
T**Time Delay**

| | | | |
|----------|--|----------------|------------|
| Type | Program Flow Control | Product | Rev |
| Syntax | <!>T<r> | 6K | 5.0 |
| Units | r = seconds | | |
| Range | 0.001-999.999 | | |
| Default | n/a | | |
| Response | n/a | | |
| See Also | GOWHEN, PS, [SS], [TIM], TTIM, TSS, WAIT | | |

The Time Delay (T) command pauses command processing for **r** seconds before continuing command execution. Once the elapsed time has expired, the command after the T command will be executed.

The minimum resolution of the T command is 2 ms. Although you can enter time delays that are not multiples of 2 ms, the time delay will be rounded up to the next multiple of 2 ms. For example, T.005 produces a 6 ms time delay.

Example:

```
T5           ; Wait 5 seconds before executing TPE command
TPE         ; Transfer position of all encoders to the terminal
```

[TAN()]**Tangent**

| | | | |
|----------|--|----------------|------------|
| Type | Operator (Trigonometric) | Product | Rev |
| Syntax | ... TAN(r) (See below) | 6K | 5.0 |
| Units | r = radians or degrees depending on RADIAN command | | |
| Range | ±17500.0000000 radians | | |
| Default | n/a | | |
| Response | n/a | | |
| See Also | [ATAN], [COS], [PI], RADIAN, [SIN], VAR | | |

The Tangent (TAN) operator is used to calculate the tangent of a number given in radians or degrees (see the RADIAN command). If "a" and "b" are coordinates of a point on a circle of radius "r", then the angle of measure "θ" can be defined by the equation: $\tan \theta = \frac{a}{b}$

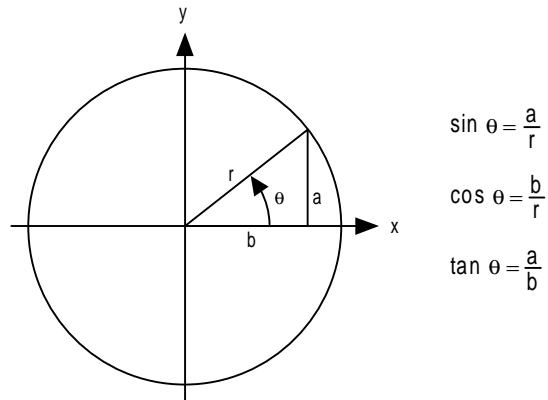
If a value is given in radians and a conversion is needed to degrees, use the following formula:
 $360^\circ = 2\pi$ radians.

Syntax: VARx=TAN(r), where x is the numeric variable number and r is a value in either radians or degrees depending on the RADIAN command.

Parentheses () must be placed around the TAN operand. The result will be specified to 5 decimal places.

Example:

```
VAR1=5 * TAN(PI/4) ; Set variable 1 = 5 times the tangent of Pi divided by 4
```



TANI Transfer Analog Input Voltage



| | | | |
|----------|--|----------------|------------|
| Type | Transfer | Product | Rev |
| Syntax | <!>TANI<.i> | 6K | 5.0 |
| Units | B = I/O brick i = location on I/O brick | | |
| Range | i = 1-32 | | |
| Default | n/a | | |
| Response | 1TANI *1TANIx,x,x,x,x,x,x,x +5.802,-4.663,-4.972, +6.023,+2.126,+2.223, ... x,x,x,x,x,x,x,x x,x,x,x,x,x,x,x 1TANI.10 *-4.663 | | |
| See Also | ANIRNG, [ANI], [FB], [PANI], TFB, TPANI | | |

The Transfer Analog Input Voltage for analog inputs (TANI) command returns the voltage level present at the ANI analog inputs located on external I/O bricks. The value reported with the TANI command is measured in volts and does not reflect the effects of distance scaling (SCLD), position offset (PSET), or commanded direction polarity (CMDDIR). To ascertain the offset ANI input value, as affected by SCLD, PSET, or CMDDIR, use the TPANI command or the TFB command.

To determine the analog value from a specific input, use the bit select operator (.). For example, to check the voltage of the 2nd analog input on the 3rd SIM (I/O location 18) of I/O brick 2, use the 2TANI.18 command. To understand more about the location of I/O points on external I/O bricks, see page 6.

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

TAS Transfer Axis Status

| | | | |
|----------|--|----------------|------------|
| Type | Transfer | Product | Rev |
| Syntax | <!><a>TAS<.i> | 6K | 5.0 |
| Units | i = bit location on the specified axis (See below) | | |
| Range | 1-32 | | |
| Default | n/a | | |
| Response | TAS: *TAS 0000_0000_0000_0000_0000_0000_0000_0000 * 0000_1000_0000_0000_0000_0000_0000_0000 * (repeated for each axis) 1TAS: *1TAS0000_0000_0000_0000_0000_0000_0000_0000 bit 1  bit 32  TAS.5: *00110000 (bit 5 of all eight axes status registers) 1TAS.5: *0 (bit 5 of status register for axis 1) | | |
| See Also | [AS], [ASX], DRFLVL, ESTALL, GOWHEN, HOM, JOG, JOY, MA, MC, SMPER, STRGTD, STRGTE, STRGTT, STRGTV, TASF, TASX, TSTAT | | |

The Transfer Axis Status (TAS) command returns the current status of all axes.

FULL-TEXT STATUS REPORT AVAILABLE

The TAS status command reports a binary bit report. If you would like to see a more descriptive text-based report, use the TASF command description.

| Bit # (left to right) | Function (1/∅) |
|---------------------------------|---|
| 1 | Moving/Not Moving. This bit is set only when motion is <u>commanded</u> on the axis. The motor may still be "moving" (e.g., due to end-of-move settling). |
| 2 | Negative/positive-direction |
| 3 | Accelerating/Not Accelerating. This bit does not indicate deceleration (bit is set to 0 during decel); to check if the axis is decelerating, the state of TAS bits 1, 3 and 4 should be: TAS1x00. |
| 4 | At Velocity/Not at Velocity |
| 5 | Home Successful (HOM) (YES/NO) |
| 6 | Absolute/Incremental (MA) |
| 7 | Continuous/Preset (MC) |
| 8 | Jog Mode/Not Jog Mode (JOG) |
| 9 | Joystick Mode/Not Joystick Mode (JOY) |
| 10 | RESERVED |
| 11 | RESERVED |
| 12 | Stall Detected (YES/NO). This bit is not usable until Stall Detect is enabled with ESTALL1 command. |
| 13 | Drive Shut Down (YES/NO) |
| 14 | Drive Fault occurred (YES/NO). A drive fault cannot be detected (this bit is always 0) until the drive fault input check is enabled with DRFEN1. <u>Note</u> : TASX bit 4 reports the hardware state of the drive fault input, regardless of DRFEN or DRIVE. |
| 15 | Positive-direction Hardware Limit Hit (YES/NO) |
| 16 | Negative-direction Hardware Limit Hit (YES/NO) |
| 17 | Positive-direction Software Limit Hit (YES/NO) |
| 18 | Negative-direction Software Limit Hit (YES/NO) |
| 19 | RESERVED |
| 20 | RESERVED |
| 21 | RESERVED |
| 22 | RESERVED |
| 23 | Position Error Exceeded (SMPER) (YES/NO). Servo axes only. |
| 24 | In Target Zone (defined with STRGTD & STRGTV) (YES/NO). Servo axes only. 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). |
| 25 | Target Zone Timeout occurred (STRGTT) (YES/NO). Servo axes only. |
| 26 | Change in motion is suspended pending GOWHEN (YES/NO). This bit is cleared if the GOWHEN condition is true, or if STOP (!S) or KILL (!K or ^K) is executed. |
| 27 | RESERVED |
| 28 | Registration move initiated by trigger since last GO command. This bit is cleared with the next GO command. |
| 29 | RESERVED |
| 30 | Pre-emptive (OTF) GO or Registration profile not possible |
| 31 | RESERVED |
| 32 | RESERVED |

