GV6
C	Continue Command Exec.	--	--	X	X
ELSE	Else Condition (IF = false)	--	--	X	X
GOSUB	Call a Subroutine	--	--	X	X
IF	IF Conditional Statement	--	--	X	X
JUMP	Jump to a Program	--	--	X	X
L	Loop	--	--	X	X
LN	End of Loop	--	--	X	X
NIF	End of IF Statement	--	--	X	X
PS	Pause Program Execution	--	--	X	X
T	Time Delay (Dwell)	--	--	X	X
WAIT	Wait for Condition	--	--	X	X

Error Handling		GT	GV	GT6	GV6
CERRLG	Clear Error Log	X	X	X	X
ERROR	Error Checking Enable	--	--	X	X
ERRORP	Error Program Assignment	--	--	X	X
TER	Transfer Error Status	X	X	X	X
TERRLG	Transfer Error Log	X	X	X	X

Drive Configuration		GT	GV	GT6	GV6
ANICDB	Analog Input Center Deadband	X	X	--	--
DABSD	Enable ABS Damping	X	--	X	--
DACTDP	Active Damping Gain	X	--	X	--
DAUTOS	Auto Standby Reduction	X	--	X	--
DCLRLR	Clear Latched Status Register	X	X	X	X
DCMDZ	Zero The Drive Command Offset	X	X	--	--
DDAMPA	Damping During Accel	X	--	X	--
DELVIS	Electronic Viscosity Enable	X	--	X	--
DIFOLD	Current Foldback Enable	--	X	--	X
DMODE	Drive Control Mode	X	X	X	X
DMTR	Identify Motor	X	X	X	X
DMTSCL	Torque/Force Scaling	--	X	--	--
DMVSCL	Velocity Scaling	X	X	--	--
DPHBAL	Phase Balance	X	--	X	--
DPHOFA	Phase A Current Offset	X	--	X	--
DPHOFB	Phase B Current Offset	X	--	X	--
DPWM	Drive PWM Frequency	--	X	--	X
DRES	Drive Resolution	X	X	X	--
DRIVE	Drive Enable	X	X	X	X
DSTALL	Stall Detect Sensitivity	X	--	X	--
DWAVEF	Waveform	X	--	X	--
ERES	Encoder/Resolver Resolution	--	X	--	X
ESK	Fault on Stall Enable	X	--	X	--
FLTDSB	Fault on Disable (DRIVE0)	X	X	X	X
FLTSTP	Fault on Startup Indexer Pulses	X	X	--	--
KDRIVE	Disable Drive on Kill	--	--	X	X
ORES	Resolution of Step and Direction Output (GT) or Encoder Output (GV)	X	X	X	X
RESET	Reset Drive	X	X	X	X
RFS	Return to Factory Settings	X	X	X	X
SFB	Servo Feedback Source Selection	--	X	--	X
SHALL	Hall Sensor Inversion	--	X	--	X
SMPER	Maximum Position Error	--	X	--	X
SMVER	Maximum Velocity Error	--	X	--	X
SRSET	Resolver Offset Angle	--	X	--	X

Tuning		GT	GV	GT6	GV6
DIBW	Current Loop Bandwidth	--	X	--	X
DIGN	Current Loop Gain	X	--	X	--
DNOTAD	Notch Filter A Depth	--	X	--	X
DNOTAF	Notch Filter A Frequency	--	X	--	X
DNOTAQ	Notch Filter A Quality Factor	--	X	--	X
DNOTBD	Notch Filter B Depth	--	X	--	X
DNOTBF	Notch Filter B Frequency	--	X	--	X
DNOTBQ	Notch Filter A Quality Factor	--	X	--	X
DNOTLD	Notch Lead Filter Break Frequency	--	X	--	X
DNOTLG	Notch Lag Filter Break Frequency	--	X	--	X
DPBW	Position Loop Bandwidth	--	X	--	X
DVBW	Velocity Loop Bandwidth	--	X	--	X

