

# Command Descriptions

---

<b>A</b>		<b>Acceleration</b>			
Type	Motion			<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>A<r>			GT	n/a
Units	r = revs/sec/sec (linear motors: see page 44 for linear/rotary conversion)			GV	n/a
Range	0.0001 - 9999.9999			GT6	1.50
Default	10.0000			GV6	1.50
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

```

---

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

```

---

<b>AD</b>		<b>Deceleration</b>			
Type	Motion			<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>AD<r>			GT	n/a
Units	r = revs/sec/sec (linear motors: see page 44 for linear/rotary conversion)			GV	n/a
Range	0, or 0.0001 - 9999.9999			GT6	1.50
Default	10.0000 (AD tracks A; to restore tracking, set AD = 0)			GV6	1.50
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)	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>ADA<r>	GT	n/a
Units	r = revs/sec/sec (linear motors: see page 44 for linear/rotary conversion)	GV	n/a
Range	0, or 0.0001 - 9999.9999	GT6	1.50
Default	10.0000 (default is a constant deceleration ramp, where ADA tracks AA; to restore tracking, set ADA = 0)	GV6	1.50
Response	ADA: *ADA10.0000		
See Also	A, AA, AD, DMEPIT, LHADA, LSADA		

---

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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>ADDR<i>	GT	1.02
Units	i = unit number (address)	GV	1.00
Range	0-99	GT6	1.50
Default	0	GV6	1.50
Response	ADDR: *ADDR0		
See Also	E		

---

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 ADDR<sub>i</sub> 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  $\emptyset\_ADDR1$  command to it.
2. Connect the unit that is to be unit #2 and transmit the  $\emptyset\_ADDR2$  command to it.
3. Connect the unit that is to be unit #3 and transmit the  $\emptyset\_ADDR3$  command to it.
4. Connect the unit that is to be unit #4 and transmit the  $\emptyset\_ADDR4$  command to it.

If you need to replace a unit in the multi drop, send the  $\emptyset\_ADDRi$  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\_OUT $\emptyset$  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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>ANICDB<r>	GT	1.61
Units	volts	GV	1.70
Range	0.00 - 10.00	GT6	n/a
Default	0.04	GV6	n/a
Response	ANICDB: *ANICDB0.04		
See Also	DCMDZ, DMTSCL, DMVSCL		

---

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:

$$\begin{aligned}
 Vel_{command} &= (V_{in} - DCMDZ - ANICDB) * \frac{DMVSCL}{10} && \text{when } V_{in} > (DCMDZ + ANICDB) \\
 Vel_{command} &= 0 && \text{when } (DCMDZ - ANICDB) \leq V_{in} \leq (DCMDZ + ANICDB) \\
 Vel_{command} &= (V_{in} - DCMDZ + ANICDB) * \frac{DMVSCL}{10} && \text{when } V_{in} < (DCMDZ - ANICDB)
 \end{aligned}$$

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

---

## BOT Beginning of Transmission Characters

Type	Communication Interface	<b>Product</b>	<b>Rev</b>
Syntax	<i>BOT<i>,<i>,<i>	GT	1.02
Units	i = numeric equivalent for ASCII character	GV	1.00
Range	i = 0-255	GT6	1.50
Default	0,0,0	GV6	1.50
Response	BOT: *BOT0,0,0		
See Also	EOT, ERROK, ERBAD, EOL		

---

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	<b>Product</b>	<b>Rev</b>
Syntax	<a_>!C	GT	n/a
Units	n/a	GV	n/a
Range	n/a	GT6	1.50
Default	n/a	GV6	1.50
Response	n/a		
See Also	COMEXR, COMEXS, INFNC, PS, S		

---

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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>CERRLG	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	TAS, TASX, TDHRS, TDTEMP, TERRLG, TMTEMP		

---

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

The error log is updated every time an 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

Type	Command Buffer Control	Product	Rev
Syntax	<a_><!>COMEXC<b>	GT	n/a
Units	b = 0 or 1	GV	n/a
Range	0 = Disable, 1 = Enable	GT6	1.50
Default	0	GV6	1.50
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.

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

With continuous command execution enabled (COMEXC1), if you wish motion to stop before executing the subsequent move, place a WAIT(AS.1=b0) statement before the subsequent GO command. If you wish to ensure the load settles adequately before the next move, use the WAIT(AS.24=b1) command instead (this requires you to define end-of-move settling criteria — see STRGTE command or the *Target Zone* section 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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>COMEXL<b>	GT	n/a
Units	b = 0 or 1	GV	n/a
Range	0 = Disable, 1 = Enable	GT6	1.50
Default	0	GV6	1.50
Response	COMEXL: *COMEXL0		
See Also	COMEXC, COMEXS, ERROR, LH, LS		

---

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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>COMEXR<b>	GT	n/a
Units	b = 0 or 1	GV	n/a
Range	0 = disable, 1 = enable	GT6	1.50
Default	0	GV6	1.50
Response	COMEXR: *COMEXR0		
See Also	C, COMEXS, INFNC		

---

The Continue Motion on Pause/Continue (COMEXR) command determines the functionality of programmable inputs defined as pause/continue inputs with the INFNCi-E command. 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

Type	Command Buffer Control	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>COMEXS<i>	GT	n/a
Units	i = function identifier	GV	n/a
Range	0, 1, or 2	GT6	1.50
Default	0	GV6	1.50
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?		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.
	Motion	Program			
!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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>D<r>	GT	n/a
Units	r = distance (counts)	GV	n/a
Range	-2,147,483,648 to +2,147,483,647	GT6	1.50
Default	4000	GV6	1.50
Response	D: *D+4000		
See Also	DMEPIT, DRES, ERES, GO, IF, MA, MC, PSET, VARI, WAIT		

---

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.

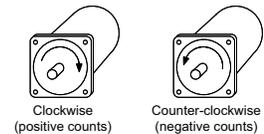
In the incremental mode (MAØ), you can specify a negative distance by placing a dash or hyphen (-) in front of the distance value (e.g., D-10000). Otherwise, the direction is considered positive. You can change direction without changing the distance value by using the +, -, or ~ operators (e.g. D+, 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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DABSD<b>	GT	1.02
Units	b = enable bit	GV	n/a
Range	0 (disable) or 1 (enable)	GT6	1.50
Default	0 (disabled)	GV6	n/a
Response	DABSD: *DABSD1		
See Also	DACTDP, DDAMPA, DELVIS		

---

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

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

\* These features are based on motor and load parameters that you set up with the configuration utility in Motion Planner (see page 6) 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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DACTDP<i>	GT	1.02
Units	i = gain	GV	n/a
Range	0-40; 0 disables active damping	GT6	1.50
Default	4	GV6	n/a
Response	DACTDP *DACTDP0		
See Also	DABSD, DELVIS, DMTR, LJRAT		

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

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

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

---

## DAUTOS      Auto Current Standby

Type	Drive Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DAUTOS<r>	GT	1.02
Units	r = % reduction of motor current	GV	n/a
Range	0.00-99.99 : ±0.01	GT6	1.50
Default	0.00 (no current reduction)	GV6	n/a
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

Type	Drive Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DCLRLR	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	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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DCMDZ<r>	GT	1.02
Units	volts	GV	1.00
Range	-10.00 - 10.00	GT6	n/a
Default	0.00	GV6	n/a
Response	n/a		
See Also	ANICDB, DMODE		

---

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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DDAMPA<b>	GT	1.02
Units	b = enable bit	GV	n/a
Range	0 (disable damping) or 1 (enable damping)	GT6	1.50
Default	0 disabled)	GV6	n/a
Response	DDAMPA: *DDAMPA1		
See Also	DABSD, DELVIS		

---

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

If your application requires more responsive acceleration and deceleration, you can disable Damping During Accel/Decel with the DDAMPA0 command (DDAMPA0 is the factory default setting). This disables ABS damping and Electronic Viscosity during acceleration and deceleration rates greater than 50 revs/sec/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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DEF PROF<i>	GT	n/a
Units	i = profile ID number	GV	n/a
Range	1-16	GT6	1.50
Default	n/a	GV6	1.50
Response	n/a		
See Also	DEF PROF, DEL PROF, END, GOBUF, PRUN PROF, TDIR, TMEM, TSS (see <i>Compiled Motion</i> section on page 49)		

---

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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DEF PROG<i>	GT	n/a
Units	i = program ID number	GV	n/a
Range	1-32	GT6	1.50
Default	n/a	GV6	1.50
Response	n/a		
See Also	DEF PROF, DEL PROF, END, GOSUB, JUMP, RUN PROG, STARTP, TDIR, TMEM, TPROG, TSS		

---

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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DEF PROF<i>	GT	n/a
Units	i = profile ID number	GV	n/a
Range	1-16	GT6	1.50
Default	n/a	GV6	1.50
Response	n/a		
See Also	DEF PROF, END, PRUN PROF, TDIR, TMEM, TSS, (Compiled Motion overview on page 49)		

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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DEF PROG<i>	GT	n/a
Units	i = program ID number	GV	n/a
Range	1-32	GT6	1.50
Default	n/a	GV6	1.50
Response	n/a		
See Also	DEF PROG, END, RUN PROG, TDIR, TMEM, TPROG, TSS		

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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DELVIS<i>	GT	1.02
Units	i = gain	GV	n/a
Range	0-7 (0 disables Electronic Viscosity)	GT6	1.50
Default	0 (disabled)	GV6	n/a
Response	DELVIS    *DELVIS0		
See Also	DACTDP, DABSD, DDAMPA, DMTR, LJRAT		

---

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

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

An overview of the 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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DIBW<i>	GT	n/a
Units	i = Hz	GV	1.00
Range	0-5000 (motor dependent)	GT6	n/a
Default	0 (DIBW of 0 results in motor configuration error)	GV6	1.50
Response	DIBW:      *DIBW1000		
See Also	DMTLIM, DMTSCL, DMVLIM, DNOTAF, DNOTAQ, DNOTLD, DNOTLG, DPBW, DVBW, SGIRAT, TASX, TCS, TGAIN, TSGSET		

---

**AUTO-SETUP:** This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6) 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	Product	Rev
Syntax	<a_><!>DIFOLD<b>	GT	n/a
Units	b = enable bit	GV	1.00
Range	0 (disable) or 1 (enable)	GT6	n/a
Default	0	GV6	1.50
Response	DIFOLD: *DIFOLD0		
See Also			

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

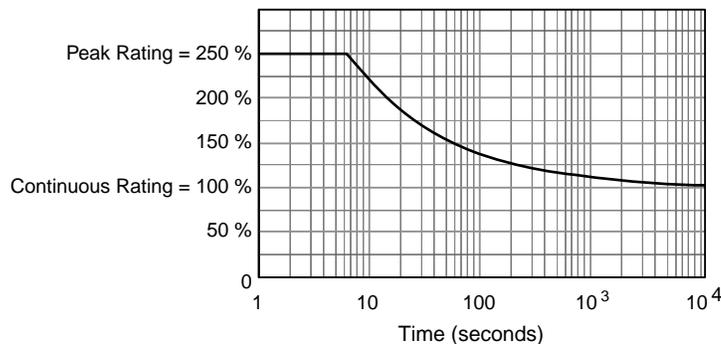
Each drive has the following specifications. Note that current ratings are for the *drive*, not for the *motor*.

	Units	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	Product	Rev
Syntax	<a_><!>DIGNc<r>	GT	1.02
Units	c = gain identifier letter (required); r = gain value	GV	n/a
Range	c = A, B, C, or D; DIGNA, DIGNB, DIGNC : r = 0.000 to 15.000 : ±0.001 DIGND : r = 0.000 to 1.000 : ±0.001	GT6	1.50
Default	r = 0.000 (DIGNc of 0 results in motor config. warning)	GV6	n/a
Response	DIGNA: *DIGNA2.306		
See Also	DMTR, TASX, TCS		

**AUTO-SETUP:** This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6) 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 \* : 
$$\text{DIGNA} = \frac{\text{Inductance (in mH)}}{2}$$

$$\text{DIGNB} = \text{DIGNA} * 0.0896$$

$$\text{DIGNC} = \text{DIGNA} * 0.3578$$

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

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

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

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

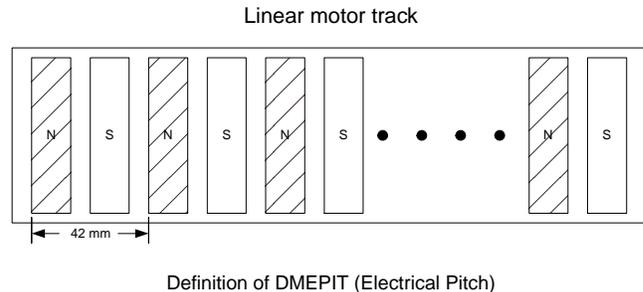
## DMEPIT Motor Electrical Pitch

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

## 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)	$\frac{DMEPIT}{1000}$	$\frac{1000}{DMEPIT}$
DMVLIM (velocity limit)	$\frac{DMEPIT}{1000}$	$\frac{1000}{DMEPIT}$
DMVSCL (velocity scaling)	$\frac{DMEPIT}{1000}$	$\frac{1000}{DMEPIT}$
SMVER (max. allowable velocity error)	$\frac{DMEPIT}{1000}$	$\frac{1000}{DMEPIT}$
TACC (display commanded accel)	$\frac{DMEPIT}{1000}$	$\frac{1000}{DMEPIT}$
TACCA (display actual accel)	$\frac{DMEPIT}{1000}$	$\frac{1000}{DMEPIT}$
TVEL (display commanded velocity)	$\frac{DMEPIT}{1000}$	$\frac{1000}{DMEPIT}$
TVELA (display actual velocity)	$\frac{DMEPIT}{1000}$	$\frac{1000}{DMEPIT}$
TVE (display velocity error)	$\frac{DMEPIT}{1000}$	$\frac{1000}{DMEPIT}$
Velocity & Accel/Decel (e.g., V, A, AD)	$\frac{DMEPIT}{1000}$	$\frac{1000}{DMEPIT}$
DMTD (motor damping)	$\frac{(2 \cdot \pi \cdot 1000)^2}{(DMEPIT)^2}$	$\frac{(DMEPIT)^2}{(2 \cdot \pi \cdot 1000)^2}$
DMTJ (motor/forcer mass)	$\frac{(2 \cdot \pi \cdot 1000)^2}{(DMEPIT)^2}$	$\frac{(DMEPIT)^2}{(2 \cdot \pi \cdot 1000)^2}$
LDAMP (load damping)	$\frac{(2 \cdot \pi \cdot 1000)^2}{(DMEPIT)^2}$	$\frac{(DMEPIT)^2}{(2 \cdot \pi \cdot 1000)^2}$
DMTLIM (force limit)	$\frac{2 \cdot \pi \cdot 1000}{DMEPIT}$	$\frac{DMEPIT}{2 \cdot \pi \cdot 1000}$
DMTSCL (force scaling)	$\frac{2 \cdot \pi \cdot 1000}{DMEPIT}$	$\frac{DMEPIT}{2 \cdot \pi \cdot 1000}$