TASF Transfer Axis Status (full-text report)

| | | | |
|----------|---|----------------|------------|
| Type | Transfer | Product | Rev |
| Syntax | <!><a>TASF | 6K | 5.0 |
| Units | n/a | | |
| Range | n/a | | |
| Default | n/a | | |
| Response | TASF: (see example below) | | |
| See Also | [AS], [ASX], DRFLVL, ESTALL, GOWHEN, HOM, JOG, JOY, MA, MC, SMPER, STRGTD, STRGTE, STRGTT, STRGTV, TAS, TASX, TSTAT | | |

The TASF command returns a text-based status report of all axes. This is an alternative to the binary report (TAS). Example TASF response:

```

*TASF                AXIS #
*                   1   2   3   4   5   6   7   8
*Moving              NO  NO  NO  NO  NO  NO  NO  NO
*Direction NEG      NO  NO  NO  NO  NO  NO  NO  NO
*Accelerating       NO  NO  NO  NO  NO  NO  NO  NO
*At Velocity        NO  NO  NO  NO  NO  NO  NO  NO
*
*Home successful    NO  NO  NO  NO  NO  NO  NO  NO
*Mode Absolute      NO  NO  NO  NO  NO  NO  NO  NO
*Mode Continuous    NO  NO  NO  NO  NO  NO  NO  NO
*Jog Mode           NO  NO  NO  NO  NO  NO  NO  NO
*
*Joystick Mode      NO  NO  NO  NO  NO  NO  NO  NO
*RESERVED           NO  NO  NO  NO  NO  NO  NO  NO
*RESERVED           NO  NO  NO  NO  NO  NO  NO  NO
*Stall Detected     NO  NO  NO  NO  NO  NO  NO  NO
*
*Drive Shutdown     NO  NO  NO  NO  NO  NO  NO  NO
*Drive Faulted      NO  NO  NO  NO  NO  NO  NO  NO
*POS Hard Limit Hit NO  NO  NO  NO  NO  NO  NO  NO
*NEG Hard Limit Hit NO  NO  NO  NO  NO  NO  NO  NO
*
*POS Sftwr Limit Hit NO  NO  NO  NO  NO  NO  NO  NO
*NEG Sftwr Limit Hit NO  NO  NO  NO  NO  NO  NO  NO
*RESERVED           NO  NO  NO  NO  NO  NO  NO  NO
*RESERVED           NO  NO  NO  NO  NO  NO  NO  NO
*
*RESERVED           NO  NO  NO  NO  NO  NO  NO  NO
*RESERVED           NO  NO  NO  NO  NO  NO  NO  NO
*Pos Error Exceeded NO  NO  NO  NO  NO  NO  NO  NO
*In Target Zone     YES YES YES YES YES YES YES YES
*
*Target Zone Timeout NO  NO  NO  NO  NO  NO  NO  NO
*Gowhen is Pending  NO  NO  NO  NO  NO  NO  NO  NO
*RESERVED           NO  NO  NO  NO  NO  NO  NO  NO
*Reg Move Commanded NO  NO  NO  NO  NO  NO  NO  NO
*
*RESERVED           NO  NO  NO  NO  NO  NO  NO  NO
*Preset Move Overshot NO NO NO NO NO NO NO NO

```

[TASK] Task Number Assignment

| | | | |
|----------|---|----------------|------------|
| Type | Assignment or Comparison | Product | Rev |
| Syntax | See below | 6K | 5.0 |
| Units | The TASK value is the number of the controlling task. | | |
| Range | 0-10 | | |
| Default | n/a | | |
| Response | n/a | | |
| See Also | %, TTASK, VAR, VARI | | |

The Task Number Assignment operator (TASK) allows the program itself to determine which task is executing it. The current task number TASK may be assigned to a numeric or integer variable or evaluated in a conditional statement, such as IF or WAIT.

Syntax: VARn=TASK or VARIn=TASK where “n” is the variable number; or TASK can be used in an expression, such as IF(TASK=3).

The TASK operator allows a single program to be used as a subroutine called from programs running in all tasks, yet this routine could contain sections of statements which are executed by some tasks and not others. The example below demonstrates statements used to execute different WAIT-for-input conditions, depending on the task that is executing the program.

Example:

```
IF (TASK=1)      ; Check if this program is operating in task 1
WAIT(1IN.3=B1)  ; If in task 1, wait for input at location 3 on I/O brick 1
NIF

IF(TASK=2)      ; Check if this program is operating in task 2
WAIT(2IN.11=B1) ; If in task 2, wait for input at location 11 on I/O brick 2
NIF
```

| TASX | | Transfer Extended Axis Status | |
|-------------|--|--------------------------------------|------------|
| Type | Transfer | Product | Rev |
| Syntax | <!><a>TASX<.i> | 6K | 5.0 |
| Units | i = bit location on the specified axis (See below) | | |
| Range | 1-32 | | |
| Default | n/a | | |
| Response | TASX: *TASX 0000_0000_0000_0000_0000_0000_0000_0000 * 0000_1000_0000_0000_0000_0000_0000_0000 * (repeated for each axis) 1TASX: *1TASX0000_0000_0000_0000_0000_0000_0000_0000 bit 1 \longleftarrow bit 32 TASX.5: *00110000 (bit 5 of all eight axes status registers) 1TASX.5: *0 (bit 5 of status register for axis 1) | | |
| See Also | [AS], [ASX], [ER], EFAIL, TAS, TASXF, TER | | |

The Transfer Extended Axis Status (TASX) command returns the current status for each axis.

FULL-TEXT STATUS REPORT AVAILABLE

The TASX status command reports a binary bit report. If you would like to see a more descriptive text-based report, use the TASXF command description.

| Bit Assignment | |
|-----------------------|--|
| (left to right) | Function (1 = yes, 0 = no) |
| 1-3 | RESERVED |
| 4 | Drive Fault Input Active (indicates the current hardware state of the drive fault input, <u>even if the drive and the drive fault input are disabled</u>) |
| 5 | Encoder failure (requires EFAIL1 enabled for the axis). This bit is cleared with the EFAIL0 command |
| 6 | Encoder Z-Channel state (1 = active, 0 = inactive) |
| 7-32 | RESERVED |

Bit #4 indicates the current hardware state of the drive fault input, even in the factory default power-up state—the drive is disabled (see DRIVE command) and the drive fault input is disabled (see DRFEN command).

TASXF Transfer Extended Axis Status, (full-text report)

| | | | |
|----------|-----------------------------------|---------|-----|
| Type | Transfer | Product | Rev |
| Syntax | <!><a>TASXF | 6K | 5.0 |
| Units | n/a | | |
| Range | n/a | | |
| Default | n/a | | |
| Response | TASXF: (see example below) | | |
| See Also | [AS], [ASX], [ER], TAS, TASX, TER | | |

The TASXF command returns a text-based status report of all axes. This is an alternative to the binary report (TASX). Example TASXF response:

```
*TASX          AXIS #
*              1   2   3   4   5   6   7   8
*RESERVED      NO NO NO NO NO NO NO NO
*RESERVED      NO NO NO NO NO NO NO NO
*RESERVED      NO NO NO NO NO NO NO NO
*Drive Fault Active NO NO NO NO NO NO NO NO
*
*Encoder Failure NO NO NO NO NO NO NO NO
*Z-Channel Active NO NO NO NO NO NO NO NO
```

TCMDER Transfer Command Error

| | | | |
|----------|--------------------------------|---------|-----|
| Type | Transfer or Program Debug Tool | Product | Rev |
| Syntax | <!>TCMDER | 6K | 5.0 |
| Units | n/a | | |
| Range | n/a | | |
| Default | n/a | | |
| Response | TCMDER: *(incorrect command) | | |
| See Also | ERRBAD, [SS], TSS | | |

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

When the bad command is detected, the controller sends an error message to the screen, followed by the ERRBAD error prompt (?). To determine which command is in error, enter the TCMDER command and the controller will display the command, including all its command fields, if any.