Tuning – continued on next page

Tuning	*(continued)*	GT	GV	GT6	GV6
SGAF	Acceleration Feedforward	--	--	--	X
SGENB	Enable a Gain Set	--	--	--	X
SGINTE	Integrator Enable	--	X	--	X
SGIRAT	Current Damping Ratio	--	X	--	X
SGPRAT	Position Damping Ratio	--	X	--	X
SGPSIG	Velocity/Position Bandwidth Ratio	--	X	--	X
SGSET	Save a Gain Set	--	--	--	X
SGVF	Velocity Feedforward Gain	--	--	--	X
SGVRAT	Velocity Damping Ratio	--	X	--	X
TGAIN	Transfer Active Gains	--	X	--	X
TSGSET	Transfer Gain Set	--	--	--	X

Motor Configuration		GT	GV	GT6	GV6
DMEPIT	Motor Electrical Pitch	--	X	--	X
DMTAMB	Motor Ambient Temperature	--	X	--	X
DMTD	Motor Damping	--	X	--	X
DMTIC	Continuous Current	X	X	X	X
DMTICD	Continuous Current Derating	--	X	--	X
DMTIND	Motor Inductance	X	--	X	--
DMTIP	Peak Current	--	X	--	X
DMTJ	Motor Rotor Inertia or Motor Mass	X	X	X	X
DMTKE	Motor Ke	--	X	--	X
DMTLMN	Minimum Motor Inductance	--	X	--	X
DMTLMX	Maximum Motor Inductance	--	X	--	X
DMTMAX	Maximum Motor Winding Temp	--	X	--	X
DMTRES	Motor Winding Resistance	X	X	X	X
DMTRWC	Motor Winding Thermal Resistance	--	X	--	X
DMTSTT	Motor Static Torque	X	--	X	--
DMTTCM	Motor Thermal Time Constant	--	X	--	X
DMTTCW	Motor Winding Time Constant	--	X	--	X
DMTW	Motor Rated Speed	--	X	--	X
DPOLE	Motor Pole Pairs	X	X	X	X
TCS	Transfer Configuration Status	X	X	X	X

System		GT	GV	GT6	GV6
DMTLIM	Torque/Force Limit	--	X	--	X
DMVLIM	Velocity Limit	X	X	X	X
LDAMP	Load Damping	--	X	--	X
LJRAT	Load-to-Rotor Inertia Ratio or Load-to-Forcer Ratio	X	X	X	X
TSS	Transfer System Status	X	X	X	X

Inputs		GT	GV	GT6	GV6
INDEB	Input Debounce Time	X	X	X	X
INFNC	Input Function Assignment	--	--	X	X
INLVL	Input Level Sense	X	X	X	X
INSELP	Program Select Mode	--	--	X	X
INUFD	User Fault Input Delay	X	X	X	X
TRGFN	Trigger Interrupt Functions (GOBUF on trigger input)	--	--	X	X
TRGLOT	Trigger Interrupt Lockout	--	--	X	X
TIN	Transfer Input Status	X	X	X	X
TINO	Transfer Enable Input Status	X	X	X	X

Outputs		GT	GV	GT6	GV6
DMONAS	Analog Monitor Output A Scaling	X	X	X	X
DMONAV	Analog Monitor Output A Variable	X	X	X	X
DMONBS	Analog Monitor Output B Scaling	X	X	X	X
DMONBV	Analog Monitor Output B Variable	X	X	X	X
ORES	Resolution of Step & Direction Output (GT) or Encoder Output (GV)	X	X	X	X
OUT	Output State Manipulation	--	--	X	X
OUTBD	Brake Output Delay	X	X	X	X
OUTFNC	Output Function Assignment	--	--	X	X
OUTLVL	Output Level Sense	X	X	X	X
POUTA	Compiled Output	--	--	X	X
TOUT	Transfer Output Status	X	X	X	X

End-of-Travel Limits		GT	GV	GT6	GV6
LH	Hardware End-of-Travel Limit Enable	X	X	X	X
LHAD	Hardware EOT Limit Decel	--	--	X	X
LHADA	Hardware EOT Limit Decel (S-Curve)	--	--	X	X
LS	Software End-of-Travel Limit Enable	--	--	X	X
LSAD	Software EOT Limit Decel	--	--	X	X
LSADA	Software EOT Limit Decel (S-Curve)	--	--	X	X
LSNEG	Software Limit Neg. Range	--	--	X	X
LSPOS	Software Limit Pos. Range	--	--	X	X
TIN	Transfer Input Status	X	X	X	X