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

## DMODE Drive Control Mode

Type	Drive Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DMODE<i>	GT	1.02
Units	i = control mode setting	GV	1.00
Range	1-17 (see table below)	GT6	1.50
Default	GT: 6 GV: 2 GT6 & GV6: 12	GV6	1.50
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 ( $\pm 10V$ ) — <b>default for GV drive</b>	Allows direct command of rotary motor torque or linear motor force.	--	<b>X</b>	--	--
3	RESERVED	-----	--	--	--	--
4**	Velocity Control ( $\pm 10V$ )	Allows direct command of the motor velocity.	<b>X</b>	<b>X</b>	--	--
5	RESERVED	-----	--	--	--	--
6	Position Control (step & direction) — <b>default for GT drive</b>	Allows direct command of the motor position.	<b>X</b>	<b>X</b>	--	--
7	Position Control (step & direction, with the direction command inverted)	See option 6 above. Allows for changing the sense of the direction signal.	<b>X</b>	<b>X</b>	--	--
8	Position Control (positive/negative)	Same as option 6, except separate step signals are supplied based on the desired direction.	<b>X</b>	<b>X</b>	--	--
9	Encoder Tracking	Allows a quadrature encoder signal to be used as the position command. The resulting position command is scaled by this ratio: $ERES / DRES$ .	<b>X</b>	<b>X</b>	--	--
10	RESERVED	-----	--	--	--	--
11	Feedback Alignment Mode	(requires a resolver card to be installed) This mode allows you to automatically set the resolver offset angle with the <code>SRSET</code> command.	--	<b>X</b>	--	<b>X</b>
12	Controller/Drive — <b>default for GT6 &amp; GV6 drives</b>	Programmed motion using internal trajectory generator.	--	--	<b>X</b>	<b>X</b>
13	Autorun	Rotates the motor at 1 rps/mps. Current is reduced by 10%. Refer also to your drive's <i>Installation Guide</i> .	<b>X</b>	<b>X</b>	<b>X</b>	<b>X</b>
14	RESERVED	-----	--	--	--	--
15	Torque/Force Tuning Mode	10 Hz step input of 25% rated motor current for torque/force mode tuning. *	--	<b>X</b>	--	<b>X</b>
16	Velocity Tuning Mode	1 Hz $\pm 2$ rps/mps step velocity command for velocity mode tuning. *	--	<b>X</b>	--	<b>X</b>
17	Position Tuning Mode	1 Hz $\pm 1/4$ rev/epitch step position command for position mode tuning. *	--	<b>X</b>	--	<b>X</b>

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

\*\* For  $\pm 10V$  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

Type	Outputs	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DMONAS<i>	GT	1.02
Units	i = scale percentage (%)	GV	1.00
Range	-2000 to 2000	GT6	1.50
Default	100 (no scaling)	GV6	1.50
Response	DMONAS: *DMONAS100		
See Also	DMONAV, DMONBS, DMONBV		

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

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

## DMONAV Analog Monitor Output A — Variable

Type	Outputs	Product	Rev
Syntax	<a_><!>DMONAV<i>	GT	1.02
Units	i = variable number	GV	1.00
Range	0-24 (0 = turn off output)	GT6	1.50
Default	0 (turn off output)	GV6	1.50
Response	DMONAV: *DMONAV0		
See Also	DMEPIT, DMONAS, DMONBS, DMONBV		

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 <i>Hardware Installation Guide</i> for details). Normalized to $\pm 10$ volts equals $\pm 250$ °C.		X		X
2	Drive Temperature	(TDTEMP) Measured internal drive temperature. Normalized to $\pm 10$ volts equals $\pm 250$ °C.	X	X	X	X
3	Position Error	This value is normalized as follows (value clips at $\pm 1$ rev or $\pm 1$ electrical pitch, regardless of the DMONAS setting): <ul style="list-style-type: none"> <li>• Rotary motors: <math>\pm 10V = \pm 1</math> rev (based on TPER + ERES)</li> <li>• Linear motors: <math>\pm 10V = \pm 1</math> epitch (based on TPER + DMEPIT)</li> </ul>		X		X
4	Velocity Setpoint	(TVEL) User commanded velocity. Normalized to $\pm 10$ volts equals $\pm 200$ revs/sec (rotary) or $\pm 8.4$ meters/sec (linear).	X	X	X	X
5	Estimated Actual Velocity	(TVELA) Output of velocity estimator, based on encoder data. Normalized to $\pm 10$ volts equals $\pm 200$ revs/sec (rotary) or $\pm 8.4$ 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 $\pm 10$ volts equals $\pm 200$ revs/sec (rotary) or $\pm 8.4$ 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 $\pm 1$ rev or $\pm 1$ electrical pitch, regardless of the DMONAS setting): <ul style="list-style-type: none"> <li>• Rotary motors: <math>\pm 10V = \pm 1</math> rev (based on TPC + ERES)</li> <li>• Linear motors: <math>\pm 10V = \pm 1</math> epitch (based on TPC + DMEPIT)</li> </ul>		X		X
24	Actual Position	This value is normalized as follows (value clips at $\pm 1$ rev or $\pm 1$ electrical pitch, regardless of the DMONAS setting): <ul style="list-style-type: none"> <li>• Rotary motors: <math>\pm 10V = \pm 1</math> rev (based on TPE + ERES)</li> <li>• Linear motors: <math>\pm 10V = \pm 1</math> epitch (based on TPE + DMEPIT)</li> </ul>		X		X

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

Stepper Drive	Nominal (V/A) Current Scaling	Servo Drive	Nominal (V/A) Current Scaling
GTU5	1.423	GVL3	0.615
GTU8	0.882	GVU3	0.615
GTL5	1.432	GVU6	0.308
GTL8	0.882	GVU12	0.205
		GVH20	0.123

## DMONBS Analog Monitor Output B — Scaling

Type	Outputs	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DMONBS<i>	GT	1.02
Units	i = scale percentage (%)	GV	1.00
Range	-2000 to 2000	GT6	1.50
Default	100 (no scaling)	GV6	1.50
Response	DMONBS: *DMONBS100		
See Also	DMONAS, DMONAV, DMONBV		

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

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

## DMONBV Analog Monitor Output B — Variable

Type	Outputs	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DMONBV<i>	GT	1.02
Units	i = variable number	GV	1.00
Range	0-24 (0 = turn off output)	GT6	1.50
Default	0 (turn off output)	GV6	1.50
Response	DMONBV: *DMONBV0		
See Also	DMONAS, DMONAV, DMONBS		

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 <i>Hardware Installation Guide</i> for details). Normalized to $\pm 10$ volts equals $\pm 250$ °C.		X		X
2	Drive Temperature	(TDTEMP) Measured internal drive temperature. Normalized to $\pm 10$ volts equals $\pm 250$ °C.	X	X	X	X
3	Position Error	This value is normalized as follows (value clips at $\pm 1$ rev or $\pm 1$ electrical pitch, regardless of the DMONAS setting): <ul style="list-style-type: none"> <li>• Rotary motors: <math>\pm 10V = \pm 1</math> rev (based on TPER + ERES)</li> <li>• Linear motors: <math>\pm 10V = \pm 1</math> epitch (based on TPER + DMEPIT)</li> </ul>		X		X
4	Velocity Setpoint	(TVEL) User commanded velocity. Normalized to $\pm 10$ volts equals $\pm 200$ revs/sec (rotary) or $\pm 8.4$ meters/sec (linear).	X	X	X	X
5	Estimated Actual Velocity	(TVELA) Output of velocity estimator, based on encoder data. Normalized to $\pm 10$ volts equals $\pm 200$ revs/sec (rotary) or $\pm 8.4$ 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	( $T_{VE}$ ) Normalized to $\pm 10$ volts equals $\pm 200$ revs/sec (rotary) or $\pm 8.4$ 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 $\pm 1$ rev or $\pm 1$ electrical pitch, regardless of the $DMONAS$ setting): <ul style="list-style-type: none"> <li>Rotary motors: <math>\pm 10V = \pm 1</math> rev (based on <math>TPC + ERES</math>)</li> <li>Linear motors: <math>\pm 10V = \pm 1</math> epitch (based on <math>TPC + DMEPIT</math>)</li> </ul>	X		X	
24	Actual Position	This value is normalized as follows (value clips at $\pm 1$ rev or $\pm 1$ electrical pitch, regardless of the $DMONAS$ setting): <ul style="list-style-type: none"> <li>Rotary motors: <math>\pm 10V = \pm 1</math> rev (based on <math>TPE + ERES</math>)</li> <li>Linear motors: <math>\pm 10V = \pm 1</math> epitch (based on <math>TPE + DMEPIT</math>)</li> </ul>	X		X	

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

Stepper Drive	Nominal (V/A) Current Scaling	Servo Drive	Nominal (V/A) Current Scaling
GTU5	1.423	GVL3	0.615
GTU8	0.882	GVU3	0.615
GTL5	1.432	GVU6	0.308
GTL8	0.882	GVU12	0.205
		GVH20	0.123

## DMTAMB Motor Ambient Temperature

Type	Motor	Product	Rev
Syntax	<a_><!>DMTAMB<r>	GT	n/a
Units	r = Degrees Celsius	GV	1.01
Range	-50.0 to 250.0 : $\pm 0.1$	GT6	n/a
Default	40.0	GV6	1.50
Response	DMTAMB: *DMTAMB40.0		
See Also	DMTMAX, DMTRWC, DMTCM, DMTCW, TASX		

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

---

## DMTD Motor Damping

Type	Motor	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DMTD<r> (does not take effect until RESET or cycle power)	GT	n/a
Units	Rotary motor: r = Nm/rad/sec Linear motor: r = N/meter/sec	GV	1.00
Range	Rotary motor: 0.000000 to 0.010000 : ±0.000001 Linear motor: DMEPIT (electrical pitch) dependent	GT6	n/a
Default	0.000000	GV6	1.50
Response	DMTD: *DMTD0.002000		
See Also	DMTR, LDAMP		

---

**NOTE:** This command does not take effect until you cycle power to the drive, or issue a RESET 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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DMTIC<r>	GT	1.02
Units	r = Amps-RMS	GV	1.00
Range	0.00 to 100.00 : ±0.01	GT6	1.50
Default	0.00 (DMTIC of 0 results in motor configuration warning)	GV6	1.50
Response	DMTIC: *DMTIC6.50		
See Also	DMTICD, TASX, TCS, TDICNT		

---

**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

Type	Motor	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DMTICD<i>    ( <u>does not take effect until RESET or cycle power</u> )	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

Type	Motor	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DMTIND<r>    ( <u>does not take effect until RESET or cycle power</u> )	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, DMMLMX, 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 DMMLMX). 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	Product	Rev
Syntax	<a_><!>DMTIP<r>	GT	n/a
Units	r = Amps-RMS	GV	1.00
Range	0.00 to 128.00 : ±0.01	GT6	n/a
Default	0.00 (DMTIP of 0 results in motor config. warning)	GV6	1.50
Response	DMTIP: *DMTIP7.50		
See Also	DMTIC, DMTICD, DMTLIM, DMTR, TASX, TCS, TDIMAX		

**AUTO-SETUP:** This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6) 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	Product	Rev
Syntax	<a_><!>DMTJ<r> (does not take effect until RESET or cycle power)	GT	1.02
Units	Rotary motor: r = kgm <sup>2</sup> * 10 <sup>-6</sup> Linear motor: r = kg	GV	1.00
Range	Rotary motor: 0.000 to 1000000.000 : ±0.001 Linear motor: DMEPIT (electrical pitch) dependent	GT6	1.50
Default	0.000 (DMTJ of 0 results in motor config. error)	GV6	1.50
Response	DMTJ: *DMTJ200.600		
See Also	DMTR, TASX, TCS		

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

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

Type	System	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DMTLIM<r>	GT	n/a
Units	Rotary motor: r = Nm Linear motor: r = N	GV	1.00
Range	Rotary motor: 0.0 to 500.0 (motor/drive dependent) : ±0.1 Linear motor: DMEPIT (electrical pitch) dependent	GT6	n/a
Default	500.0	GV6	1.50
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):

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

\*  
peak value

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

\*  
peak value

#### 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

Type	Motor	Product	Rev
Syntax	<a_><!>DMTLMN<r> ( <u>does not take effect until RESET or cycle power</u> )	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

Type	Motor	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DMTLMX<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 (DMTLMX of 0 results in motor config. error)	GV6	1.50
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

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

---

**AUTO-SETUP:** This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6) 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 (DMTTTCM), the motor winding time constant (DMTTTCW), the motor thermal resistance (DMTRWC) and the continuous motor current (DMTIC), is used in a real-time estimation of the motor winding temperature. When the winding temperature exceeds DMTMAX, the drive faults and TASX bit #30 is set.