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

Example:

```
DEF badprg          ; Begin definition of program called badprg
MA11                ; Select the absolute preset positioning mode
A25,40              ; Set acceleration
AD11,26             ; Set deceleration
V5,8                ; Set velocity
VAR1=0              ; Set variable #1 equal to zero
GO11                ; Initiate move on both axes
IF(VAR1<)16         ; Mistyped IF statement--should be typed as: IF(VAR1<16)
VAR1=VAR1+1         ; If variable #1 is less than16, increment the counter by 1
NIF                 ; End IF statement
END                 ; End programming of program called badprg
RUN badprg          ; Run the program called badprg
                    ; (this will cause an error --see comment box below)
; *****
; * 1. When you run the badprg program, you should see this error *
; * message on your screen: "**INCORRECT DATA" (this error message *
; * indicates incorrect command syntax). *
; * 2. Type "TCMDER" and press enter. This queries the controller to *
; * display the command that caused the error. In this case, the *
; * response will be "*IF(VAR1<)16". *
; *****
```

TDAC Transfer Digital-to-Analog Converter (DAC) Voltage

| | | | |
|----------|--|----------------|------------|
| Type | Transfer | Product | Rev |
| Syntax | <!><@><a>TDAC | 6K | 5.0 |
| Units | Reported value represents volts | | |
| Range | Range of reported value is -10 to +10 | | |
| Default | n/a | | |
| Response | TDAC: *TDAC10.000,10.000,10.000,10.000 ... 1TDAC: *1TDAC10.000 | | |
| See Also | [DAC], DACLIM, SFB, SGAF, SGI, SGP, SGV, SGVF, SOFFS | | |

This command allows you to display the voltage being commanded at the digital-to-analog converter (DAC). This is the *analog command signal* (plus any voltage offset set with the SOFFS command) output by the servo controller. The DAC output is a 12-bit, $\pm 10V$ analog signal. At any point, the voltage that is currently being commanded can be displayed using the TDAC command. If direct control over the analog voltage is required, it can be accomplished by setting the servo algorithm gains (SGP, SGI, SGV, SGVF, & SGAF) to zero and using the SOFFS command.

Example:

```
TDAC           ; Display the actual output voltage for each axis.  
              ; Example response is: *TDAC4.552,5.552,5.552,5.552
```

TDIR Transfer Program Directory

| | | | |
|----------|--|----------------|------------|
| Type | Transfer | Product | Rev |
| Syntax | <!>TDIR | 6K | 5.0 |
| Units | n/a | | |
| Range | n/a | | |
| Default | n/a | | |
| Response | TDIR: *NO PROGRAMS DEFINED *33000 OF 33000 BYTES (100%) PROGRAM MEMORY REMAINING *500 OF 500 SEGMENTS (100%) COMPILED MEMORY REMAINING | | |
| See Also | DEF, INFNC, LIMFNC, MEMORY, PLCP, [SEG] TMEM, TSEG | | |

The Transfer Program Directory (TDIR) command returns the names of all the programs and subroutines defined with the DEF command, and the amount of memory each consumes. The format of the response is as follows:

```
*1 - PROG1 USES 345 BYTES  
*2 - PROG2 USES 333 BYTES  
*32322 OF 33000 BYTES (98%) PROGRAM MEMORY REMAINING  
*500 OF 500 SEGMENTS (100%) COMPILED MEMORY REMAINING
```

(In the above example, PROG1 and PROG2 are names of programs.)

NOTE: The amount of memory available is product-dependent.

The number in front of the program name is the number to use when defining specific inputs (INFNC) to correspond to a specific program (function P of INFNC or LIMFNC), or when programs are selected via BCD (function B of INFNC or LIMFNC).

If the program is intended to be a compiled profile and has been successfully compiled (PCOMP), then the line item for a compiled contouring or GOBUF program is amended with "COMPILED AS A PATH", and the line item for a compiled PLCP program is "COMPILED AS A PLC PROGRAM."

| | |
|-------|---|
| 5 | RESERVED (refer to the ERROR command) |
| 6 | Kill Input: When an input is defined as a Kill input (INFNCi-C or LIMFNCi-C), and that input becomes active. |
| 7 | User Fault Input: When an input is defined as a User Fault input (INFNCi-F or LIMFNCi-F), and that input becomes active. |
| 8 | Stop Input: When an input is defined as a Stop input (INFNCi-D or LIMFNCi-D), and that input becomes active. |
| 9 | Enable input is activated (not grounded). |
| 10 | Pre-emptive (on-the-fly) GO or registration move profile not possible. |
| 11 ** | Target Zone Settling Timeout Period (set with the STRGTT command) is exceeded. |
| 12 ** | Maximum Position Error (set with the SMPER command) is exceeded. |
| 13 | RESERVED |
| 14 | Position relationship in GOWHEN already true when GO, GOL, FSHFC, or FSHFD was executed. |
| 15 | RESERVED |
| 16 | Bad command detected (bit is cleared with TCMDER command). |
| 17 | Encoder failure (EFAIL1 must be enabled before error can be detected; error is cleared by sending EFAIL0 to the affected axis). |
| 18 | Cable to an expansion I/O brick is disconnected, or power to the I/O brick is lost; to clear the error, reconnect the I/O brick (or restore power to the I/O brick) and issue the ERROR.18-0 command and then the ERROR.18-1 command. |
| 19-32 | RESERVED |

* Stepper axes only; ** Servo axes only

When error bit 5 (Commanded Kill or Stop) of the ERROR command is enabled (ERROR.5-1), a Stop (!S) or a Kill (!K or <ctrl>K) command will cause the controller to GOSUB or GOTO to the error program (ERRORP). Within the error program the cause of the error will need to be determined. The transfer error status (TER) command can be used to determine the cause of the error. If none of the error status bits are set, the cause of the error is a commanded kill or a commanded stop. The reason for not setting a bit on this error condition is that there is no way to clear the error condition upon leaving the error program.

TERF Transfer Error Status (full-text report)

| | | | |
|----------|--|---------|-----|
| Type | Transfer | Product | Rev |
| Syntax | <!><%>TERF | 6K | 5.0 |
| Units | n/a | | |
| Range | n/a | | |
| Default | n/a | | |
| Response | TERF: (see example below) | | |
| See Also | [ASX], DRFLVL, EFAIL, [ER], ERROR, ESTALL, GOWHEN, INFNC, LH, LIMFNC, LS, SMPER, STRGTT, TASX, TCMDER, TER | | |

The TERF command returns a text-based status report of all axes. This is an alternative to the binary report (TER). Example TERF response:

| *TERF | AXIS # | | | | | | | |
|----------------------|--------|----|----|----|----|----|----|----|
| * | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| *Stall Detected | NO | NO | NO | NO | NO | NO | NO | NO |
| *Hard Limit Hit | NO | NO | NO | NO | NO | NO | NO | NO |
| *Soft Limit Hit | NO | NO | NO | NO | NO | NO | NO | NO |
| *Drive Fault Active | NO | NO | NO | NO | NO | NO | NO | NO |
| * | | | | | | | | |
| *RESERVED | NO | NO | NO | NO | NO | NO | NO | NO |
| *Kill Input Active | NO | NO | NO | NO | NO | NO | NO | NO |
| *User Fault Input | NO | NO | NO | NO | NO | NO | NO | NO |
| *Stop Input Active | NO | NO | NO | NO | NO | NO | NO | NO |
| * | | | | | | | | |
| *Enable Input OK | NO | NO | NO | NO | NO | NO | NO | NO |
| *Profile Impossible | NO | NO | NO | NO | NO | NO | NO | NO |
| *Target Zone Timeout | NO | NO | NO | NO | NO | NO | NO | NO |
| *Max Position Error | NO | NO | NO | NO | NO | NO | NO | NO |
| * | | | | | | | | |
| *RESERVED | NO | NO | NO | NO | NO | NO | NO | NO |
| *GOWHEN cond true | NO | NO | NO | NO | NO | NO | NO | NO |
| *RESERVED | NO | NO | NO | NO | NO | NO | NO | NO |
| *Bad command | NO | NO | NO | NO | NO | NO | NO | NO |
| | | | | | | | | |
| *Encoder Failure | NO | NO | NO | NO | NO | NO | NO | NO |
| *I/O Brick Failure | NO | NO | NO | NO | NO | NO | NO | NO |

TEX Transfer Program Execution Status

| | | | |
|----------|------------------------------|----------------|------------|
| Type | Transfer | Product | Rev |
| Syntax | !<%>TEX | 6K | 5.0 |
| Units | n/a | | |
| Range | n/a | | |
| Default | n/a | | |
| Response | !TEX: *PROGRAM NOT EXECUTING | | |
| See Also | DEF | | |

The Transfer Program Execution Status (TEX) command reports the status of any programs in progress (in the specified task). If using multi-tasking, you must prefix the TEX with the task you want to check (e.g., 2%TEX).