Homing		GT	GV	GT6	GV6
HOM	Go Home	--	--	X	X
HOMA	Home Acceleration	--	--	X	X
HOMBAC	Home Backup	--	--	X	X
HOMDF	Home Final Direction	--	--	X	X
HOMEDG	Home Reference Edge	--	--	X	X
HOMV	Home Velocity	--	--	X	X
HOMVF	Home Final Velocity	--	--	X	X
HOMZ	Home to Encoder Z Channel	--	--	--	X

Target Zone		GT	GV	GT6	GV6
STRGTD	Target Zone Distance	--	--	--	X
STRGTE	Target Zone Mode Enable	--	--	--	X
STRGTT	Target Zone Timeout	--	--	--	X
STRGTV	Target Zone Velocity	--	--	--	X
TSTLT	Transfer Settling Time	--	--	--	X

Registration		GT	GV	GT6	GV6
RE	Registration Enable	--	--	X	X
REG	Registration Distance	--	--	X	X
REGLOD	Registration Lockout Distance	--	--	X	X

Transfer (Display/Report)		GT	GV	GT6	GV6
TACC	Transfer Commanded Acceleration	--	--	X	X
TACCA	Transfer Actual Accel.	--	--	--	X
TANI	Transfer Analog Input Voltage	X	X	--	--
TAS	Transfer Axis Status	X	X	X	X
TASF	Transfer Axis Status (full-text report)	X	X	X	X
TASX	Transfer Extended Axis Status	X	X	X	X
TASXF	TASX (full-text report)	X	X	X	X
TCS	Transfer Configuration Status	X	X	X	X
TDHRS	Transfer Operating Hours	X	X	X	X
TDICNT	Transfer Continuous Current Rating	X	X	X	X
TDIMAX	Transfer Maximum Current Rating	--	X	--	X
TDIR	Transfer Stored Programs & Profiles	--	--	X	X
TDTEMP	Transfer Drive Temperature	X	X	X	X
TDVBUS	Transfer Bus Voltage	X	X	X	X
TER	Transfer Error Status	X	X	X	X
TERF	Transfer Error Status (full-text report)	X	X	X	X
TERRLG	Transfer Error Log	X	X	X	X
TGAIN	Transfer Active Gains	--	X	--	X
THALL	Hall Sensor Values	--	X	--	X
TIN	Transfer Input Status	X	X	X	X
TINO	Transfer Other Input Status	X	X	X	X
TMEM	Transfer Memory Usage	--	--	X	X
TMTEMP	Transfer Motor Temp.	--	X	--	X
TOUT	Transfer Output Status	X	X	X	X
TPC	Transfer Commanded Pos.	X	X	X	X
TPE	Transfer Actual Position	--	X	--	X
TPER	Transfer Position Error	--	X	--	X
TPRA	Transfer Absolute Resolver Position	--	X	--	X
TPROG	Transfer Program Contents	--	--	X	X
TREV	Transfer Revision Level	X	X	X	X
TSGSET	Transfer Gain Set	--	--	--	X
TSROFF	Transfer Resolver Offset Angle	--	X	--	X
TSS	Transfer System Status	X	X	X	X
TSSF	Transfer System Status (full-text report)	X	X	X	X
TSTLT	Transfer Settling Time	--	--	--	X
TTRQ	Transfer Commanded Torque/Force	--	X	--	X
TTRQA	Transfer Actual Torque/Force	--	X	--	X
TVE	Transfer Velocity Error	--	X	--	X
TVEL	Transfer Commanded Vel.	X	X	X	X
TVELA	Transfer Actual Velocity	--	X	--	X

Variables		GT	GV	GT6	GV6
VARCLR	Variable Clear	--	--	X	X
VARI	Integer Variable	--	--	X	X

Debug Tools *(see also Transfer commands)*		GT	GV	GT6	GV6
TAS	Transfer Axis Status	X	X	X	X
TASX	Transfer Extended Axis Status	X	X	X	X
TCS	Transfer Configuration Status	X	X	X	X
TERRLG	Transfer Error Log	X	X	X	X
THALL	Hall Sensor Values	--	X	--	X
TRACE	Program Trace Mode	--	--	X	X