## DMTR Identify (and Load) Motor

Type	Drive Configuration	Product	Rev
Syntax	<a_><!>DMTR<i>	GT	1.02
Units	i = Parker motor identification number	GV	1.00
Range	0-2000	GT6	1.50
Default	-1 (motor config. parameters are not configured)	GV6	1.50
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	Servo Motor Data Parameters	
DIGNA..... Current Loop Gain A	DIBW ..... Current Loop Bandwidth	DMTTCM .... Motor Thermal Time Constant
DIGNB..... Current Loop Gain B	DMEPIT..... Motor Electrical Pitch	DMTTCW .... Motor Winding Time Constant
DIGNC..... Current Loop Gain C	DMTD ..... Motor Damping	DMTW ..... Motor Rated Speed
DIGND..... Current Loop Gain D	DMTIC..... Continuous Current	DMVLIM .... Velocity Limit
DMTIC..... Continuous Current	DMTICD..... Continuous Current Derating	DMVSCL .... Velocity Scaling
DMTIND.... Motor Inductance	DMTIP ..... Peak Current	DPBW ..... Position Loop Bandwidth
DMTJ ..... Motor Rotor Inertia	DMTJ ..... Motor Rotor Inertia	DPOLE ..... Number of Motor Pole Pairs
DMTRES.... Motor Winding Resistance	DMTKE ..... Motor Ke	DRES ..... Drive Resolution
DMTSTT.... Motor Static Torque	DMTLIM..... Torque/Force Limit	DVBW ..... Velocity Loop Bandwidth
DMVLIM.... Velocity Limit	DMTLMN..... Minimum Motor Inductance	ERES ..... Encoder Resolution
DMVSCL.... Velocity Scaling	DMTLMX..... Maximum Motor Inductance	ORES ..... Encoder Output Resolution
DPOLE ..... Number of Motor Pole Pairs	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

Type	Motor	<b>Product</b>	<b>Rev</b>
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

Type	Motor	<b>Product</b>	<b>Rev</b>
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, DMTCM, DMTCW, 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

Type	Drive Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DMTSCL<r>	GT	n/a
Units	Rotary motor: r = Nm Linear motor: r = N	GV	1.00
Range	Rotary motor: 0.0 to 500.0 (motor/drive dependent) : ±0.1 Linear motor: DMEPIT (electrical pitch) dependent	GT6	n/a
Default	0	GV6	1.50
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:

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

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

\* peak value

## DMTSTT Motor Static Torque

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

Type	Motor	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!\>DMTTCM<r> (does not take effect until RESET or cycle power)	GT	n/a
Units	r = minutes	GV	1.00
Range	0.0 to 300.0 : ±0.1	GT6	n/a
Default	0.0	GV6	1.50
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 its final temperature, given constant power. Note that motor mounting will affect this.

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

---

## DMTTCW Motor Winding Time Constant

Type	Motor	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!\>DMTTCW<r> (does not take effect until RESET or cycle power)	GT	n/a
Units	r = minutes	GV	1.00
Range	0.00 to 100.00 : ±0.01	GT6	n/a
Default	0.00	GV6	1.50
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 its final temperature rise above the rest of the motor, given constant power. Note that this is NOT the time constant usually specified in motor data sheets (see DMTTCM); the DMTTCW value is typically much faster.

---

## DMTW Motor Rated Speed

Type	Motor	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DMTW<r> (does not take effect until RESET or cycle power)	GT	n/a
Units	Rotary motor: r = revs/sec Linear motor: r = meters/sec	GV	1.00
Range	Rotary motor: 0.0 to 200.0 : ±0.1 Linear motor: DMEPIT (electrical pitch) dependent	GT6	n/a
Default	0.0 (DMTW of 0 results in motor config. error)	GV6	1.50
Response	DMTW: *DMTW150.0		
See Also	DMEPIT, DMTICD, DMTR, TASX, TCS		

---

**NOTE:** This command does not take effect until you cycle power to the drive, or issue a RESET 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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DMVLIM<r>	GT	1.02
Units	Rotary motor: r = revs/sec Linear motor: r = meters/sec	GV	1.00
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	GT6	1.50
Default	GT/GT6: 50.000000 GV/GV6: 200.000000	GV6	1.50
Response	DMVLIM: *DMVLIM50.000000		
See Also	DCLRLR, DMODE, DMTR, DMVSL, DVBW, SGVRAT, TASX, TGAIN, TSGSET		

---

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

---

<b>DMVSCL</b>		<b>Velocity Scaling</b>		
Type	Drive Configuration		<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DMVSCL<r>		GT	1.02
Units	Rotary motor: r = revs/sec Linear motor: r = meters/sec		GV	1.00
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		GT6	n/a
Default	GT: 50.000000 GV: 0.000000		GV6	n/a
Response	DMVSCL: *DMVSCL40.000000			
See Also	DMODE, DMVLIM, DVBW, LDAMP, SGVRAT			

---

**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:

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

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

---

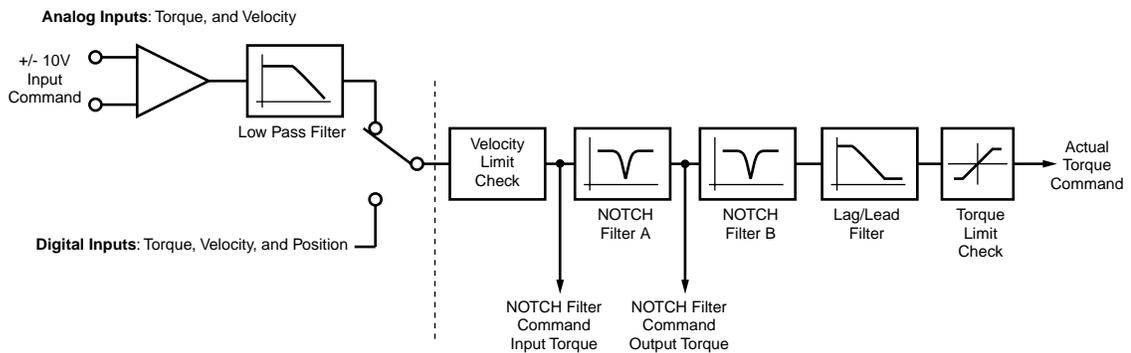
**GV:** Changing the DMVSCL value will affect overall position loop response when used with the drive operating in the ±10V Velocity Mode (DMODE4), so it should ideally be set before tuning.

## DNOTAD Notch Filter A Depth

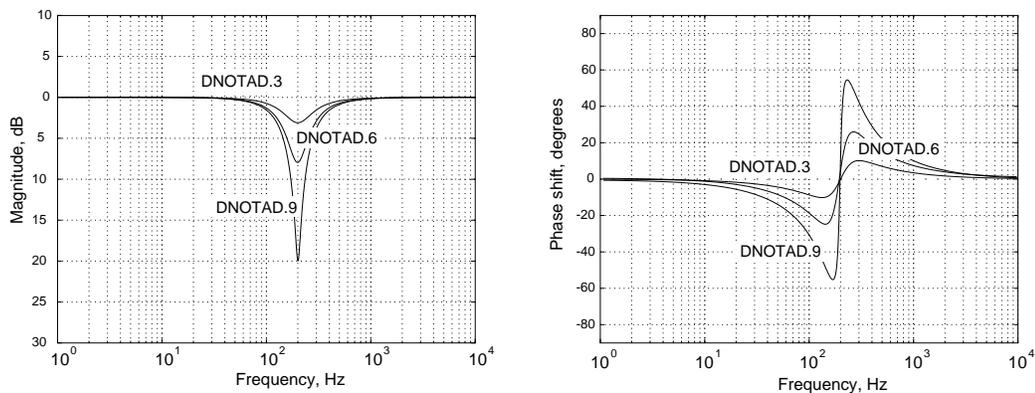
Type	Tuning	Product	GT	Rev	n/a
Syntax	<a_><!>DNOTAD<i>	GV	GT6	GV6	1.01 n/a 1.50
Units	n/a				
Range	0.0000 - 1.0000				
Default	0.0000 (depth is zero)				
Response	DNOTAD: *DNOTAD.5				
See Also	DMODE, DNOTAF, DNOTAQ, DNOTBD, DNOTBF, DNOTBQ, DNOTLD, DNOTLG, DPBW, DVBW, TGAIN, TSGSET				

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.

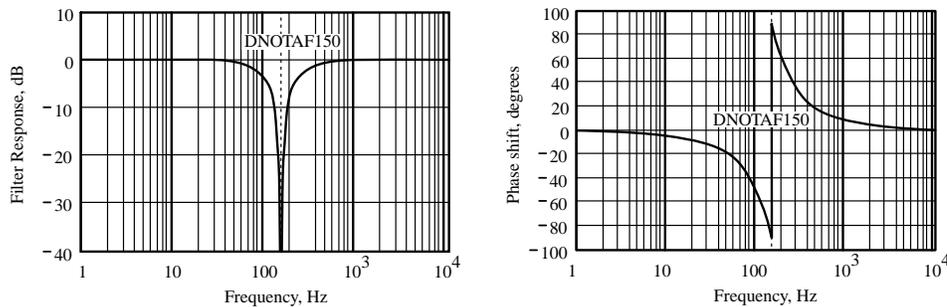
- Servo tuning process: refer to your Gemini drive’s *Hardware Installation Guide*.
- Check the values of all active gains (DNOTAD is one of many servo gains): use TGAIN.
- Creating and invoking gain sets: see SGSET, SGENB, TGAIN, TSGSET.

## DNOTAF Notch Filter A Frequency

Type	Tuning	Product	Rev
Syntax	<a_><!>DNOTAF<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	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

Type	Tuning	Product	Rev
Syntax	<a_><!>DNOTAQ<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	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**

Type	Tuning	<b>Product</b>	<b>Rev</b>
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**

Type	Tuning	<b>Product</b>	<b>Rev</b>
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**

Type	Tuning	<b>Product</b>	<b>Rev</b>
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

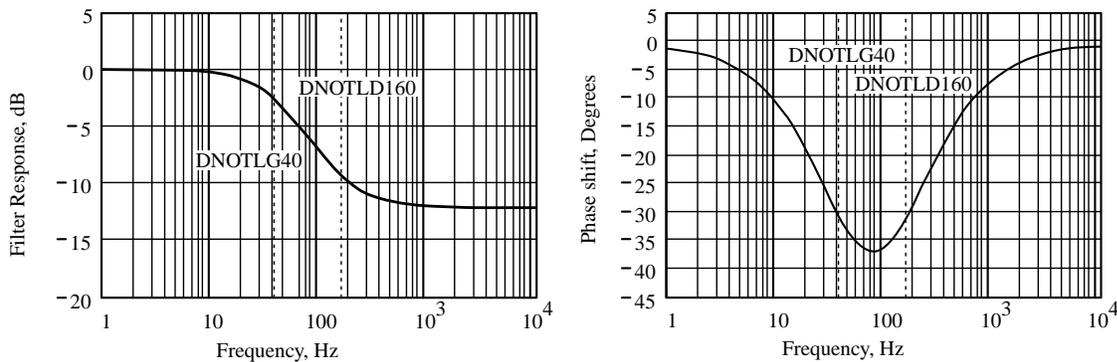
Type	Tuning	Product	Rev
Syntax	<a_><!>DNOTLD<i>	GT	n/a
Units	i = Hz	GV	1.00
Range	0 (disable), or 80-1000	GT6	n/a
Default	0 (filter is disabled)	GV6	1.50
Response	DNOTLD: *DNOTLD200		
See Also	DNOTAD, DNOTAF, DNOTAQ, DNOTBD, DNOTBF, DNOTLG, TASX, TGAIn, TSGSET		

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

Type	Tuning	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DNOTLG<i>	GT	n/a
Units	i = Hz	GV	1.00
Range	0 (disable), or 20-1000	GT6	n/a
Default	0 (filter is disabled)	GV6	1.50
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

Type	Tuning	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DPBW<r>	GT	n/a
Units	r = Hz	GV	1.00
Range	1.00 to 100.00 : ±0.01	GT6	n/a
Default	5.00	GV6	1.50
Response	DPBW:      *DPBW10.00		
See Also	DIBW, DMTR, DVBW, SGPRAT, LJRAT, TGAIN, TSGSET		

---

<p><b>AUTO-SETUP:</b> 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.</p>
---

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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DPHBAL<r>	GT	1.02
Units	r = % of (peak) nominal commanded current	GV	n/a
Range	90.0 to 110.0 : ±0.1	GT6	1.50
Default	100.0	GV6	n/a
Response	DPHBAL: *DPHBAL100.0		
See Also	DPHOFA, DPHOFB, DWAVEF		

---

The DPHBAL command adjusts the current amplitude of phase B with respect to phase A.

A procedure for configuring Phase Balance is provided in the *Configuration* chapter of 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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DPHOFA<r>	GT	1.02
Units	r = no units	GV	n/a
Range	-10.000 to 10.000 : ±0.001	GT6	1.50
Default	0.000	GV6	n/a
Response	DPHOFA: *DPHOFA5.000		
See Also	DPHBAL, DPHOFB, DWAVEF		

---

The DPHOFA command adjusts the current offset of phase A.

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

---

## DPHOFB Phase B Current Offset

Type	Drive Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DPHOFB<r>	GT	1.02
Units	r = no units	GV	n/a
Range	-10.000 to 10.000 : ±0.001	GT6	1.50
Default	0.000	GV6	n/a
Response	DPHOFB: *DPHOFB5.000		
See Also	DPHBAL, DPHOFA, DWAVEF		

---

The DPHOFB command adjusts the current offset of phase B.