If the program PAUL was in progress (in task 0), and within that program a loop was in progress, the response to !TEX could look like the following: *PROGRAM=PAUL COMMAND=LN LOOP COUNT=12

TFB Transfer Position of Selected Feedback Devices

| | | | |
|----------|---|----------------|---------------------------------|
| Type | Transfer | Product | Rev |
| Syntax | <!><a>TFB | 6K | 5.0 |
| Units | Response is position of the selected feedback devices | | |
| Range | n/a | | (applicable only to servo axes) |
| Default | n/a | | |
| Response | TFB *TFB+0,+0,+0,+0,+0,+0,+0,+0 1TFB *1TFB+0 | | |
| See Also | [ANI], CMDDIR, ENCPOL, [FB], [PANI], [PE], PSET, SCALE, SCLD, SFB, TANI, TPANI, TPCE, TPE | | |

Use the TFB command to return the current values of the feedback sources selected with the SFB command. If you do not change the default SFB selection, the response will indicate the encoder position.

If scaling is **not** enabled, the position values returned will be counts (encoder or analog input). If scaling is enabled (SCALE1), the values will be scaled by the SCLD value.

If you issue a PSET command, the feedback device position value will be offset by the PSET command value.

Example:

```
SFB2,1 ; Select ANI feedback on axis 1 and encoder feedback on axis 2
TFB    ; Report ANI input #1's voltage and encoder #2's position.
       ; Sample response is *TFB4.256,2.436
```

TFS Transfer Following Status

| | | | |
|----------|---|----------------|------------|
| Type | Following; Transfer | Product | Rev |
| Syntax | <!><a>TFS | 6K | 5.0 |
| Units | n/a | | |
| Range | n/a | | |
| Default | n/a | | |
| Response | TFS *TFS0000_0000_0000_0000_0000_0000 * 0000_0000_0000_0000_0000_0000 (repeated for each axis) 1TFS *1TFS0000_0000_0000_0000_0000_0000 bit 1 \leftarrow \leftarrow bit 24 | | |
| See Also | FGADV, FMCLN, FMCP, FOLEN, FOLMAS, FPPEN, [FS], FSHFC, FSHFD, MC, [NMCY], [PMAS], TFSF | | |

The Transfer Following Status (TFS) command returns the current Following status of all axes. The response for TFS is as follows (**Note:** response is product dependent):

| |
|--|
| FULL-TEXT STATUS REPORT AVAILABLE |
|--|

The TFS status command reports a binary bit report. If you would like to see a more descriptive text-based report, use the TFSF command description.

| Bit Assignment (left to right) | Function (YES = 1; NO = 0) | |
|--|----------------------------|---|
| 1 | Follower in Ratio Move | A Following move is in progress. |
| 2 | Ratio is Negative | The current ratio is negative (i.e., the follower counts are counting in the opposite direction from the master counts). |
| 3 | Follower Ratio Changing | The follower is ramping from one ratio to another (including a ramp to or from zero ratio). |
| 4 | Follower At Ratio | The follower is at constant non-zero ratio. |
| Bits 1-4 indicate the status of Following motion. They mimic the meaning and organization of Axis Status (TAS & AS) bits 1-4, except that each bit indicates the current state of the ratio, rather than the current state of the velocity. | | |
| * 5 | FOLMAS Active | A master is specified with the FOLMAS command. |
| * 6 | FOLEN Active | Following has been enabled with the FOLEN command. |
| * 7 | Master is Moving | The specified master is currently in motion. |
| 8 | Master Dir Neg | The current master direction is negative. (Bit must be cleared to allow Following move in preset mode—MC0). |
| Bits 5-8 indicate the status required for Following motion (i.e., a master must be assigned, Following must be enabled, the master must be moving, and for many features, the master direction must be positive). Unless the master is a commanded position of another axis, it is likely that minor vibration of the master will cause bits 7-8 to toggle on and off, even if the master is nominally "at rest". These bits are meant primarily as a quick diagnosis for the absence of master motion, or master motion in the wrong direction. Many features require positive master counting to work properly. | | |
| 9 | OK to Shift | Conditions are valid to issue shift commands (FSHFD or FSHFC). |
| 10 | Shifting now | A shift move is in progress. |
| 11 | Shift is Continuous | An FSHFC-based shift move is in progress. |
| 12 | Shift Dir is Neg | The direction of the shift move in progress is negative. |
| Bits 9-12 indicate the shift status of the follower. Shifting is super-imposed motion, but if viewed alone, can have its own status. In other words, bits 10-12 describe only the shifting portion of motion. | | |
| 13 | Master Cyc Trig Pend | A master cycle restart is pending the occurrence of the specified trigger. |
| 14 | Mas Cyc Len Given | A non-zero master cycle length has been specified with the FMCLLEN command. |
| 15 | Master Cyc Pos Neg | The current master cycle position (PMAS) is negative. This could be by caused by a negative initial master cycle position (FMCP), or if the master is moving in the negative direction. |
| 16 | Master Cyc Num > 0 | The master position (PMAS) has exceeded the master cycle length (FMCLLEN) at least once, causing the master cycle number (NMCY) to increment. |
| Bits 13-16 indicate the status of master cycle counting. If a Following application is taking advantage of master cycle counting, these bits provide a quick summary of some important master cycle information. | | |
| 17 | Mas Pos Prediction On | Master position prediction has been enabled (FPPEN). |
| 18 | Mas Filtering On | A non-zero value for master position filtering (FFILT) is in effect. |
| Bit 17-18 indicate the status of master position measurement features. | | |
| 19 | RESERVED | |
| 20 | RESERVED | |
| 21 | RESERVED | |
| 22 | RESERVED | |
| 23 | OK to do FGADV move | OK to do Geared Advance move (master assigned with FOLMAS, Following enabled with FOLEN, and follower axis is either not moving, or moving at constant ratio in continuous mode). |
| 24 | FGADV move underway | Geared Advance move profile is in progress. |

* All these conditions must be true before Following motion will occur.

TFSF Transfer Following Status (full-text report)

| | | | |
|----------|---|----------------|------------|
| Type | Following; Transfer | Product | Rev |
| Syntax | <!><a>TFSF | 6K | 5.0 |
| Units | n/a | | |
| Range | n/a | | |
| Default | n/a | | |
| Response | TFSF: (see example below) | | |
| See Also | FGADV, FMCLLEN, FMCP, FOLEN, FOLMAS, FPPEN, [FS], FSHFC, FSHFD, MC, [NMCY], [PMAS], TFS | | |

The TFSF command returns a text-based status report of all axes. This is an alternative to the binary report (TFS).