Appendix C:
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	RS
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	∅
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	¢
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	¡
174	AE	«
175	AF	»
176	B0	▒
177	B1	▓
178	B2	▓
179	B3	¦
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	┬
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	Ï
245	F5	⌠
246	F6	÷
247	F7	≈
248	F8	°
249	F9	•
250	FA	·
251	FB	√
252	FC	η
253	FD	2
254	FE	●
255	FF	

Appendix D:
Communications Server
(COM6SRVR.EXE)

The Communications Server (COM6SRVR.EXE) is a 32-bit OLE automation server which facilitates communications between Gemini drives (as well as 6K controllers) and PC software applications. It is compatible with any 32-bit software application or programming environment which can utilize an OLE automation component, including the Visual Basic, Visual C++, and Delphi. The Motion Planner installation program installs COM6SRVR.EXE in the Motion Planner directory.

To begin serial (RS-232 or RS-485) communications, an application simply needs to request a connection to a Gemini drive through the Communications Server. (You need to specify the PC COM port on which to connect.) The Communications Server manages the actual connection to each Gemini drive, and can feed information from a particular drive to all client applications which require the information.

Although the Communications Server only makes one connection to each Gemini drive, it can feed the information from that one connection to multiple client applications. This means, for example, that a terminal application created in Visual Basic and a terminal in Motion Planner can be connected to the same Gemini at the same time. They will both receive the same responses coming from the drive, instead of competing for the data. It is also possible for an application to request multiple connections to multiple Gemini units via the Communications Server.

The syntax for requesting a connection to the Communications Server varies depending on the programming environment being used. Page 196 provides examples in the Visual Basic, Visual C++, and Delphi programming formats (refer also to the samples in the Motion Planner directory). To disconnect, refer to “How to Disconnect” instructions on page 197.

COM6SRVR Application Programming Interface (API): Once the proper object variable has been created and a connection is established, there is a standard set of Gemini methods which the client application(s) can access (see page 198).

Visual Basic Connection Example

```
'create an object variable, initialize it to a serial interface
'with the Gemini and make a connection to PC COM1

Dim MyMachine As Object
Dim ConnectReturnValue As Integer
Set MyMachine = CreateObject("COM6SRVR.GEMINI")
ConnectReturnValue = MyMachine.Connect(1)
```

Note: When using VBScript, the syntax is identical to the example above, except that the variable declaration should omit the "`As Object`" and "`As Integer`" keywords.

Visual C++ Connection Example

```
1. Add the class com6srvr.tlb to your project. This file is located in
C:\Window\System under the default Motion Planner installation.

2. In the project header file, add the line #include "com6srvr.tlb"

3. In the main project source file, there should be a function BOOL
C project App::InitInstance(). Add the line AfxO1eInit().

4. To initialize the communication server object and establish a
connection with RS-232 in your program, include the following lines:
     IGemini MyMachine;
     MyMachine.CreateDispatch ("COM6SRVR.GEMINI");
     int ConnectReturnValue = MyMachine.Connect(2);  /*connect to COM2*/
```

Delphi Connection Example

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ComObj;

type
  TForm1 = class(TForm)
    Button1: TButton;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
    CommServer: Variant;         { Create the object variable }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin
  { Initialize CommServer object to a Gemini interface }
  CommServer := CreateOleObject('COM6SRVR.GEMINI);
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  { Make a serial connection to the Gemini drive on COM2 }
  CommServer.Connect(2);
end;

end.
```

How to Disconnect

The Communications Server is designed as an "EXE" (out-of-process) server rather than a "DLL" (in-process) server. This means that it runs independently of the client application's process. This feature allows the same data from the Communications Server to be shared among several clients. It also provides a more secure connection model by insulating the Communications Server from failure on any singular client.

With the use of an *in-process* server, the server itself runs in the client's process. If the client application fails or shuts down, the server will be shutdown along with the client. With the use of an *out-of-process* server, the server runs independently of the client and is therefore insulated from a failure in the client's process. If a particular client application fails, the server will continue to run and provide data to any other client applications requiring its service.

As an out-of-process server, the Communications Server does not shutdown until all client applications have disconnected from the server. In many cases, a proper disconnect does not take place if an unhandled error occurs in the client application and the program exits abnormally. This means that care must be exercised on the part of the client program to disconnect from the server on such occasions or when its services are no longer needed.

VB and VBScript

For VB/VBScript applications, an object variable is typically released when the variable loses scope. However, it is always a good practice to explicitly release the object by setting it to nothing.