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

## DPOLE Number or Motor Pole Pairs

Type	Motor	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DPOLE<i> (does not take effect until RESET or cycle power)	GT	1.02
Units	i = pole pairs	GV	1.00
Range	1-200	GT6	1.50
Default	0 (DPOLE of 0 results in motor config. error)	GV6	1.50
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 ( $\omega_e$ ) is related to the mechanical speed ( $\omega_m$ ) 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

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

Drive	Input Voltage	CONTINUOUS CURRENT				PEAK CURRENT			
		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

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

Type	Drive Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DRIVE<b>	GT	1.02
Units	b = enable bit	GV	1.00
Range	0 (shutdown the drive) or 1 (enable the drive)	GT6	1.50
Default	1 (enabled - if hardware enable interlock is closed)	GV6	1.50
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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DSTALL<i>	GT	1.02
Units	i = sensitivity level selection	GV	n/a
Range	0-50 (0 disables the stall detect function)	GT6	1.50
Default	0 (disabled)	GV6	n/a
Response	DSTALL: *DSTALL20		
See Also	ERROR, ESK, KDRIVE, TAS, TASX, TER		

---

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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DVBW<i>	GT	n/a
Units	i = Hz	GV	1.00
Range	0-500	GT6	n/a
Default	0	GV6	1.50
Response	DVBW: *DVBW200		
See Also	DIBW, DMTR, SGVRAT, TGAIN, TSGSET		

---

**AUTO-SETUP:** This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6) 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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>DWAVEF<r>	GT	1.02
Units	r = % 3 <sup>rd</sup> harmonic	GV	n/a
Range	-20.00 to 10.00 : ±0.01	GT6	1.50
Default	-4.00 (-4% 3 <sup>rd</sup> harmonic injection)	GV6	n/a
Response	DWAVEF *DWAVEF-4		
See Also	DPHBAL, DPHOFA, DPHOFB		

---

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

---

## E Enable Communication

Type	Communication Interface	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>E<b>	GT	1.02
Units	b = enable bit	GV	1.00
Range	0 (serial communication off) or 1 (serial communication on)	GT6	1.50
Default	1	GV6	1.50
Response	n/a		
See Also	ADDR, ECHO, XONOFF		

---

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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>ECHO<b>	GT	1.02
Units	b = enable bit	GV	1.00
Range	0 (disable) or 1 (enable)	GT6	1.50
Default	1	GV6	1.50
Response	ECHO: *ECHO1		
See Also	EOL, EOT, ERRLVL, TSS		

---

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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>ELSE	GT	n/a
Units	n/a	GV	n/a
Range	n/a	GT6	1.50
Default	n/a	GV6	1.50
Response	n/a		
See Also	IF, NIF		

---

ELSE is used in conjunction with the IF and NIF commands to provide conditional branching. If the expression contained within the parentheses of the IF command evaluates true, then the commands between the IF and the ELSE are executed, and the commands after the ELSE until the NIF are ignored. If the expression evaluates false, the commands between the ELSE and the NIF are executed, and the commands between IF and ELSE are ignored. The ELSE command is **optional** and does not have to be included in the IF statement. IF( )...ELSE...NIF commands can be nested up to 16 levels deep.

Programming order: IF(expression) ...commands... ELSE ...commands... NIF

### Example:

```
IF(IN.1=b1) ; Specify condition: if 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

Type	Program Definition	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>END	GT	n/a
Units	n/a	GV	n/a
Range	n/a	GT6	1.50
Default	n/a	GV6	1.50
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

Type	Communication Interface	<b>Product</b>	<b>Rev</b>
Syntax	<!>EOL<i>,<i>,<i>	GT	1.02
Units	n/a	GV	1.00
Range	i = 0-255	GT6	1.50
Default	13,10,0	GV6	1.50
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

Type	Communication Interface	<b>Product</b>	<b>Rev</b>
Syntax	<!>EOT<i>,<i>,<i>	GT	1.02
Units	n/a	GV	1.00
Range	i = 0-255	GT6	1.50
Default	13,0,0	GV6	1.50
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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>ERASE	GT	n/a
Units	n/a	GV	n/a
Range	n/a	GT6	1.70
Default	n/a	GV6	1.70
Response	n/a		
See Also	DEF PROF, DEF PROG, DEL PROF, DEL PROG, RFS		

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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>ERES<i> (does not take effect until RESET or cycle power)	GT	n/a
Units	Rotary motor: i = counts/rev Linear motor: i = counts/electrical pitch	GV	1.00
Range	200-1024000	GT6	n/a
Default	4000 (If SFB4, the default is 4096 counts/rev)	GV6	1.50
Response	ERES: *ERES4000		
See Also	DMEPIT, DMTR, DRES, ORES, SFB, SMPER, TPE, TPER		

**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



## ERRLVL Error Detection Level

Type	Error Handling	Product	Rev
Syntax	<a_><!>ERRLVL<i>	GT	1.02
Units	i = error level setting	GV	1.00
Range	i = 0, 2, 3 or 4	GT6	1.50
Default	4	GV6	1.50
Response	ERRLVL: *ERRLVL4		
See Also	BOT, EOL, EOT, ERBAD, ERROK		

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	<b>Product</b>	<b>Rev</b>
Syntax	<!>ERROK<i>,<i>,<i>,<i>	GT	1.02
Units	n/a	GV	1.00
Range	i = 0-255	GT6	1.50
Default	13,10,62,32	GV6	1.50
Response	ERROK: *ERROK13,10,62,32		
See Also	ERRBAD, ERRLVL		

---

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.



---

## ERRORP Error Program Assignment

Type	Error Handling	Product	Rev
Syntax	<a_><!>ERRORP PROG<i>	GT	n/a
Units	i = number of the program created with DEF PROG	GV	n/a
Range	0, 1-32 (0 clears the existing program assignment)	GT6	1.50
Default	0 (no program is assigned)	GV6	1.50
Response	ERRORP: *ERRORP PROG8		
See Also	DEF PROG, DEL PROG, ERLVL, 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: STRGTE0, D0, 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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>ESK<b>	GT	1.02
Units	b = enable bit	GV	n/a
Range	0 (disable) or 1 (enable)	GT6	1.50
Default	1	GV6	n/a
Response	ESK:        *ESK0		
See Also	DRIVE, DSTALL, TAS, TASX, TER		

---

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 #1 if 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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>FLTDSB<b>	GT	1.02
Units	b = enable bit	GV	1.00
Range	0 (disable) or 1 (enable)	GT6	1.50
Default	1	GV6	1.50
Response	FLTDSB: *FLTDSB1		
See Also	DRIVE, FLTSTP, TAS, TASX, TER		

---

Use the FLTDSB command to enable/disable the Fault on Drive Disable mode. If Fault on Drive Disable is enabled (FLTDSB1 – the default setting), and the drive is disabled via the DRIVE0 command or the “Enable” input (pin 1 on the DRIVE I/O connector), 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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>FLTSTP<b> <i>(does not take effect until RESET or cycle power)</i>	GT	1.02
Units	b = enable bit	GV	1.00
Range	0 (disable) or 1 (enable)	GT6	n/a
Default	1	GV6	n/a
Response	FLTSTP:    *FLTSTP1		
See Also	DMODE, DRIVE, FLTDSB, TAS, TASX, TER		

---

**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`.

<b>GO</b>		<b>Initiate Motion</b>		<b>Product</b>	<b>Rev</b>
Type	Motion			GT	1.02
Syntax	<a_><!>GO<b>			GV	1.00
Units	n/a			GT6	1.50
Range	b = 0 (don't go) or 1 (go)			GV6	1.50
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				

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

Type	Compiled Motion	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>GOBUF<b>	GT	n/a
Units	n/a	GV	n/a
Range	b = 0 (don't go) or 1 (go)	GT6	1.50
Default	1	GV6	1.50
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.

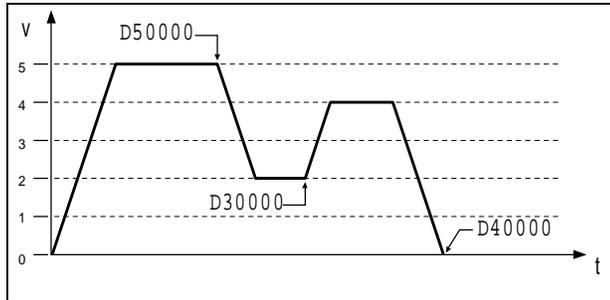
**NOTE:** Each GOBUF motion segment can consume 12 bytes of compiled memory. If there is no more space left in compiled memory, the drive will respond with the ERFBAD prompt (default prompt is "?").

### Example:

```
DEL PROF1 ; Delete profile #1
DEF PROF1 ; Begin definition, profile #1
MCO       ; 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	Product	Rev
Syntax	<a_><!>GOSUB PROG<i>	GT	n/a
Units	i = number of the program created with DEF PROG	GV	n/a
Range	1-32	GT6	1.50
Default	n/a	GV6	1.50
Response	n/a		
See Also	DEF PROG, END, ERROR, ERRORP, JUMP, RUN PROG		

---

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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>GOWHEN(T = <i>)	GT	n/a
Units	i = time in milliseconds	GV	n/a
Range	1 to 999999	GT6	1.50
Default	n/a	GV6	1.50
Response	n/a		
See Also	COMEXC, GOBUF, T, TAS, TER, TRGFN, WAIT		

---

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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>HOM<b>	GT	n/a
Units	n/a	GV	n/a
Range	b = 0 (home in positive direction), or 1 (home in negative direction)	GT6	1.50
Default	n/a	GV6	1.50
Response	n/a		
See Also	HOMA, HOMBAC, HOMDF, HOMEDG, HOMV, HOMVF, HOMZ, INLVL, PSET, TAS, TIN		

---

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).*

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

Pause and resume functions are not recommended during the homing operation. A Pause command or input will pause the homing motion; however, when the subsequent Resume command or input occurs, motion will resume at the beginning of the homing motion sequence.
--

---

The homing operation has several parameters that determine the homing algorithm:

- Home acceleration & 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
HOMO ; Execute go home in positive-direction
END ; End definition of program #20
```

---

## HOMA Home Acceleration

Type	Homing	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>HOMA<r>	GT	n/a
Units	r = revs/sec/sec (linear motors: see page 44 for linear/rotary conversion)	GV	n/a
Range	0.0001 – 9999.9999	GT6	1.50
Default	10.0000	GV6	1.50
Response	HOMA: *HOMA10.0000		
See Also	DMEPIT, HOM, HOMBAC, HOMDF, HOMEDG, HOMV, HOMVF, HOMZ		

---

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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>HOMBAC<b>	GT	n/a
Units	n/a	GV	n/a
Range	b = 0 (disable) or 1 (enable)	GT6	1.50
Default	0	GV6	1.50
Response	HOMBAC: *HOMBAC0		
See Also	HOM, HOMA, HOMDF, HOMEDG, HOMV, HOMVF, HOMZ		

---

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

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

---

## HOMDF Home Final Direction

Type	Homing	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>HOMDF<b>	GT	n/a
Units	n/a	GV	n/a
Range	b = 0 (positive-direction) or 1 (negative-direction)	GT6	1.50
Default	0	GV6	1.50
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.

---

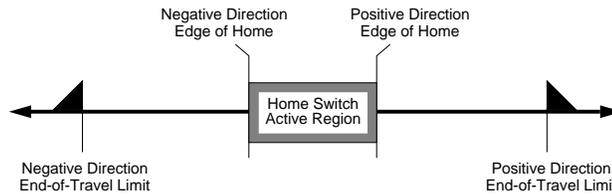
## HOMEDG Home Reference Edge

Type	Homing	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>HOMEDG<b>	GT	n/a
Units	n/a	GV	n/a
Range	b = 0 (positive-direction edge) or 1 (negative-direction edge)	GT6	1.50
Default	0	GV6	1.50
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

Type	Homing	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>HOMV<r>	GT	n/a
Units	r = revs/sec (linear motors: see page 44 for linear/rotary conversion)	GV	n/a
Range	0.0000–200.0000	GT6	1.50
Default	1.0000	GV6	1.50
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

Type	Homing	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>HOMVF<r>	GT	n/a
Units	r = revs/sec (linear motors: see page 44 for linear/rotary conversion)	GV	n/a
Range	0.0000-200.0000	GT6	1.50
Default	0.1000	GV6	1.50
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

Type	Homing	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>HOMZ<b>	GT	n/a
Units	n/a	GV	n/a
Range	b = 0 (disable) or 1 (enable)	GT6	n/a
Default	0	GV6	1.50
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

Type	Program Flow Control; Conditional Branching	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>IF(expression)	GT	n/a
Units	(see syntax examples below)	GV	n/a
Range	(see syntax examples below)	GT6	1.50
Default	n/a	GV6	1.50
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.

<b>Syntax Example: Binary Data</b>	<i>In this example, the IF condition evaluates true when bit #12 of the Axis Status register is set (binary value is "1").</i>
<p style="text-align: center;">IF (AS = bxxxxxxxxxxx1)</p> <p>Operand for the selected status register. ——— ↑ ↑ ↑</p> <p>Options are:</p> <p>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)</p> <p style="text-align: right;">Binary state.  0 = bit is false, or not set  1 = bit is true, or set  X = ignore bit (mask)</p> <p style="text-align: right;">The bit pattern is numbered 1-n, left to right.  This example evaluates true if bit #12 is set.</p> <p style="text-align: right;">"b" is required to prefix the binary state.  "=" is required.</p> <p style="text-align: center;"><u>Syntax Example, using the binary bit-select shortcut:</u></p> <p style="text-align: center;">IF (AS.12 = b0)</p> <p>AS = Axis Status register ——— ↑ ↑ ↑</p> <p>Bit select operator (.) is required ——— ↑ ↑</p> <p>Bit #12 is selected ——— ↑ ↑</p> <p style="text-align: right;">Bit state (0 = false, 1 = true)  "b" is required to prefix the binary state  "=" is required</p>	

<b>Syntax Example: Integer/Variable Data Comparison</b> <i>(This capability is available as of OS rev 1.60)</i>	<i>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.</i>
<p style="text-align: center;">IF (PE &lt; VARI5)</p> <p>Operand options (left &amp; right): ——— ↑ ↑ ↑</p> <ul style="list-style-type: none"> <li>• Numeric constant</li> <li>• Integer variable (VARI<sub>n</sub>)</li> <li>• System variables options (e.g., VARI5=PC): <ul style="list-style-type: none"> <li>A ..... Programmed acceleration (see A)</li> <li>AD ..... Programmed deceleration (see AD)</li> <li>V ..... Programmed velocity (see V)</li> <li>D ..... Programmed distance (see D)</li> <li>ANI ..... Analog input (see TANI)</li> <li>PC ..... Commanded position (see TPC)</li> <li>PE ..... Encoder/resolver position (see TPE)</li> <li>PER ..... Position error (see TPER)</li> </ul> </li> </ul> <p>Operator options:</p> <ul style="list-style-type: none"> <li>= ..... Equals</li> <li>&lt;&gt; ..... Not equal to</li> <li>&gt; ..... Greater than</li> <li>&lt; ..... Less than</li> <li>&gt;= ..... Greater than or equal to</li> <li>&lt;= ..... Less than or equal to</li> </ul> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p style="text-align: center;"><b>NOTE</b></p> <p>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 &lt;= 5.0000 units/sec, use this syntax: IF (V&lt;=50000).</p> </div>	

**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	Product	Rev
Syntax	<a_><!>INDEB<i>	GT	1.02
Units	i = time in milliseconds (ms)	GV	1.00
Range	i = 2-250	GT6	1.50
Default	50	GV6	1.50
Response	INDEB: *INDEB50		
See Also	INFNC, INLVL, TIN		

---

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	<i>(input not available)</i>	INFNC4-H (Trigger interrupt)
5	35	<i>(input not available)</i>	INFNC5-A (General-purpose)
6	37	<i>(input not available)</i>	INFNC6-A (General-purpose)
7	38	<i>(input not available)</i>	INFNC7-A (General-purpose)
8	39	<i>(input not available)</i>	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	Product	Rev
Syntax	<a_><!>INFNC<i>-<c>	GT	n/a
Units	i = general-purpose input #; c = function identifier letter	GV	n/a
Range	i = 1-8; c = A, B, C, D, E, F, H, R, S, T	GT6	1.50
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)	GV6	1.50
Response	INFNC6: *INFNC6-A NO FUNCTION - STATUS OFF <i>(NOTE: input # must be specified for report back)</i>		
See Also	COMEXR, COMEXS, ERROR, INDEB, INLVL, INSELP, INUFD, K, LH, PSET, TER, TIN, TRGFN, TRGLOT, TSS		

---

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., ERRORxxxxxxx1), 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.

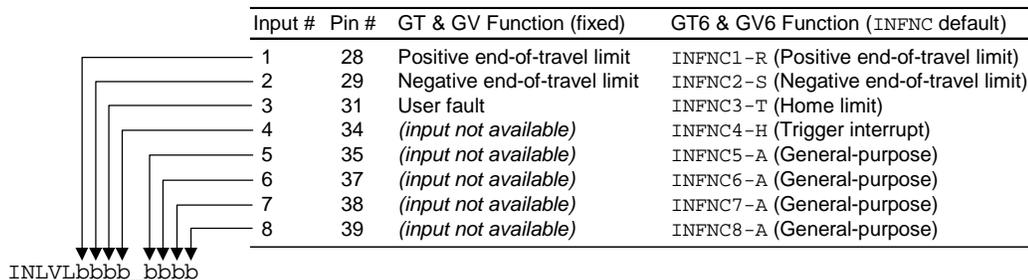
**INFNCi-T Home Limit**

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		Product	Rev
Type	Input			GT	1.02
Syntax	<a_><!>INLVL<bbbbbbbb>			GV	1.00
Units	n/a			GT6	1.50
Range	b = 0 (active low) or 1 (active high)			GV6	1.50
Default	11000000				
Response	INLVL: *INLVL1100_0000				
See Also	INDEB, INFNC, TIN				

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):



**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	Product	Rev
Syntax	<a_><!>INSELP<b>,<i>	GT	n/a
Units	b = strobe enable/disable bit i = strobe time in milliseconds	GV	n/a
Range	b = 0 (disable) or 1 (enable) i = 0-5000	GT6	1.50
Default	0,0	GV6	1.50
Response	INSELP: *INSELP1,40		
See Also	COMEXS, DEF PROG, INFNC, INLVL, TSS		

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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>INUFD<i>	GT	1.61
Units	milliseconds	GV	1.61
Range	0 - 1000	GT6	1.70
Default	0	GV6	1.70
Response	INUFD: *INUFD100		
See Also	INFNC, OUTBD, OUTFNC		

---

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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>JUMP PROG<i>	GT	n/a
Units	i = program number	GV	n/a
Range	1-32	GT6	1.50
Default	n/a	GV6	1.50
Response	n/a		
See Also	DEF PROG, DEL PROG, END, ERROR, ERRORP, GOSUB, IF, L, RUN PROG		

---

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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>K<b>	GT	n/a
Units	n/a	GV	n/a
Range	b = 0 (don't kill) or 1 (kill)	GT6	1.50
Default	n/a	GV6	1.50
Response	n/a		
See Also	ERROR, ERRORP, GO, KDRIVE, LHAD, LHADA, S, TAS, TER		

---

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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>KDRIVE<b>	GT	n/a
Units	b = enable bit	GV	n/a
Range	0 (disable) or 1 (enable)	GT6	1.50
Default	0	GV6	1.50
Response	KDRIVE: *KDRIVE0		
See Also	DRIVE, INFNC, K		

---

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 (∅, 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
```