Example TFSF response:

| *TFSF | AXIS # | | | | | | | |
|-------------------------|--------|-----|-----|-----|-----|-----|-----|-----|
| * | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| *Follower in Ratio Move | NO | NO | NO | NO | NO | NO | NO | NO |
| *Ratio is Negative | NO | NO | NO | NO | NO | NO | NO | NO |
| *Followr Ratio Changing | NO | NO | NO | NO | NO | NO | NO | NO |
| *Follower At Ratio | NO | NO | NO | NO | NO | NO | NO | NO |
| * | | | | | | | | |
| *FOLMAS Active | NO | NO | NO | NO | NO | NO | NO | NO |
| *FOLEN Active | NO | NO | NO | NO | NO | NO | NO | NO |
| *Master is Moving | NO | NO | NO | NO | NO | NO | NO | NO |
| *Master Dir Neg | NO | NO | NO | NO | NO | NO | NO | NO |
| * | | | | | | | | |
| *OK to Shift | NO | NO | NO | NO | NO | NO | NO | NO |
| *Shifting now | NO | NO | NO | NO | NO | NO | NO | NO |
| *Shift is Continuous | NO | NO | NO | NO | NO | NO | NO | NO |
| *Shift Dir is Neg | NO | NO | NO | NO | NO | NO | NO | NO |
| * | | | | | | | | |
| *Master Cyc Trig Arm | NO | NO | NO | NO | NO | NO | NO | NO |
| *Mas Cyc Len Given | NO | NO | NO | NO | NO | NO | NO | NO |
| *Master Cyc Pos Neg | NO | NO | NO | NO | NO | NO | NO | NO |
| *Master Cyc Num > 0 | NO | NO | NO | NO | NO | NO | NO | NO |
| * | | | | | | | | |
| *Pos Prediction On | YES | YES | YES | YES | YES | YES | YES | YES |
| *Master Filtering On | NO | NO | NO | NO | NO | NO | NO | NO |
| *RESERVED | NO | NO | NO | NO | NO | NO | NO | NO |
| *RESERVED | NO | NO | NO | NO | NO | NO | NO | NO |
| * | | | | | | | | |
| *RESERVED | NO | NO | NO | NO | NO | NO | NO | NO |
| *RESERVED | NO | NO | NO | NO | NO | NO | NO | NO |
| *OK to do FGADV move | NO | NO | NO | NO | NO | NO | NO | NO |
| *FGADV move underway | NO | NO | NO | NO | NO | NO | NO | NO |

TGAIN Transfer Servo Gains

| | | | |
|----------|--|----------------|---------------------------------|
| Type | Transfer | Product | Rev |
| Syntax | <@><a>TGAIN | 6K | 5.0 |
| Units | n/a | | |
| Range | n/a | | |
| Default | n/a | | (applicable to servo axes only) |
| Response | TGAIN: *SGP1,2,3,4 ... *SGI.1,.1,0,0 ... *SGV25,25,40,40 ... *SGVF100,100,100,100 ... *SGAF0,0,0,0 ... 1TGAIN: *1SGP1 *1SGI.1 *1SGV25 *1SGVF100 *1SGAF0 | | |
| See Also | SFB, SGAF, SGENB, SGI, SGILIM, SGP, SGSET, SGV, SGVF, SOFFS, TSGSET | | |

This command allows you to display the current value of each of the control algorithm gains (SGP, SGI, SGV, SGAF, & SGVF). Each time an individual gain is entered, the current value is updated to be that value. When a gain set is enabled with the SGENB command, the current value of each gain is set to the values saved in that particular gain set.

NOTE

Tuning gains are specific to the feedback source that was in use (selected with the last SFB command) at the time the gains were established with the respective gain commands (SGI, SGP, etc.).

Example:

```
SGP5,5,10,10 ; Set the gains for the proportional gain
SGI.1,.1,0,0 ; Set the gains for the integral gain
SGV50,60,0,0 ; Set the gains for the velocity gain
SGVF5,6,10,11 ; Set the gains for the velocity feedforward gain
SGAF0,0,0,0 ; Set the gains for the acceleration feedforward gain
TGAIN ; Display current values for all gains. Example response:
; *SGP5,5,10,10
; *SGI.1,.1,0,0
; *SGV50,60,0,0
; *SGVF5,6,10,11
; *SGAF0,0,0,0
```

[TIM] Current Timer Value

| | | | |
|----------|--|---------|-----|
| Type | Assignment or Comparison | Product | Rev |
| Syntax | See below | 6K | 5.0 |
| Units | Milliseconds | | |
| Range | Maximum count is 999,999,999 (approx. 11 days, 13 hours) | | |
| Default | n/a | | |
| Response | n/a | | |
| See Also | TIMINT, TIMST, TIMSTP, TTIM | | |

The Current Timer Value (TIM) command is used to assign the timer value to a variable, or to make a comparison against another value. The value returned is in milliseconds.

Syntax: VARx=<n%>TIM where x is a numeric variable number, or TIM can be used in an expression such as IF(TIM<2400). Multi-tasking: If addressing the timer of a specific task, include the n% prefix.

Example:

```
VAR1=TIM ; Timer value is assigned to variable 1
IF(TIM<1000) ; If timer value is < 1000 milliseconds, do the IF statement
VAR1=TIM + 10 ; Timer value plus 10 assigned to variable 1
NIF ; End IF statement
```

TIMINT Timer Value to Cause Alarm Event

| | | | |
|----------|--|---------|-----|
| Type | Timer; Alarm Event | Product | Rev |
| Syntax | <!>TIMINT,<i> | 6K | 5.0 |
| Units | i = milliseconds | | |
| Range | b = 0 (reset and start) or 1 (stop) i = 0 - 999,999,999 | | |
| Default | 0,0 | | |
| Response | TIMINT: *TIMINT0,0 | | |
| See Also | INTHW, [TIM], TIMST, TIMSTP, TTIM | | |

The TIMINT command sets the timer value upon which the 6K controller will trigger an Alarm Event. The time value at which the alarm event will occur is specified by the second field in the command.

NOTES

- To use TIMINT, you must first issue the INTHW.21-1 command to enable checking for the alarm event.
- When using multi-tasking, this feature only works with the timer for Task zero.

The TIMINT command also determines if the timer is to be stopped when the value is reached, or if the timer is to be reset and started again. If the timer is to be stopped upon reaching the alarm value, a one should be specified for the first field. If the timer is to be reset and restarted upon reaching the alarm value,

a zero should be specified for the first field. By specifying a zero in the first field, an alarm will occur repeatedly.

Example:

```
INTHW.21-1      ; Enable checking for the timer-driven alarm event
TIMINT1,10000   ; Trigger alarm once after 10000 ms, do not restart the timer
TIMST0         ; Reset and start timer
```

| | | | | |
|--------------|---|--------------------|----------------|------------|
| TIMST | | Start Timer | | |
| Type | Timer | | Product | Rev |
| Syntax | <!>TIMST<r> | | 6K | 5.0 |
| Units | b = Enable bit r = time (milliseconds) if b = 0, task # if b = 1 | | | |
| Range | b = 0 (reset and start) or 1 (start from previous TIMSTP) r = absolute time 0-999,999,999 if b = 0, or r = task # 0-10 if b = 1 | | | |
| Default | 0 | | | |
| Response | TIMST: No response, acts as if TIMST1 command was issued | | | |
| See Also | SSFR, [TIM], TIMINT, TIMSTP, TTIM | | | |

The Start Timer (TIMST) command is used to start the timer.

- If TIMST0, you can start the timer at a specific time in milliseconds (e.g., TIMST0, 500).
- If TIMST1, you can resume the timer (after stopping it with the TIMSTP command) with the value of the time of the specified task (e.g., TIMST1, 3).

The timer resolution is 2 ms. The delay for executing TIMST and TIMSTP in combination is 4-6 ms.

If the timer is started and allowed to roll over the maximum timer count of 999,999,999 milliseconds (11 days, 13 hours, 46 minutes, 39.999 seconds), the timer will be stopped, and the value will be frozen at the maximum value.

Multi-Tasking: Each task has its own timer.