```
'assuming the commserver is an object variable
'representing a Communications Server connection

Set commserver = Nothing;   'free the object - disconnect from the
server
```

C++

In C++, the same rule applies to the scope of an object variable, but again it is good programming practice to explicitly release the object.

```
//assuming the commserver is an object variable
//representing a Communications Server connection

commserver.ReleaseDispatch();     // release the IDispatch connection
```

Delphi

Again, in Delphi, the same rule applies.

```
{ assuming the CommServer is an object variable        }
{ representing a Communications Server connection   }

CommServer := UnAssigned;     { release the connection }
```

Gemini Methods

Read ( ) The Read method retrieves command responses from the drive. There are no arguments for this method. The read method does not wait for incoming responses from the drive. It returns immediately with a string containing the drive's response at the time of the request. If no response is available, this method will return an empty string.

If you were to call the Read method twice in rapid succession, you could get the response in one call to Read, and the `ERROK` characters or `ERRBAD` characters in the next call to Read. If this poses a problem, there are three primary alternatives:

- Call the Read method until you receive the `ERROK` or `ERRBAD` characters.
- Use the `ERRLVL0` setting or the `ERRLVL2` setting, both of which eliminate the transmission of `ERROK` and `ERRBAD` characters.
- Wait a longer period of time before the subsequent Read, but be aware that this may degrade performance to an unacceptable level.

Flush The Flush method removes all characters from the client's receive buffer. This method allows you to clear the receive buffer prior to making a read.

USE WITH CAUTION. This method allows you to clear the receive buffer, such that a subsequent Read call can yield a clean response. However, data arriving in the receive buffer is asynchronous to the application program and a thorough understanding of how the application program is structured is necessary to use this method correctly (for example, it would not be beneficial to Flush the buffer if only a partial response has been received).

SendFile (*filename*) The SendFile method is used to download files to the drive. The **filename** argument represents the name of the program file (containing Gemini commands) to be downloaded. If the filename is an empty string, then you will be prompted for the filename. The method returns a positive value (long integer) if the operation is successful; otherwise, it returns an error code (see error code table below).

SendOS (*filename*) The SendOS method downloads the Gemini drive's operating system. The **filename** argument represents the name of operating system file. If filename is an empty string, then you will be prompted for the operating system file name. The method returns a TRUE value if the operation is successful; otherwise, a FALSE value is returned. (This method returns a Boolean value.)

Write (cmd) The Write method is used to send commands to the drive. **cmd** is a string of commands to be sent. Multiple commands can be sent, but each command should be separated with a valid command delimiter (colon, carriage return, line feed). The command string should be limited to 1024 characters or less. Excessively large command strings, may cause an overflow in the drive's command buffer. This method returns a positive value (long integer) corresponding to the number of bytes sent, or a negative error code (see error code table below).

Error Codes

Error Code	Description
-1	Bad Connection
-2	Connection was shutdown
-3	Connection attempt failed
-4	Maximum number of connections exceeded
-5	Connection not yet established
-6	Reserved
-7	Unable to locate specified file
-8	Unable to open specified file

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

Gemini Series Command Reference

General Programming

- Explanation of command description format — see page 2.
- Command syntax guidelines — see pages 3 & 4.
- Programming tools, to assist with configuration, terminal emulation, developing programs (GT6 & GV6), and downloading and uploading files:
  - Motion Planner, for Windows 95/98/NT — see page 5.
  - Pocket Motion Planner, for Windows CE — see page 9.
  - Communications Server (32-bit OLE automation server for communication between the Gemini and your custom Windows applications) — see page 195.

  NOTE: The commands described in this document can be used only with Motion Planner, Pocket Motion Planner, or the Communications Server.