---

<b>L</b>	<b>Loop</b>	<b>Product</b>	<b>Rev</b>
Type	Loops; Program Flow Control		
Syntax	<a_><!>L<i>	GT	n/a
Units	i = number of times to loop	GV	n/a
Range	0-999,999,999 (o = infinite loop)	GT6	1.50
Default	0	GV6	1.50
Response	L: No response; instead, this has the same function as L0		
See Also	C, COMEXS, GOSUB, LN, PLOOP, PS, S, VARI		

---

When you combine the Loop (L) command with the end of loop (LN) command, all of the commands between L and LN will be repeated the number of times specified by L<i>. If <i> =  $\emptyset$ , or if no argument is specified, all the commands between L and LN will be repeated indefinitely. The loop can be stopped by issuing 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

```

---

<b>LDAMP</b>	<b>Load Damping</b>	<b>Product</b>	<b>Rev</b>
Type	System		
Syntax	<a_><!>LDAMP<r>	GT	n/a
Units	Rotary motor: r = Nm/rad/sec Linear motor: r = N/meter/sec	GV	1.00
Range	Rotary motor: 0.0000 to 1.0000 : $\pm 0.0001$ Linear motor: DMEPIT (electrical pitch) dependent	GT6	n/a
Default	0.0000	GV6	1.50
Response	LDAMP: *LDAMP0.2000		
See Also	DMTD, SGPRAT, SGVRAT, TGAIN, TSGSET		

---

The LDAMP command specifies the damping provided by the mechanical load only, not including the motor itself (which is specified by DMTD).

**Working with servo gains.**

- Servo tuning process: refer to 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

Type	EOT Limit; Drive Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>LH<i>	GT	1.02
Units	n/a	GV	1.00
Range	i = 0 (disable both), 1 (disable positive-direction), 2 (disable negative-direction), or 3 (enable both)	GT6	1.50
Default	GT and GV: 0 (both are disabled) GT6 and GV6: 3 (both are enabled)	GV6	1.50
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.

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

- 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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>LHAD<r>	GT	n/a
Units	r = revs/sec/sec (linear motors: see page 44 for linear/rotary conversion)	GV	n/a
Range	0.0001 - 9999.9999	GT6	1.50
Default	100.0000	GV6	1.50
Response	LHAD: *LHAD100.0000		
See Also	AD, DMEPIT, DRES, INFNC, INLVL, K, KDRIVE, LH, LHADA, LSAD		

---

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)	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>LHADA<r>	GT	n/a
Units	r = revs/sec/sec (linear motors: see page 44 for linear/rotary conversion)	GV	n/a
Range	0.0001 - 9999.9999	GT6	1.50
Default	100.0000 (default is a constant deceleration ramp, where LHADA tracks LHAD; to restore tracking, set AA = 0)	GV6	1.50
Response	LHADA: *LHADA100.0000		
See Also	AA, ADA, DMEPIT, INFNC, INLVL, K, LHAD, LSADA		

---

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	Product	Rev
Syntax	<a_><!>LJRAT<r>	GT	1.02
Units	Rotary motor: r = ratio of load inertia to motor rotor inertia Linear motor: r = ratio of load mass to forcer mass	GV	1.00
Range	0.0 - 100.0	GT6	1.50
Default	0.0	GV6	1.50
Response	LJRAT: *LJRAT4		
See Also	DACTDP, DELVIS, TGAIN, TSGSET		

---

The LJRAT command sets the system's load-to-rotor inertia ratio (rotary motors) or load-to-forcer mass ratio (linear motors). The ratio is expressed in the following equation:

### Rotary Motors:

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	Product	Rev
Syntax	<a_><!>LN	GT	n/a
Units	n/a	GV	n/a
Range	n/a	GT6	1.50
Default	n/a	GV6	1.50
Response	n/a		
See Also	GOSUB, JUMP, L, PLN		

---

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

		Product	Rev
Type	EOT Limit	GT	n/a
Syntax	<a_><!>LS<i>	GV	n/a
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	0		
Response	LS: *LS0		
See Also	LH, LSAD, LSADA, LSNEG, LSPOS, PSET, TAS, TER		

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

		Product	Rev
Type	EOT Limit; Motion	GT	n/a
Syntax	<a_><!>LSAD<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		
Response	LSAD: *LSAD100.0000		
See Also	DMEPIT, DRES, LHAD, LS, LSADA, LSNEG, LSPOS		

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

Type	EOT Limit; Motion (S-Curve)	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>LSADA<r>	GT	n/a
Units	r = revs/sec/sec (linear motors: see page 44 for linear/rotary conversion)	GV	n/a
Range	0.0001 – 9999.9999	GT6	1.50
Default	100.0000 (default is a constant deceleration ramp, where LSADA tracks LSAD; to restore tracking, set AA = 0)	GV6	1.50
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

Type	EOT Limit	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>LSNEG<r>	GT	n/a
Units	r = distance (counts)	GV	n/a
Range	-2,147,483,648 to +2,147,483,647	GT6	1.50
Default	+0	GV6	1.50
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

Type	EOT Limit	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>LSPOS<r>	GT	n/a
Units	r = distance (counts)	GV	n/a
Range	-2,147,483,648 to +2,147,483,647	GT6	1.50
Default	+0	GV6	1.50
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	Product	Rev
Syntax	<a_><!>MA<b>	GT	n/a
Units	n/a	GV	n/a
Range	b = 0 (incremental mode) or 1 (absolute mode)	GT6	1.50
Default	0	GV6	1.50
Response	MA: *MA0		
See Also	COMEXC, D, GO, GOBUF, PSET		

---

The MA command specifies whether the subsequent motion will be made with respect to current position (incremental mode) or with respect to an absolute zero position (absolute mode).

In incremental mode (MA0), all moves are made with respect to the position at the beginning of the move. This mode is useful for repeating moves of the same distance.

In absolute mode (MA1), all moves are made with respect to the absolute zero position. The absolute zero position is equal to zero upon power up and 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	Product	Rev
Syntax	<a_><!>MC<b>	GT	n/a
Units	n/a	GV	n/a
Range	b = 0 (preset mode) or 1 (continuous mode)	GT6	1.50
Default	0	GV6	1.50
Response	MC: *MC0		
See Also	A, AD, COMEXC, COMEXS, D, GO, GOBUF, K, MA, PSET, S, V		

---

The MC command causes subsequent moves to go a specified distance (MC0), or a specified velocity (MC1).

In the Preset Mode (MC0), all moves will go a specific distance. The actual distance traveled is specified by the D 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.



<b>ORES</b>		<b>Resolution of Step &amp; Direction Output (GT) or Encoder Output (GV)</b>	
Type	Drive Configuration; Outputs	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>ORES<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	1.00
Range	GT: 200-128000; 0 to disable GV: 200-1024000; 0 to disable	GT6	1.50
Default	GT: 0 GV: 4000	GV6	1.50
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.



## OUTFNC Output Function

Type	Outputs	Product	Rev
Syntax	<a_><!>OUTFNC<i>-<c>	GT	n/a
Units	i = output # c = function identifier (letter)	GV	n/a
Range	i = 1-7 c = A-I	GT6	1.50
Default	(see table below)	GV6	1.50
Response	OUTFNC1 *OUTFNC1-A PROGRAMMABLE OUTPUT - STATUS OFF		
See Also	INFNC, OUT, OUTLVL, POUTA, SMPER, TAS, TASK, 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

```

### **OUTFN*C*<sub>i</sub>-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).

### **OUTFN*C*<sub>i</sub>-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.

### **OUTFN*C*<sub>i</sub>-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.

### **OUTFN*C*<sub>i</sub>-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.

### **OUTFN*C*<sub>i</sub>-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.

### **OUTFN*C*<sub>i</sub>-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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>OUTLVL<bbbbbbb>	GT	1.02
Units	n/a	GV	1.00
Range	b = 0 (active low) or 1 (active high)	GT6	1.50
Default	0000000	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

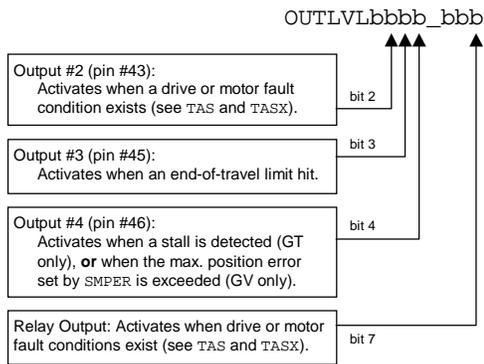
The OUTLVL command defines the active state of Gemini drive’s digital outputs on the DRIVE I/O connector, as well as the relay output (labeled “RELAY COM” and “RELAY N.O.”) on the 4-pin removable connector. The factory default state is active low for the digital outputs, and normally open for the relay 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.

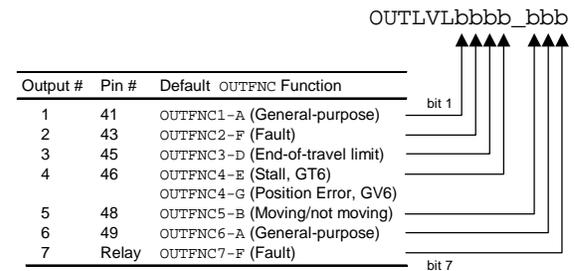
OUTLVL bit assignments (bits are numbered 1-7 from left to right):

### GT & GV:



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).

---

## PLN

### End of Loop, Compiled Motion

Type	Compiled Motion	<b>Product</b>	<b>Rev</b>
Syntax	<a_>PLN	GT	n/a
Units	n/a	GV	n/a
Range	n/a	GT6	1.50
Default	n/a	GV6	1.50
Response	n/a		
See Also	DEF PROF, PLOOP		

---

Use the PLN command to specify the end of a compiled motion profile loop, as initiated with the PLOOP command.

Programming Example: see PLOOP.