Example:

```
TIMST0         ; Reset and start timer
GO1100         ; Initiate motion on axes 1 and 2
TIMSTP        ; Stop timer
TTIM          ; Transfer time required for move
```

| | | | | |
|---------------|------------------------------------|-------------------|----------------|------------|
| TIMSTP | | Stop Timer | | |
| Type | Timer | | Product | Rev |
| Syntax | <!><%>TIMSTP | | 6K | 5.0 |
| Units | n/a | | | |
| Range | n/a | | | |
| Default | n/a | | | |
| Response | n/a | | | |
| See Also | SSFR, [TIM], TIMINT, TIMST, TTIM | | | |

The Stop Timer (TIMSTP) command stops the timer. This command in conjunction with the start timer (TIMST) command, provides a timer that can be used to time internal or external events.

The timer resolution is 2 ms. The delay for executing TIMST and TIMSTP in combination is 4-6 ms.

Multi-Tasking: Each task has its own timer.

Example:

```
TIMST0         ; Reset and start timer
GO1100         ; Initiate motion on axes 1 and 2
TIMSTP        ; Stop timer
TTIM          ; Transfer time required for move
```

TIN Transfer Input Status

| | | | |
|----------|--|----------------|------------|
| Type | Transfer | Product | Rev |
| Syntax | <!><@>TIN<.i> | 6K | 5.0 |
| Units | i = input number on the specified I/O brick (B) – see page 6 | | |
| Range | 1-32 (product dependent) | | |
| Default | n/a | | |
| Response | TIN: *0000_0000_0000_0000_0 (onboard trigger inputs) 1TIN *0000_0000_0000_0000_0000_0000_0000_0000 1TIN.4: *1 (status of I/O point 4 on I/O brick 1) | | |

See Also [IN], INFNC, INLVL, TINO, TLIM

The Transfer Input Status (TIN) command returns the current status (active or inactive) of the programmable inputs. The input is *active* when it is grounded. The active level (active high or active low) for the inputs is established with the INLVL command. “High” means that current is flowing and no voltage is present at the input terminal; conversely, “low” means that no current is flowing and a voltage may be present at the input terminal. If the active level is set to active low (INLVLØ – default), the TIN response indicates active with a one (1) and inactive with a zero (Ø). If the active level is set to active high (INLVL1), the TIN response indicates active with a zero (Ø) and inactive with a one (1).

The inputs are numbered 1 to *n* from left to right (*n* is the maximum number of I/O points on the I/O brick). The amount of onboard and external inputs varies by product and number of external I/O bricks — refer to page 6 for details.

If the status of a specific input is required, use the bit select operator (.). For example, 1TIN.9 reports the status of the 1st I/O point on the 2nd SIM of I/O brick 1.

TINO Transfer Other Input Status

| | | | |
|----------|--|----------------|------------|
| Type | Transfer | Product | Rev |
| Syntax | <!>TINO<.i> | 6K | 5.0 |
| Units | i = number of input (see below) | | |
| Range | 6 | | |
| Default | n/a | | |
| Response | TINO: *TINO0000_0100 TINO.6: *1 (status of input number 6 – ENABLE input) | | |

See Also [INO], TINOF

The Transfer Other Input Status (TINO) command returns the status of all of the inputs not covered by the TLIM or TIN commands. These 8 additional inputs may be used for status feedback.

TINO response: *TINO[^]bbbb[^]_bbbb
 Bit #1 Bit #12

| |
|--|
| FULL-TEXT STATUS REPORT AVAILABLE |
|--|

The TINO status command reports a binary bit report. If you would like to see a more descriptive text-based report, use the TINOF command description.

| Bit | Function | Location |
|-----|----------------------------------|-------------------|
| 1-5 | RESERVED | |
| 6 | Enable input (1 = OK for motion) | “ENABLE” terminal |
| 7-8 | RESERVED | |

TINOF Transfer Other Input Status (full-text report)

| | | | |
|----------|-------------------------------|----------------|------------|
| Type | Transfer | Product | Rev |
| Syntax | <!>TINOF | 6K | 5.0 |
| Units | n/a | | |
| Range | n/a | | |
| Default | n/a | | |
| Response | TINOF: (see example below) | | |
| See Also | [INO], TINO | | |

The TINOF command returns a text-based status report of all axes. This is an alternative to the binary report (TINO).

Example response:

```
*TINOF
*Enable input OK    YES
```

TIO Transfer Current Expansion I/O Status

| | | | |
|----------|-----------------------------|----------------|------------|
| Type | Transfer | Product | Rev |
| Syntax | <!>TIO | 6K | 5.0 |
| Units | B = I/O brick number | | |
| Range | 1-8 | | |
| Default | n/a | | |
| Response | (see example below) | | |
| See Also | TIN, TINO, TLIM, TOUT, TANI | | |

The TIO command displays the status of the current I/O configuration for the controller's expansion I/O bricks. If an I/O brick is not connected, it will not be included in the status report. Onboard I/O is not reported.

The I/O bricks are connected in a series to the "EXPANSION I/O" connector (see *Installation Guide* for instructions). The 1st I/O brick in the series (closest to the 6K product) is BRICK 1. The next is BRICK 2, and so on.

Each I/O brick has 4 SIM slots and can hold from 1 to 4 I/O SIM modules. A SIM slot may hold a digital input SIM, a digital output SIM, or an analog input SIM. Each SIM provides 8 inputs or outputs; therefore, each I/O brick has 32 I/O addresses, referenced as absolute I/O point locations:

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

The TIO response for each I/O brick is separated into four lines, one for each SIM. I/O points 1-8 represent SIM #1, 9-16 represents SIM #2, 17-24 represents SIM #3, and 25-32 represents SIM #4. When digital outputs are detected, the report also indicates whether the jumper is set to SINKING or SOURCING. When digital inputs and outputs are detected, TIO displays the current hardware state and programmed function (INFNC for inputs and OUTFNC for outputs). When analog inputs are detected, TIO reports the current voltage present on each input.

Example TIO responses (in this example, 2 I/O bricks are connected to the controller):

```

>TIO
*BRICK 1:  SIM Type      Status      Function
          1-8: DIGITAL INPUTS 0000_0000  AAAA_AAAA
          9-16: DIGITAL INPUTS 0000_0000  AAAA_AAAA
          17-24: DIGITAL INPUTS 0000_0000  AAAA_AAAA
          25-32: ANALOG INPUTS 0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000
*BRICK 2:  SIM Type      Status      Function
          1-8: DIGITAL OUTPUTS 0000_0000  AAAA_AAAA -- SINKING
          9-16: DIGITAL INPUTS 0000_0000  AAAA_AAAA
          17-24: NO SIM PRESENT
          25-32: DIGITAL OUTPUTS 0000_0000  AAAA_AAAA -- SOURCING

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

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

```

| TLABEL | | Transfer Labels | |
|---------------|----------------------------|------------------------|------------|
| Type | Transfer | Product | Rev |
| Syntax | <!>TLABEL | 6K | 5.0 |
| Units | n/a | | |
| Range | n/a | | |
| Default | n/a | | |
| Response | TLABEL: *NO LABELS DEFINED | | |
| See Also | \$ | | |

The Transfer Labels (TLABEL) command returns the names of all the labels defined with the \$ command.

The response to a TLABEL command if the labels call and open are defined in a program named prog1 is as follows:

```

*CALL DEFINED IN PROGRAM PROG1
*OPEN DEFINED IN PROGRAM PROG1

```

TNMCY Transfer Master Cycle Number

| | | | |
|----------|--|----------------|------------|
| Type | Following; Transfer | Product | Rev |
| Syntax | <!><a>TNMCY | 6K | 5.0 |
| Units | n/a | | |
| Range | n/a | | |
| Default | n/a | | |
| Response | TNMCY *TNMCY0,0,0,0,0,0,0,0 1TNMCY *1TNMCY0 | | |
| See Also | FMCLLEN, FMCNEW, [FS], TRGFN, TFS | | |

The Transfer Master Cycle Number (TNMCY) command displays the current master cycle number for all axes, or the axis specified. The value represents the current cycle number, not the position of the master (or the follower). The master cycle number is set to zero when master cycle counting is restarted, and is incremented each time a master cycle finishes (i.e., rollover occurs). It will often correspond to the number of complete parts in a production run. This value may be used for subsequent decision making, or simply recording the cycle number corresponding to some other event.