- Programming guide for GT6 & GV6 users — see page 19.
  - Programming fundamentals (creating and executing programs, program flow control, error handling, etc.).
  - Basic operation setup (setup wizards, resetting the drive, limits and homing, target zone, programmable I/O, etc.).
  - Motion programming (basic and advanced profiling).
- **CAUTION — 6K Users**: If you are using a 6K Controller and Gemini Drives, you should try to use two COM ports – one for the 6K and one for the Gemini Drive. If your computer has only one COM port, you will have to swap the serial cable connection between the 6K and the Gemini. Be aware that you must quit Motion Planner before swapping the serial cable between products. After the cable swap, you may launch Motion Planner and select the newly connected product. Failure to quit Motion Planner before the swap will corrupt communications with the attached Gemini or 6K.
- Drive configuration procedures:
  - If you are using Motion Planner — see page 6.
  - If you are using Pocket Motion Planner — see page 11.
- Drive operating system (firmware) update. The file, GEM_*n_nn*.ops, can be downloaded from the Compumotor web site (http://www.compumotor.com).
  - If you are using Motion Planner — see page 8.
  - If you are using Pocket Motion Planner — see page 13.
- Parker motor configuration table update. The file, GEM_motors.mtr, is used by Motion Planner and Pocket Motion Planner to auto-configure various operating parameters for specific motors. This file can be downloaded from the Compumotor web site (http://www.compumotor.com).
  - If you are using Motion Planner, place the updated file in the Motion Planner directory (default location is \Program Files\Compumotor\Motion Planner).
  - If you are using Pocket Motion Planner, transfer the updated file to your hand-held PC and copy it to the \My Documents\Gemini directory.
- Command quick-reference tables (syntax, range, etc.):
  - Commands listed alphabetically — see page 187.
  - Commands listed by functional group — see page 191.

Troubleshooting

- LED diagnostic table — see page 15.
- Error messages — see page 15.
- Commonly used status commands (binary status bits are numbered 1 to n, from left to right):
  - `TERRLG` ..... Error log reports the last 10 error conditions (cleared with `CERRLG`).
  - `TAS` ..... General report, including fault conditions.
  - `TASX` ..... Additional report of conditions not covered with `TAS`.
  - `TCS` ..... If `TASX` bit #7 or bit #28 is set, you can identify the cause with `TCS`.
  - `TINO` ..... Bit #6 indicates status of Enable input ("1" = OK to enable drive).
  - `TIN` ..... Status of digital inputs, including end-of-travel inputs.
  - `TOUT` ..... Status of digital outputs.
- Hardware and serial communication problems – refer to the Troubleshooting chapter in your Gemini drive's *Hardware Installation Guide*.
- You must configure all motor parameters (see `DMTR` for list); otherwise, the drive will detect a motor configuration error, which prevents the drive from being enabled. Be sure to follow the drive configuration procedure (page 6 for Motion Planner, page 11 for Pocket Motion Planner) to help avoid this error condition.
- Any fault condition (see `TAS` and `TASX` reports) will cause the drive to shut down.
- The drive cannot be enabled (`DRIVE1`) unless the Enable input is grounded and the Reset input is not grounded.
- The GT6 & GV6 drives are shipped from the factory with the hardware end-of-travel limits enabled (`LH3`), but not connected. Therefore, motion will not be allowed until you do one of the following:
  - Install limit switches or jumper the end-of-travel limit terminals to ground.
  - Disable the limits with the `LH0` command (only if the motor is not coupled to the load).
  - Reverse the active level of the limits with `INLVL00`.
- There are three methods of resetting the drive (all command settings are remembered after reset):
  - Issue the `RESET` command.
  - Momentarily close the Reset input.
  - Cycle power to the drive.
- To return the drive to its factory default settings, issue the `RFS` command. Be sure to save the current configuration first (in Motion Planner or Pocket Motion Planner), in case you need to restore the most recent settings. GT6 & GV6 users: Save your program/profiles files.
- If you need technical assistance, contact your local ATC or distributor, or refer to the numbers and internet addresses listed on the inside of this manual's front cover.

Parker Automation

Parker Hannifin Corporation
Compumotor Division
Rohnert Park, CA, USA
http://www.compumotor.com