---

## PLOOP

### Beginning of Loop, Compiled Motion

Type	Compiled Motion	<b>Product</b>	<b>Rev</b>
Syntax	<a_>PLOOP<i>	GT	n/a
Units	i = number of times to loop	GV	n/a
Range	0-999,999,999 (0 = infinite loop)	GT6	1.50
Default	0	GV6	1.50
Response	n/a		
See Also	DEF PROF, GOBUF, PLN, PRUN PROF		

---

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)

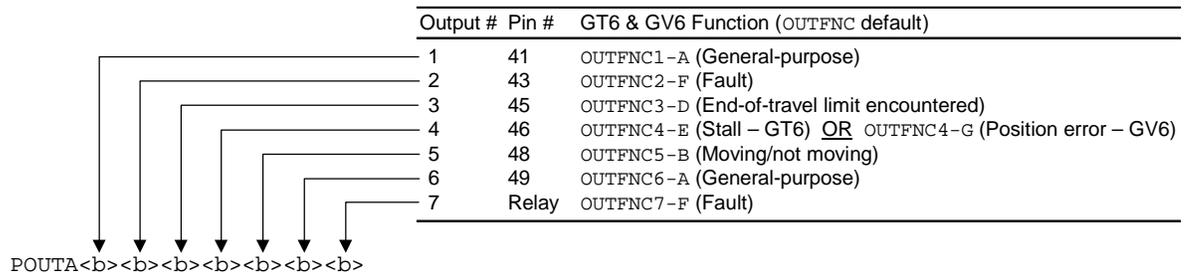
END          ; End definition of profile #1

PRUN PROF1   ; Execute compiled profile #1
```

## POUTA Compiled Output

Type	Compiled Motion; Outputs	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>POUTA<b><b><b><b><b><b><b>	GT	n/a
	<a_><!>POUTA.<i>-<b>	GV	n/a
Units	b = enable bit; i = output number	GT6	1.50
Range	b = 0 (off), 1 (on), or X (don't change); i = 1-7	GV6	1.50
Default	0		
Response	n/a		
See Also	DEF PROF, GOBUF, OUT, OUTFNC, OUTLVL, PRUN PROF		

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.



**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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>PRUN PROF<i> <a_><!>PROF<i>	GT	n/a
Units	i = profile ID number	GV	n/a
Range	1-16	GT6	1.50
Default	n/a	GV6	1.50
Response	n/a		
See Also	COMEXC, DEF PROF, DEL PROF, END, GOBUF, TDIR, TMEM, TSS, (Compiled Motion overview on page 49)		

---

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 COMEXC0 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
MCO            ; 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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>PS	GT	n/a
Units	n/a	GV	n/a
Range	n/a	GT6	1.50
Default	n/a	GV6	1.50
Response	n/a		
See Also	C, COMEXR, COMEXS, K, S, [ SS ], TSS		

---

The PS command pauses execution of commands in the command buffer. If a PS command is executed, no commands after the PS will be executed until a !C command is received. However, additional commands may still be placed in the command buffer.

The PS command does not pause motion. In order for motion to be paused, the S and the COMEXS commands should be used.

### Example:

```
PS             ; Stop execution of command buffer until !C command
MA0           ; Select incremental positioning mode
D10000        ; Set distance to 10000 counts
GO1           ; Initiate motion
; *****
; * NOTE:                                           *
; * No commands after the PS command will be executed until a !C *
; * command is received.                             *
; *****
```

---

## PSET Establish Absolute Position

Type	Motion	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>PSET<r>	GT	n/a
Units	r = counts (absolute position)	GV	n/a
Range	-2,147,483,648 to +2,147,483,647	GT6	1.50
Default	n/a	GV6	1.50
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

Type	Registration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>RE<b>	GT	n/a
Units	n/a	GV	n/a
Range	b = 0 (disable), 1 (enable)	GT6	1.50
Default	0	GV6	1.50
Response	RE: *RE0		
See Also	COMEXC, INFNC, REG, REGLD, 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:  $\pm 1$  encoder count
- GT6:  $\pm 50\mu\text{s}$  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 REGLD command specifies what distance an axis must travel before any input assigned as a registration input will be recognized. Refer to the REGLD command description for further details and an application example.

## Registration Move Status & Error Handling

**Axis Status — Bit #28:** This status bit is set when a registration move has been initiated by the registration input. This status bit is cleared with the next 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.

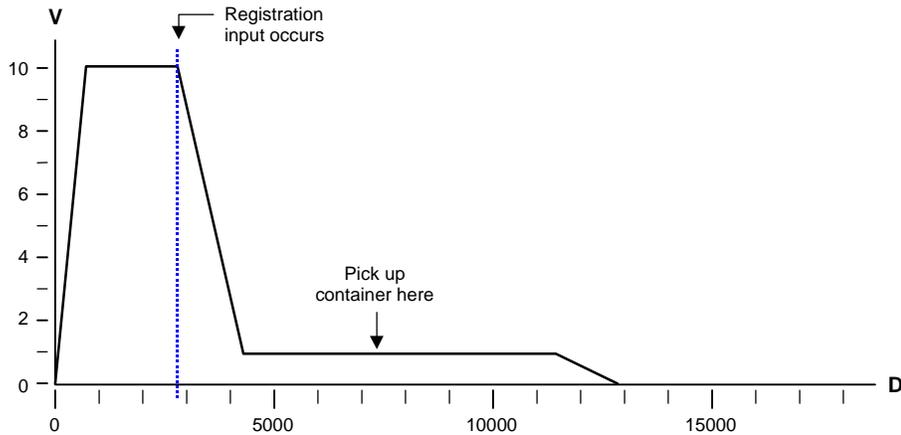
```
DEL PROG19 ; Delete program #19
DEF PROG19 ; Begin program definition
```