The master must be assigned first (FOLMAS command) before this command will be useful.

For a complete discussion of master cycles, please refer to the Following chapter in the *Programmer's Guide*.

TNTMAC Transfer Ethernet Address

| | | | |
|----------|------------------------------------|----------------|------------|
| Type | Transfer; Communications Interface | Product | Rev |
| Syntax | <!>TNTMAC | 6K | 5.0 |
| Units | n/a | | |
| Range | n/a | | |
| Default | n/a | | |
| Response | TMAC: *0,144,85,0,0,1 | | |
| See Also | NTADDR, NTMASK | | |

The TNTMAC command reports the 6K product's Ethernet address.

TOUT Transfer Output Status

| | | | |
|----------|--|----------------|------------|
| Type | Transfer | Product | Rev |
| Syntax | <!><@>TOUT<.i> | 6K | 5.0 |
| Units | i = input number on the specified I/O brick (B) – see page 6 | | |
| Range | 1-32 (Product dependent) | | |
| Default | n/a | | |
| Response | TOUT: *0000_0000 (onboard outputs) 1TOUT *0000_0000_0000_0000_0000_0000_0000_0000 1TOUT.4: *1 (status of I/O point 4 on I/O brick 1) | | |
| See Also | OUT, OUTFNC, OUTLVL, TIN, TINO | | |

The Transfer Output Status (TOUT) command returns the current status (active or inactive) of the programmable outputs. The output is *active* when it is grounded. The active level (active high or active low) for the outputs is established with the OUTLVL command. “High” means that current is flowing and no voltage is present at the output terminal; conversely, “low” means that no current is flowing and a voltage may be present at the output terminal. If the active level is set to active low (OUTLVLØ – default), the TOUT response indicates active with a one (1) and inactive with a zero (Ø). If the active level is set to active high (OUTLVL1), the TOUT response indicates active with a zero (Ø) and inactive with a one (1).

The outputs are numbered 1 to *n* from left to right (*n* is the maximum number of I/O points on the I/O brick). The amount of onboard and external outputs varies by product and number of external I/O bricks — refer to page 6 for details.

If the status of a specific output is required, use the bit select operator (.). For example, 1TOUT.9 reports the status of the 1st I/O point on the 2nd SIM of I/O brick 1.

TPANI Transfer Position of ANI Inputs

| | | | |
|----------|--|----------------|---------------------------------|
| Type | Transfer | Product | Rev |
| Syntax | <!>TPANI<.i> | 6K | 5.0 |
| Units | i = location of the analog input on the I/O brick () | | |
| Range | 1-32 (depending on I/O brick configuration) | | (applicable only to servo axes) |
| Default | n/a | | |
| Response | 1TPANI: *1TPANIxxxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx 1TPANI.1 *108 (position of analog input at I/O pin 1 on I/O brick 1) | | |
| See Also | [ANI], ANIRNG, CMDDIR, [FB], [PANI], PSET, SCALE, SCLD, SFB, TANI, TFB | | |

The TPANI command returns the value of the ANI analog inputs as modified by scaling (SCLD), offset (PSET), and commanded direction polarity (CMDDIR).

The TPANI and PANI commands are designed for applications in which the ANI input is scaled and/or used as position feedback. If you are using the ANI input to monitor an analog signal, the TANI and ANI commands would be more appropriate (TANI and ANI values are measured in volts and are unaffected by scaling, polarity, or command direction).

The TPANI value is represented in analog-to-digital converter (ADC) units if scaling is disabled (SCALE0). The ADC has a 12-bit resolution, giving a range of +2047 to -2048 counts when using the full $\pm 10V$ range of the analog input (205 counts/volt). If scaling is enabled (SCALE1), an SCLD scale factor of 205 (the default value when analog input feedback is selected) allows units of volts to be used.

NOTE: If you change the voltage range of the analog input (with the ANIRNG command), the resolution of the PANI response will change accordingly. The default is $\pm 10V$.

TPC Transfer Position Commanded

| | | | |
|----------|---|----------------|------------|
| Type | Transfer | Product | Rev |
| Syntax | <!><@><a>TPC | 6K | 5.0 |
| Units | Reported value represents distance units (scalable by SCLD) | | |
| Range | Range of the reported value is $\pm 2,147,483,648$ | | |
| Default | n/a | | |
| Response | TPC: *TPC+0,+0,+0,+0,+0,+0,+0,+0 1TPC: *1TPC+0 | | |
| See Also | CMDDIR, ERES, [PC], [PCC], PSET, SCALE, SCLD, SMPER, TAS, TFB, TPCC, TPER | | |

This command allows you to display the current *commanded position* of each axis. The TPC value is scaled by the distance scaling factor (SCLD) if scaling is enabled with the SCALE1 command.

Servo Axes: The reported value is measured in encoder or analog input (ANI) counts.

Stepper Axes: The reported value is measured in commanded counts (“motor counts”).

If you issue a PSET command, the commanded position value will be offset by the PSET command value.

Servo Axes: The commanded position (TPC) and the actual position (TFB) are used in the control algorithm to calculate the position error ($TPC - TFB = TPER$) and thereby determine the corrective control signal.

Example:

```
TPC          ; Display the current commanded position for each axis:
              ; *TPC4000,4000,4000,4000 (setpoints displayed in steps)
TFB          ; Display the current actual position for each axis:
              ; *TFB4004,4005,4004,4003 (actual positions displayed in steps)
TPER         ; Display current position error of each axis:
              ; *TPER-4,-5,-4,-3 (error displayed in steps)
```

TPCC Transfer Captured Commanded Position

| | | | |
|----------|---|----------------|------------|
| Type | Transfer | Product | Rev |
| Syntax | <!>aTPCCc | 6K | 5.0 |
| Units | a = axis # c = trigger input letter (A, B or M) for axis "a" (Reported value is commanded counts, scalable by SCLD) | | |
| Range | n/a | | |
| Default | n/a | | |
| Response | 1TPCCA: *1TPCCA+0 | | |
| See Also | CMDDIR, ENCCNT, INFNC, [PC], [PCC], [PCMS], PSET, SCALE, SCLD, SFB, TFB, TPC [TRIG], TRGLOT, TTRIG | | |

Use the TPCC command to display the current captured commanded position of a specific axis, captured with the specific "trigger interrupt" input.

| Trigger Input (Axis 1-4 "TRIGGERS/OUTPUTS" connector) * Axis | | | Dedicated Axis | TPCC Syntax | Trigger Input (Axis 5-8 "TRIGGERS/OUTPUTS" connector) * Axis | | | Dedicated Axis | TPCC Syntax |
|---|------------|--|-------------------|----------------|---|------------|--|-------------------|----------------|
| Pin 23, | Trigger 1A | | 1 | 1TPCCA | Pin 23, | Trigger 5A | | 5 | 5TPCCA |
| Pin 21, | Trigger 1B | | 1 | 1TPCCB | Pin 21, | Trigger 5B | | 5 | 5TPCCB |
| Pin 19, | Trigger 2A | | 2 | 2TPCCA | Pin 19, | Trigger 6A | | 6 | 6TPCCA |
| Pin 17, | Trigger 2B | | 2 | 2TPCCB | Pin 17, | Trigger 6B | | 6 | 6TPCCB |
| Pin 15, | Trigger 3A | | 3 | 3TPCCA | Pin 15, | Trigger 7A | | 7 | 7TPCCA |
| Pin 13, | Trigger 3B | | 3 | 3TPCCB | Pin 13, | Trigger 7B | | 7 | 7TPCCB |
| Pin 11, | Trigger 4A | | 4 | 4TPCCA | Pin 11, | Trigger 8A | | 8 | 8TPCCA |
| Pin 9, | Trigger 4B | | 4 | 4TPCCB | Pin 9, | Trigger 8B | | 8 | 8TPCCB |

* The number of trigger inputs available varies by product (refer to your product's *Installation Guide*).