```

D50000      ; Set distance to 50000 counts
GO1        ; Initiate motion
END         ; End program definition

```

Registration Profile:



<b>REG</b>		<b>Registration Distance</b>	
Type	Registration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>REG<i>	GT	n/a
Units	i = distance (counts)	GV	n/a
Range	0 to 2,147,483,647 (positive direction only)	GT6	1.50
Default	0 (do not make a registration move)	GV6	1.50
Response	REG: *REG0		
See Also	INFNC, RE, REGLD, TAS, TER, TIN, TRGLOT		

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	Product	Rev
Syntax	<a_><!>REGLOD<i>	GT	n/a
Units	i = distance (counts)	GV	n/a
Range	0 to +2,147,483,647	GT6	1.50
Default	0	GV6	1.50
Response	REGLOD: *REGLOD0		
See Also	INFNC, RE, REG, TRGLOT		

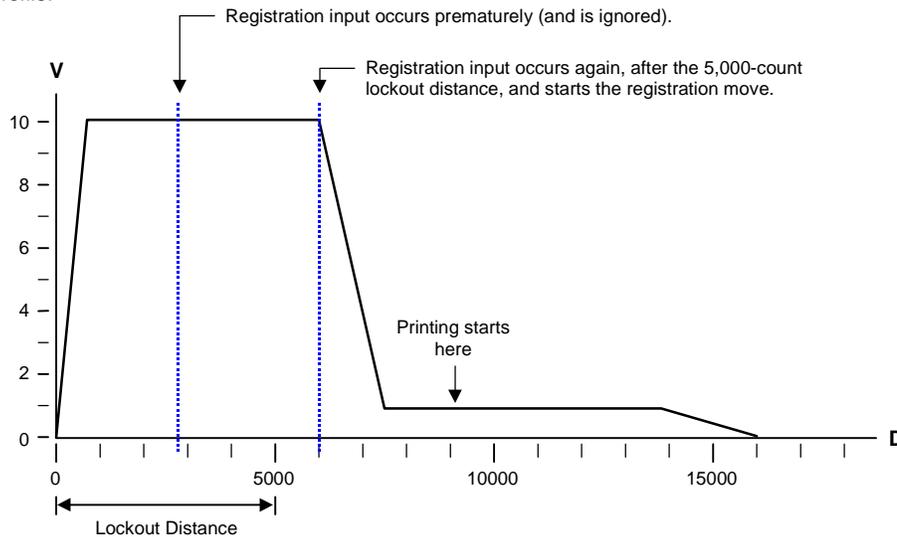
The REGLOD command specifies the distance to travel before a registration input will be recognized. The lock-out distance is measured incrementally from the start of motion to the commanded position (TPC).

### 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

Type	Communication Interface	<b>Product</b>	<b>Rev</b>
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

Type	Drive Configuration	<b>Product</b>	<b>Rev</b>
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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>RUN PROG<i> <a_><!>PROG<i>	GT	n/a
Units	i = program ID number	GV	n/a
Range	1-32	GT6	1.50
Default	n/a	GV6	1.50
Response	n/a		
See Also	DEF PROG, DEL PROG, END, GOSUB, JUMP, PRUN PROF, STARTP, TDIR, TMEM		

---

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	Product	Rev
Syntax	<a_><!>S<b>	GT	n/a
Units	n/a	GV	n/a
Range	b = 0 (do not stop) or 1 (stop)	GT6	1.50
Default	1	GV6	1.50
Response	n/a		
See Also	C, COMEXC, COMEXS, GO, K		

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?		Resume Motion Profile.	Resume Program.	Save Command Buffer.
	Motion	Program	Allow resume with a !C command or a resume input (INFNCi-E).	Allow resume with a !C command or a resume input (INFNCi-E).	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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>SFB<i>	GT	n/a
Units	i = feedback source code	GV	1.60
Range	1 (encoder) or 4 (resolver)	GT6	n/a
Default	1 (encoder)	GV6	1.60
Response	SFB: *SFB1		
See Also	ERES, SRSET, TPE		

---

**AUTO-SETUP:** This command is automatically set according to the Parker motor selected with the configuration utility in Motion Planner (see page 6) 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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>SGAF<i>	GT	n/a
Units	i = %	GV	n/a
Range	0 - 500	GT6	n/a
Default	100	GV6	1.50
Response	SGAF: *SGAF100		
See Also	DMTJ, LJRAT, SGENB, SGSET, SGVF, TGAIN, TSGSET		

---

Use the SGAF command to set the gain for the acceleration feedforward term in the servo control algorithm. Introducing acceleration feedforward control improves *position tracking performance* when the system is commanded to accelerate or decelerate. Acceleration feedforward control does not affect the servo system's stability, nor does it have any effect at constant velocity.

The SGAF value is multiplied by the *commanded acceleration* calculated by the 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

Type	Tuning	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>SGENB<i>	GT	n/a
Units	i = gain set identification number (see SGSET command)	GV	n/a
Range	1-3	GT6	n/a
Default	n/a	GV6	1.50
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

Type	Tuning	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>SGINTE<b>	GT	n/a
Units	b = enable bit	GV	1.00
Range	0 (disable), 1 (enable) or 2 (enable only at end of move)	GT6	n/a
Default	1 (enabled)	GV6	1.50
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

Type	Tuning	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!\>SGIRAT<r>	GT	n/a
Units	r = ratio	GV	1.00
Range	0.500 to 2.000 : ±0.001	GT6	n/a
Default	1.000	GV6	1.50
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

Type	Tuning	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!\>SGPRAT<r>	GT	n/a
Units	r = ratio	GV	1.00
Range	0.500 to 2.000 : ±0.001	GT6	n/a
Default	1.000	GV6	1.50
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

Type	Tuning	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!\>SGPSIG<r>	GT	n/a
Units	r = ratio	GV	1.00
Range	0.100 to 2.000 : ±0.001	GT6	n/a
Default	1.000	GV6	1.50
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.

<b>SGSET</b>		<b>Save a Gain Set</b>	
Type	Tuning	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>SGSET<i>	GT	n/a
Units	i = gain set identification number	GV	n/a
Range	1-3	GT6	n/a
Default	n/a	GV6	1.50
Response	n/a		
See Also	SGENB, TGAIN, TSGSET, (see list below)		

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

Type	Tuning	Product	Rev
Syntax	<a_><!>SGVF<i>	GT	n/a
Units	i = %	GV	n/a
Range	0-500	GT6	n/a
Default	100	GV6	1.50
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

Type	Tuning	Product	Rev
Syntax	<a_><!>SGVRAT<r>	GT	n/a
Units	r = ratio	GV	1.00
Range	0.500 to 2.000 : ±0.001	GT6	n/a
Default	1.000	GV6	1.50
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

Type	Drive Configuration	<b>Product</b>	<b>Rev</b>
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
Default	0	GV6	1.50
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

Type	Drive Configuration	<b>Product</b>	<b>Rev</b>
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
Default	4000	GV6	1.50
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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>SMVER<r>	GT	n/a
Units	r = feedback device revs/sec (linear motors: see DMEPIT for linear/rotary conversion)	GV	1.00
Range	0.000000 to 200.000000 : ±0.000001	GT6	n/a
Default	0.000000 (do not monitor velocity error condition)	GV6	1.50
Response	SMVER: *SMVER10.000000		
See Also	DMEPIT, TASX, TVE, TVEL, TVELA		

---

SMVER determines the maximum velocity error allowed before an error condition occurs. The velocity error is the difference between the commanded velocity (TVEL) and estimated actual velocity (TVELA). If the error exceeds this value, a fault will result in which the drive is shut down (DRIVE0) and TASX bit #9 is set. The DRIVE1 command re-enables the drive, clears TASX bit #9, and sets TVEL equal to TVELA.

You can check the actual velocity error with the TVE command.

If the SMVER value is set to zero (SMVER0), the velocity error condition is not monitored, allowing the velocity error to accumulate without causing a fault.

GV only: SMVER command does not apply to GV operating in DMODE2.

---

## SRSET Resolver Offset Angle

Type	Drive Configuration	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>SRSET<r>	GT	n/a
Units	r = angle in degrees	GV	1.60
Range	-180.0 to +180.0 : ±0.1 (no value = use auto align if in DMODE11)	GT6	n/a
Default	0	GV6	1.60
Response	SRSET: (returns the calculated value only if in DMODE11)		
See Also	DMODE, SFB, TSROFF		

---

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

Type	Program Definition	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>STARTP PROG<i>	GT	n/a
Units	i = program ID #	GV	n/a
Range	0, 1-32 (0 clears the existing program assignment)	GT6	1.50
Default	0 (no program is assigned)	GV6	1.50
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

Type	Target Zone	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>STRGTD<i>	GT	n/a
Units	i = distance (counts)	GV	n/a
Range	0-999,999,999	GT6	n/a
Default	50	GV6	1.50
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

Type	Target Zone	Product	Rev
Syntax	<a_><!>STRGTE<b>	GT	n/a
Units	n/a	GV	n/a
Range	b = 0 (disable) or 1 (enable)	GT6	n/a
Default	0	GV6	1.50
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

Type	Target Zone	Product	Rev
Syntax	<a_><!>STRGTT<i>	GT	n/a
Units	i = milliseconds	GV	n/a
Range	0-5000	GT6	n/a
Default	1000	GV6	1.50
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

Type	Target Zone	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>STRGTV<r>	GT	n/a
Units	r = revs/sec (linear motors: see page 44 for linear/rotary conversion)	GV	n/a
Range	0.0000-200.0000	GT6	n/a
Default	1.0000	GV6	1.50
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)

Type	Program Flow Control	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>T<r>	GT	n/a
Units	r = seconds	GV	n/a
Range	0.001-999.999	GT6	1.50
Default	n/a	GV6	1.50
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

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TACC	GT	n/a
Units	revs/sec/sec (linear motors: see DMEPIT for linear/rotary conversion)	GV	n/a
Range	n/a	GT6	1.50
Default	n/a	GV6	1.50
Response	TACC: *TACC100		
See Also	A, AD, DMEPIT, HOMA, LHAD, LSAD, TACCA, TVEL		

---

TACC reports the commanded acceleration.



Bit # (left to right)	Function (1 = Yes; 0 = No)	G T	GV	G T6	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 <u>not</u> a "fault condition"; instead, it performs a kill (!K).	X	--	X	--	GT: DRIVE1 GT6: GO
13	Drive shut down. <b>NOTE:</b> If operating in the FLTDSB1 mode, a shutdown (DRIVE0 or open the Enable input interlock) will cause a "fault condition" – see note * below.	X	X	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 <u>not</u> a "fault condition".	X	X	X	X	GT/GV: DRIVE1 or LH0. GT6/GV6: LH0 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 <u>not</u> a "fault condition".	X	X	X	X	GT/GV: DRIVE1 or LH0. GT6/GV6: LH0 or GO in opposite direction.
17	Positive-direction Software Limit (LSPOS) Hit.	--	--	X	X	GT6/GV6: LS0 or GO in opposite direction.
18	Negative-direction Software Limit (LSNEG) Hit.	--	--	X	X	GT6/GV6: LS0 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 <i>successful completion</i> of a move (if STRGTD and STRGTV criteria have been satisfied). This bit is usable even if the Target Zone mode is not enabled (STRGTE0).	--	--	--	X	GO1
25	Target Zone Timeout occurred (STRGTT).	--	--	--	X	GO1
26	Change in motion is suspended, pending 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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TASF	GT	1.70
Units	n/a	GV	1.70
Range	n/a	GT6	1.70
Default	n/a	GV6	1.70
Response	TASF: (see example below)		
See Also	TAS, TASX, TASXF, TER, TERF, TSS, TSSF		

---

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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TASX	GT	1.02
Units	n/a	GV	1.00
Range	n/a	GT6	1.50
Default	n/a	GV6	1.50
Response	TASX: *TASX 0000_0000_0000_0000_0000_0000_0000_0000		
See Also	DCLRLR, DRIVE, RESET, TAS, TCS, TER		

TASX reports the axis status conditions. \*TASXbbbb\_bbbb\_bbbb\_bbbb\_bbbb\_bbbb\_bbbb\_bbbb  
↑ Bit 1
Bit 32 ↑

Bit #	Function (1 = Yes; 0 = 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). <b>NOTE:</b> 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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TASXF	GT	1.70
Units	n/a	GV	1.70
Range	n/a	GT6	1.70
Default	n/a	GV6	1.70
Response	TASXF: (see example below)		
See Also	TAS, TASF, TASX, TER, TERF, TSS, TSSF		

---

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 Temperature 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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TCS	GT	1.02
Units	Fault/Warning code (see table below)	GV	1.00
Range	n/a	GT6	1.50
Default	n/a	GV6	1.50
Response	TCS: *TCS46		
See Also	DMTR, TASX		

---

TASX bit #7 is set when a motor configuration error occurs. TASX bit #28 is set when a motor configuration warning occurs. An error causes the drive to be shut down (DRIVE0). A warning means the drive attempted to control the motor outside of safe operating limits (in this case, the maximum safe configuration value is used). To help ascertain the cause of the error or warning, the TCS command reports any existing configuration error or warning conditions (refer to the 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 $\neq$ 4096 during SFB4	Change ERES to 4096 (ERES4096).	Bit 7
-32238	Fault	GV/GV6	Internal position loop gains $\leq$ 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 $\leq$ 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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TDHRS	GT	1.02
Units	Lifetime operating hours (resolution is ¼ or 0.25 hours)	GV	1.00
Range	Hour counter rolls over at 16384.00 hours	GT6	1.50
Default	n/a	GV6	1.50
Response	TDHRS:      *TDHRS16.5		
See Also	RESET, TERRLG		

---

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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TDICNT	GT	1.02
Units	Amps peak	GV	1.00
Range	n/a	GT6	1.50
Default	n/a	GV6	1.50
Response	TDICNT: *TDICNT10		
See Also	DMTIC, DMTIP, TDIMAX		

---

The TDICNT command reports the continuous current rating of the drive in amps peak. Note that most other current-related parameters (e.g., DMTIC, DMTIP) are in amps RMS.

---

## TDIMAX      Transfer Maximum Current Rating

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TDIMAX	GT	n/a
Units	Amps peak	GV	1.00
Range	n/a	GT6	n/a
Default	n/a	GV6	1.50
Response	TDIMAX: *TDIMAX10		
See Also	DMTIP, DMTIP, TDICNT		

---

The TDIMAX command reports the maximum current rating of the drive in amps peak.

Note that most other current-related parameters (e.g., DMTIC, DMTIP) are in amps RMS.

---

## TDIR      Transfer Programs/Profiles Stored in Memory

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TDIR	GT	n/a
Units	n/a	GV	n/a
Range	n/a	GT6	1.50
Default	n/a	GV6	1.50
Response	TDIR:      *PROG1		
See Also	DEF PROF, DEF PROG		

---

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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TDTEMP	GT	1.02
Units	Degrees C	GV	1.00
Range	n/a	GT6	1.50
Default	n/a	GV6	1.50
Response	TDTEMP:    *TDTEMP50		
See Also	DMONAV, DMONBV, TERRLG		

---

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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TDVBUS	GT	1.02
Units	Volts	GV	1.00
Range	n/a	GT6	1.50
Default	n/a	GV6	1.50
Response	TDVBUS:    *TDVBUS170		
See Also			

---

**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.

$$169V (= \sqrt{2} * 120) \quad \text{or} \quad 339V (= \sqrt{2} * 240)$$

## TER Transfer Error Status

Type	Transfer	Product	Rev
Syntax	<a_><!>TER	GT	1.02
Units	n/a	GV	1.00
Range	n/a	GT6	1.50
Default	n/a	GV6	1.50
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; 0 = 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 LH0. GT6/GV6: LH0 or GO in opposite direction.
3	Software end-of-travel limit hit. This condition is detectable only when the limits are enabled (LS3).	--	--	X	X	GT6/GV6: LS0 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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TERF	GT	1.70
Units	n/a	GV	1.70
Range	n/a	GT6	1.70
Default	n/a	GV6	1.70
Response	TERF: (see example below)		
See Also	TAS, TASF, TASX, TASXF, TER, TSS, TSSF		

---

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

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TERRLG	GT	1.02
Units	n/a	GV	1.00
Range	n/a	GT6	1.50
Default	n/a	GV6	1.50
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

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TGAIN	GT	n/a
Units	n/a	GV	1.03
Range	n/a	GT6	n/a
Default	n/a	GV6	1.50
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)
- `SGVFF` (velocity feedforward gain)
- `SGVRAT` (velocity damping ratio)

## THALL Transfer Hall Sensor Values

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>THALL	GT	n/a
Units	n/a	GV	1.50
Range	1-6 (0 or 7 is a fault condition - see TASX bit #21)	GT6	n/a
Default	n/a	GV6	1.50
Response	THALL: *THALL6		
See Also	SHALL, TASX		

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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TIN	GT	1.02
Units	n/a	GV	1.00
Range	n/a	GT6	1.50
Default	n/a	GV6	1.50
Response	TIN: *TIN0000_0000		
See Also	INDEB, INFNC, INLVL, TINO		

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)

\*TINbbbb\_bbbb

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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TINO	GT	1.02
Units	(bit #6 = ENABLE input. 1 = OK for motion)	GV	1.00
Range	n/a	GT6	1.50
Default	n/a	GV6	1.50
Response	TINO: *TINO0000_0100		
See Also	TIN		

---

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.

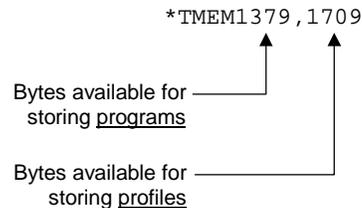
---

## TMEM Transfer Memory Usage

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TMEM	GT	n/a
Units	n/a	GV	n/a
Range	n/a	GT6	1.50
Default	n/a	GV6	1.50
Response	TMEM: *TMEM2996,1598		
See Also	DEF PROG, DEF PROF, TDIR, TPROG		

---

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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TMTEMP	GT	n/a
Units	Degrees C	GV	1.00
Range	n/a	GT6	n/a
Default	n/a	GV6	1.50
Response	TMTEMP: *TMTEMP45		
See Also	DMONAV, DMONBV, DMTRWC, DMTCM, DMTCW, TASX, TERRLG		

---

The TMTEMP reports the predicted temperature of the motor winding for Parker motors. The temperature is estimated using the winding and motor time constants, the rated continuous current, and the winding thermal resistance. The motor will fault (reported with TASX bit 30) at an estimated winding temperature of DMTMAX, assuming the ambient temperature is DMTAMB.

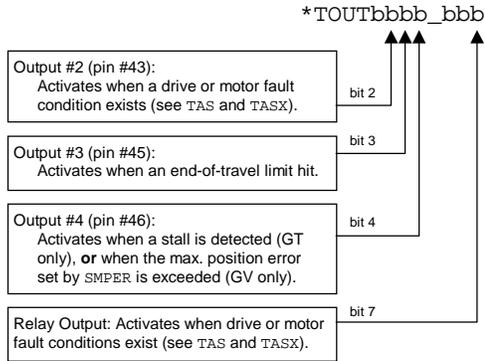
If you are using a non-Parker motor, the TMTEMP value depends on customer-supplied values for the DMTRWC, DMTCM and the DMTCW parameters.

# TOUT Transfer Output Status

Type	Transfer	Product	Rev
Syntax	<a_><!>TOUT	GT	1.02
Units	n/a	GV	1.00
Range	n/a	GT6	1.50
Default	n/a	GV6	1.50
Response	TOUT: *TOUT0000_000		
See Also	OUTFNC, OUTLVL, SMPER, TAS, TASX, TIN, TINO		

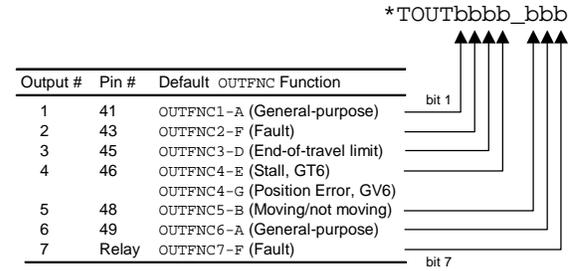
The Transfer Output Status (TOUT) command returns the present status (active or inactive) of the outputs on the DRIVE I/O connector, as well as the dry contact relay output (labeled “RELAY COM” and “RELAY N.O.”) on the 4-pin removable connector. The TOUT response (bits are numbered 1-7 from left to right) 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, OUT:xx1 activates output #3).

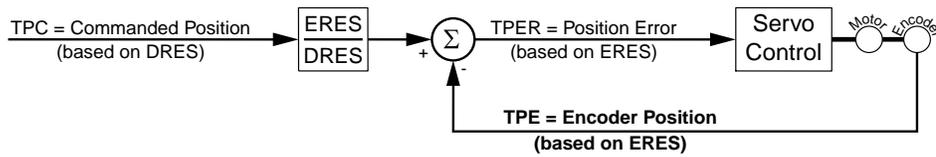
## TPC Transfer Position Commanded

Type	Transfer	Product	Rev
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,  $TPER = TPC * \left(\frac{ERES}{DRES}\right) - TPE$ .

**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  $DRES = ERES$ ):

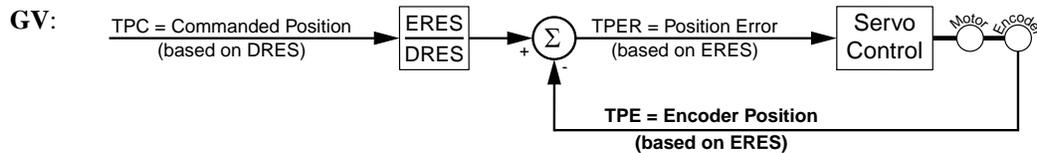
```

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

Type	Transfer	Product	Rev
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).



Position error: If  $DRES = ERES$ , then  $TPER = TPC - TPE$ ; otherwise,  $TPER = TPC * \left(\frac{ERES}{DRES}\right) - TPE$ .

**GV6:** The position error calculation is always:  $TPER = TPC - TPE$ .

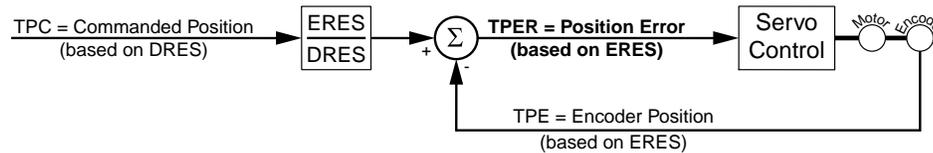
## TPER Transfer Position Error

Type	Transfers	Product	Rev
Syntax	<a_><!>TPER	GT	n/a
Units	Counts	GV	1.00
Range	n/a	GT6	n/a
Default	n/a	GV6	1.50
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  $\mu$ s.

**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,  $TPER = TPC * \left(\frac{ERES}{DRES}\right) - TPE$ .

**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

Type	Transfer	Product	Rev
Syntax	<a_><!>TPRA	GT	n/a
Units	counts	GV	1.61
Range	0 - (ERES-1)	GT6	n/a
Default	n/a	GV6	1.61
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

Type	Transfer	Product	Rev
Syntax	<a_><!>TPROG PROG<i>	GT	n/a
Units	i = program ID number	GV	n/a
Range	1-32	GT6	1.50
Default	n/a	GV6	1.50
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 ERBAD 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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TRACE<b>	GT	n/a
Units	n/a	GV	n/a
Range	b = 0 (disable), 1 (enable)	GT6	1.50
Default	0	GV6	1.50
Response	TRACE: *TRACE0		
See Also	TSS		

---

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

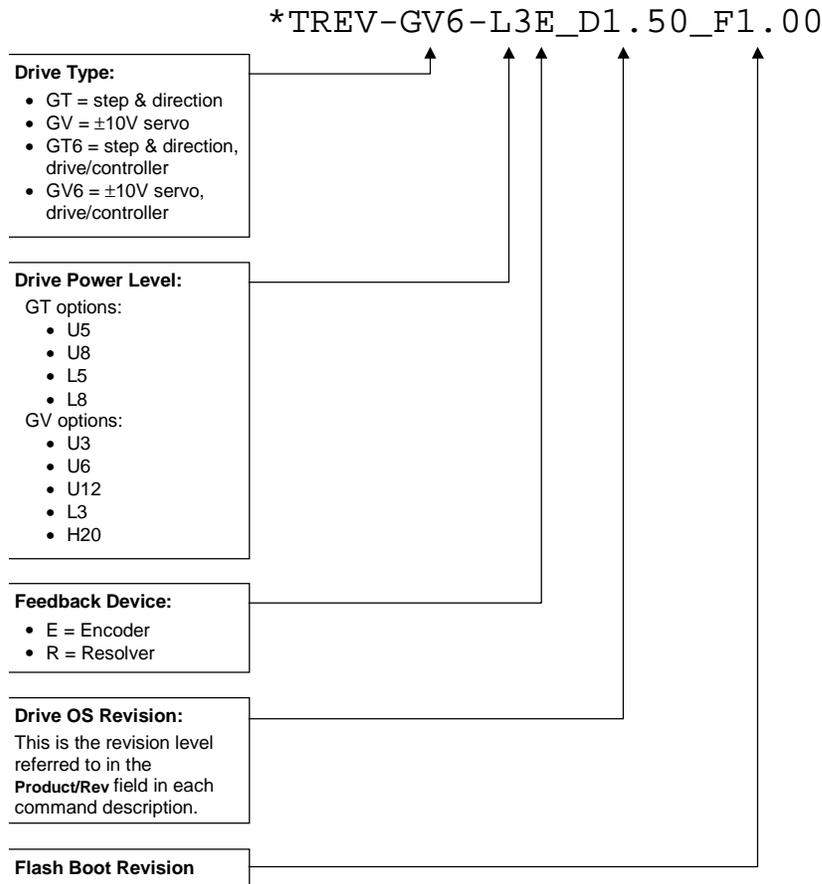
### Transfer Revision Level

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TREV	GT	1.02
Units	n/a	GV	1.00
Range	n/a	GT6	1.50
Default	n/a	GV6	1.50
Response	TREV: *TREV-GV6-L3E_D1.50_F1.00 (response varies by product)		
See Also	RESET, RFS		

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\_###.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

Type	Inputs; Compiled Motion	Product	Rev
Syntax	<a_><!>TRGFN<c><b>	GT	n/a
Units	c = letter corresponding to one of the 8 inputs; b = bit to select Conditional GOBUF function	GV	n/a
Range	c = A-H, corresponding to inputs 1-8, respectively; b = 0 (disable function) or 1 (enable function)	GT6	1.50
Default	c = D (default function for input 4 is "trigger interrupt"); b = 0	GV6	1.50
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).



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 ( <b>trigger interrupt</b> )
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

Type	Inputs	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TRGLOT<i>	GT	n/a
Units	i = time in milliseconds	GV	n/a
Range	0-250	GT6	1.50
Default	24	GV6	1.50
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

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TSGSET<i>	GT	n/a
Units	i = gain set identification number (see SGSET command)	GV	n/a
Range	1-3	GT6	n/a
Default	n/a	GV6	1.50
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	Product	Rev
Syntax	<a_><!>TSROFF	GT	n/a
Units	angle offset, in degrees	GV	1.60
Range	-180.0 to +180.0	GT6	n/a
Default	n/a	GV6	1.60
Response	TSROFF: *TSROFF-90.0		
See Also	DMODE, SRSET		

Use the TSROFF command to ascertain the actual resolver offset angle. SRSET sets the resolver offset angle. When a non-zero value is specified for SRSET, it becomes the new offset angle. When no value is specified, and the drive is in DMODE11 (feedback alignment mode), then a routine is executed to automatically set the resolver angle.

**WARNING:** Motion (less than 1 rev) will occur when you initiate the auto alignment mode.

## TSS Transfer System Status

Type	Transfer	Product	Rev
Syntax	<a_><!>TSS	GT	1.02
Units	n/a	GV	1.00
Range	n/a	GT6	1.50
Default	n/a	GV6	1.50
Response	TSS: *TSS1000_1100_0000_0000_0000_0000_0000_0000		
See Also	TAS		

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, 0 = 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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TSSF	GT	1.70
Units	n/a	GV	1.70
Range	n/a	GT6	1.70
Default	n/a	GV6	1.70
Response	TSSF: (see example below)		
See Also	TAS, TASF, TASX, TASXF, TER, TERF, TSS		

---

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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TSTLT	GT	n/a
Units	Reported value represents milliseconds	GV	n/a
Range	n/a	GT6	n/a
Default	n/a	GV6	1.50
Response	TSTLT: *TSTLT502		
See Also	STRGTD, STRGTE, STRGTT, STRGTV		

---

TSTLT allows you to display the actual time it took the last move to settle into the target zone (that is, within the distance zone defined by STRGTD and less than or equal to the velocity defined by STRGTV). The reported value represents milliseconds. **This command is usable whether or not the Target Zone Settling Mode is enabled with the STRGTE1 command.**

\*\*\* For more information on target zone operation, refer to page [37](#).

---

## TTRQ                      Transfer Commanded Torque/Force

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TTRQ	GT	n/a
Units	% of the DMTSCL value	GV	1.00
Range	-100.00% to +100.00% : ±0.01	GT6	n/a
Default	n/a	GV6	1.50
Response	TTRQ:        *TTRQ50.00		
See Also	DMODE, DMONAV, DMONBV, DMTLIM, DMTSCL, TTRQ		

---

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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TTRQA	GT	n/a
Units	% of full-scale torque/force set by DMTSCL command	GV	1.00
Range	-100.00% to +100.00% : ±0.01	GT6	n/a
Default	n/a	GV6	1.50
Response	TTRQA:       *TTRQA55		
See Also	DMONAV, DMONBV, DMTKE, DMTLIM, DMTSCL, TTRQ		

---

The TTRQA command reports the calculated torque/force, based on q-axis current and the motor's Ke, as a percentage of the full-scale torque/force set by DMTSCL command.

---

## TVE                        Transfer Velocity Error

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TVE	GT	n/a
Units	Revs/sec (linear motors: see DMEPIT for linear/rotary conversion)	GV	1.00
Range	-200 to 200	GT6	n/a
Default	n/a	GV6	1.50
Response	TVE:           *TVE3		
See Also	DMEPIT, DMONAV, DMONBV, DMVLIM, DMVSCL, SMVER, TVEL, TVELA, V		

---

The TVE command reports velocity error in revs/sec (rotary) or meters/sec (linear). The velocity error is the difference between the commanded velocity (TVEL) and estimated actual velocity (TVELA). 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

Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TVEL	GT	1.02
Units	Revs/sec (linear motors: see DMEPIT for linear/rotary conversion)	GV	1.00
Range	-200.000000 to 200.000000 : ±0.000001	GT6	1.50
Default	n/a	GV6	1.50
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

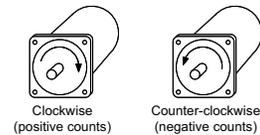
Type	Transfer	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>TVELA	GT	n/a
Units	Revs/sec (linear motors: see DMEPIT for linear/rotary conversion)	GV	1.00
Range	-200.000000 to 200.000000 : ±0.000001	GT6	n/a
Default	n/a	GV6	1.50
Response	TVELA: *TVELA+1.55		
See Also	DMEPIT, DMONAV, DMONBV, DMVLIM, DMVSCL, TPE, 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

Type	Motion	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>V<r>	GT	n/a
Units	revs/sec (linear motors: see page 44 for linear/rotary conversion)	GV	n/a
Range	0.0000 to 200.0000 : ±0.0001	GT6	1.50
Default	1.0000	GV6	1.50
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

Type	Variables	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>VARCLR	GT	n/a
Units	n/a	GV	n/a
Range	n/a	GT6	1.70
Default	n/a	GV6	1.70
Response	n/a		
See Also	VARI		

---

VARCLR resets all variables (VARI) to the default value of zero.

---

## VARI Variable (Integer)

Type	Variables	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>VARI<i><=i>	GT	n/a
Units	1 <sup>st</sup> i = variable number. 2 <sup>nd</sup> i = integer value (assignment).	GV	n/a
Range	1 <sup>st</sup> i = 1-99. 2 <sup>nd</sup> i = -2,147,483,648 to +2,147,483,647.	GT6	1.60
Default	0 (not stored in EEPROM)	GV6	1.60
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<sub>n</sub>, 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<sub>n</sub> = <assignment>

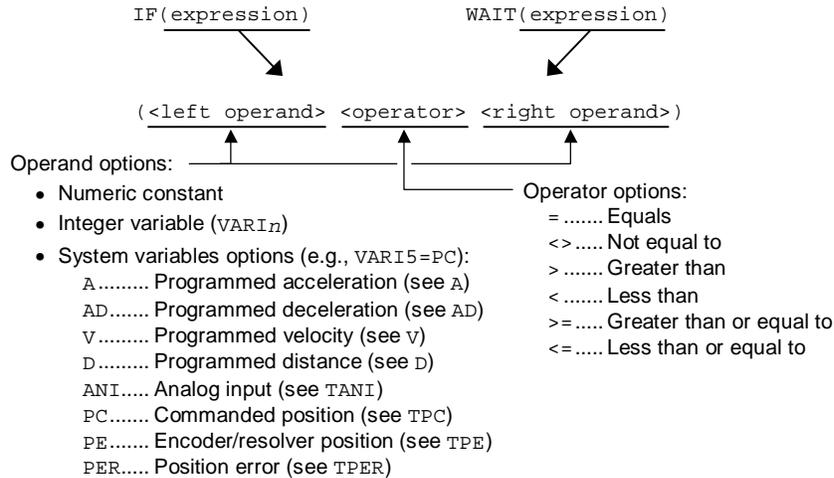
Number of the integer variable.  
Range is 1-99.

“=” 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

Type	Compiled Motion	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>VF<r>	GT	n/a
Units	n/a	GV	n/a
Range	0 (non-zero values result in error message)	GT6	1.50
Default	0	GV6	1.50
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

Type	Program Flow Control	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>WAIT(expression)	GT	n/a
Units	(see syntax examples below)	GV	n/a
Range	(see syntax examples below)	GT6	1.50
Default	n/a	GV6	1.50
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.

<b>Syntax Example: Binary Data</b>	<i>In this example, the WAIT condition evaluates true when bit #12 of the Axis Status register is set (binary value is “1”).</i>
--	--

WAIT (AS = bxxxxxxxxxxx1)

Operand for the selected status register. ———>

Options are:

- AS (axis status – see TAS)
- ASX (extended axis status – see TASX)
- FBS (fieldbus status – see TFBS)
- IN (input status – see TIN)
- 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:

WAIT (AS.12 = b0)

AS = Axis Status register ———>

Bit select operator (.) ———>

Bit #12 is selected ———>

Bit state (0 = false, 1 = true)

“b” is required to prefix the binary state

“=” is required

<p><b>Syntax Example:</b>  <b>Integer/Variable Data Comparison</b>  <i>(This capability is available as of OS rev 1.60)</i></p>	<p><i>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.</i></p>	
<p>WAIT (PE &gt;= 16000)</p>		
<p>Operand options (left &amp; right):</p> <ul style="list-style-type: none"> <li>• Numeric constant</li> <li>• Integer variable (VARI<math>n</math>)</li> <li>• System variables options (e.g., VARI5=PC): <ul style="list-style-type: none"> <li>A..... Programmed acceleration (see A)</li> <li>AD..... Programmed deceleration (see AD)</li> <li>V..... Programmed velocity (see v)</li> <li>D..... Programmed distance (see D)</li> <li>ANI..... Analog input (see TANI)</li> <li>PC..... Commanded position (see TPC)</li> <li>PE..... Encoder/resolver position (see TPE)</li> <li>PER..... Position error (see TPER)</li> </ul> </li> </ul>		<p>Operator options:</p> <ul style="list-style-type: none"> <li>= ..... Equals</li> <li>&lt;&gt; ..... Not equal to</li> <li>&gt; ..... Greater than</li> <li>&lt; ..... Less than</li> <li>&gt;= ..... Greater than or equal to</li> <li>&lt;= ..... Less than or equal to</li> </ul>
<p><b>NOTE</b></p> <p>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 &lt;= 5.0000 units/sec, use this syntax: WAIT(V&lt;=50000).</p>		

**Example:**

```

MCI          ; 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	<b>Product</b>	<b>Rev</b>
Syntax	<a_><!>XONOFF<b>	GT	1.02
Units	b = enable bit	GV	1.01
Range	0 (disable) or 1 (enable)	GT6	1.50
Default	1	GV6	1.50
Response	XONOFF: *XONOFF1		
See Also	BOT, E, EOT, ERBAD, ERROK		

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. XONOFF0 disables XON/XOFF.

**NOTE:** If you are using RS-485 multi-drop or RS-232 daisy-chain, disable XON/XOFF with XONOFF0.