To report an axis position captured with the MASTER TRIG input, use aPCCM, where "a" can be any axis number.

About Position Capture: The commanded position can be captured only by a trigger input that is defined as "trigger interrupt" input with the INFNCi-H command (see INFNC for details). Each trigger input, when configured as a "trigger interrupt" input, is dedicated to capture the position of a specific axis (see table above). When a "trigger interrupt" input is activated, the commanded position of the dedicated axis is captured and the position is available through the use of the PCC operator and the TPCC display command.

Note for Stepper Axes: By default, stepper axes capture only the commanded position. However, if the axis has Encoder Capture Mode enabled with the ENCCNT command, only the encoder position is captured.

Position Capture Status, Longevity of Captured Position: Use the TTRIG and TRIG commands to ascertain if a trigger interrupt input has been activated. TTRIG displays the status as a binary report, and TRIG is an assignment/comparison operator for using the status information in a conditional expression (e.g., in an IF statement). Once the captured commanded position value is displayed with the TPCC command, the TTRIG/TRIG status bit for that trigger input is cleared; but the position information remains available until it is overwritten by a subsequent position capture from the same trigger input.

Position Capture Accuracy: The commanded position capture accuracy is ± 1 count.

Scaling and Position Offset: If scaling is enabled (SCALE1), the commanded position is scaled by the distance scaling factor (SCLD). If scaling is not enabled (SCALE0), the value reported will be actual commanded counts. If you issue a PSET (establish absolute position reference) command, any previously captured commanded positions will be offset by the PSET command value.

Example:

```
1TPCCA      ; Report axis 1's captured command position, which was captured
            ; when the dedicated trigger (TRG-1A) was activated
3TPCCB      ; Report axis 3's captured command position, which was captured
            ; when the dedicated trigger (TRG-3B) was activated
2TPCCM      ; Report axis 2's captured command position, which was captured
            ; when the master trigger (TRG-M) was activated
```

TPCE Transfer Position of Captured Encoder

| | | | |
|----------|---|----------------|------------|
| Type | Transfer | Product | Rev |
| Syntax | <!>aTPCEc | 6K | 5.0 |
| Units | a = axis # c = trigger input letter (A, B or M) for axis "a" (Reported value represents encoder counts, scalable by SCLD) | | |
| Range | n/a | | |
| Default | n/a | | |
| Response | 1TPCEA: *1TPCEA+0 | | |
| See Also | CMDDIR, ENCCNT, ENCPOL, INFNC, [PCE], PESET, PSET, SCALE, SCLD, SFB, TPE | | |

Use the TPCE command to display the current captured encoder position, from the time of the last trigger interrupt.

| Trigger Input (Axis 1-4 "TRIGGERS/OUTPUTS" connector) * Axis | Dedicated Axis | TPCE Syntax | Trigger Input (Axis 5-8 "TRIGGERS/OUTPUTS" connector) * Axis | Dedicated Axis | TPCE Syntax |
|---|----------------|-------------|---|----------------|-------------|
| Pin 23, Trigger 1A | 1 | 1TPCEA | Pin 23, Trigger 5A | 5 | 5TPCEA |
| Pin 21, Trigger 1B | 1 | 1TPCEB | Pin 21, Trigger 5B | 5 | 5TPCEB |
| Pin 19, Trigger 2A | 2 | 2TPCEA | Pin 19, Trigger 6A | 6 | 6TPCEA |
| Pin 17, Trigger 2B | 2 | 2TPCEB | Pin 17, Trigger 6B | 6 | 6TPCEB |
| Pin 15, Trigger 3A | 3 | 3TPCEA | Pin 15, Trigger 7A | 7 | 7TPCEA |
| Pin 13, Trigger 3B | 3 | 3TPCEB | Pin 13, Trigger 7B | 7 | 7TPCEB |
| Pin 11, Trigger 4A | 4 | 4TPCEA | Pin 11, Trigger 8A | 8 | 8TPCEA |
| Pin 9, Trigger 4B | 4 | 4TPCEB | Pin 9, Trigger 8B | 8 | 8TPCEB |

* The number of trigger inputs available varies by product (refer to your product's *Installation Guide*).

To report an axis position captured with the MASTER TRIG input, use aPCEM, where "a" can be any axis number.

About Position Capture: The encoder position can be captured only by a trigger input that is defined as "trigger interrupt" input with the `INFNCi-H` command (see `INFNC` command). Each trigger input, when configured as a "trigger interrupt" input, is dedicated to capture the position of a specific axis (see table above). When a "trigger interrupt" input is activated, the encoder position of the dedicated axis is captured and the position is available through the use of the `PCE` operator and the `TPCE` display command. **Stepper Axes:** By default, stepper axes capture only the commanded position. To capture the encoder position, the axis must be in the Encoder Capture Mode (see `ENCCNT` command).

Position Capture Status, Longevity of Captured Position: Use the `TTRIG` and `TRIG` commands to ascertain if a trigger interrupt input has been activated. `TTRIG` displays the status as a binary report, and `TRIG` is an assignment/comparison operator for using the status information in a conditional expression (e.g., in an `IF` statement). Once the captured encoder position value is reported with the `TPCE` command, the `TTRIG/TRIG` status bit for that trigger input is cleared; but the position information remains available until it is overwritten by a subsequent position capture from the same trigger input.

Position Capture Accuracy: The encoder position capture accuracy is ± 1 encoder count.

Scaling and Position Offset: If scaling is enabled (`SCALE1`), the encoder position is scaled by the distance scaling factor (`SCLD`). If scaling is not enabled (`SCALE0`), the value reported will be actual encoder counts. If you issue a `PSET` (establish absolute position reference) command, any previously captured encoder positions will be offset by the `PSET` command value.

Example:

```
1TPCEA ; Report axis 1's captured encoder position, which was captured
        ; when the dedicated trigger (TRG-1A) was activated
3TPCEB ; Report axis 3's captured encoder position, which was captured
        ; when the dedicated trigger (TRG-3B) was activated
2TPCEM ; Report axis 2's captured encoder position, which was captured
        ; when the master trigger (TRG-M) was activated
```

TPCME Transfer Captured Master Encoder Position

| | | | |
|----------|---|---------|-----|
| Type | Transfer | Product | Rev |
| Syntax | <!>TPCME | 6K | 5.0 |
| Units | n/a | | |
| Range | n/a | | |
| Default | n/a | | |
| Response | TPCME *TPCME+0 | | |
| See Also | INFNC, MEPOL, MESND, [PME], [PCME], [PCMS], PMECLR, PMESET, TPME, TPCMS | | |

Use the TPCME command to display the current captured master encoder position. The master encoder is connected to the connector labeled “Master Encoder.”

Syntax: VARn=PCME where n is the variable number; or PCME can be used in an expression such as IF(PCME>23450).

About Position Capture: The master encoder position can be captured only by the Master Trigger input (labeled “MASTER TRIG”), and only when that input is defined as a “trigger interrupt” input with the INFNC17-H command (see INFNC command). When the “trigger interrupt” input is activated (active edge), the master encoder position is captured and the position is available through the use of the PCME operator and the TPCME display command.

Position Capture Status, Longevity of Captured Position: Use the TTRIG and TRIG commands to ascertain if a trigger interrupt input has been activated. TTRIG displays the status as a binary report, and TRIG is an assignment/comparison operator for using the status information in a conditional expression (e.g., in an IF statement). Once the captured master encoder position value is displayed with the TPCME command, TTRIG/TRIG status bit #17 is cleared; but the position information remains available until it is overwritten by a subsequent position capture from the master trigger input.

Position Capture Accuracy: The master encoder position capture accuracy is ± 1 encoder count.

Scaling and Position Offset: The TPCME value is always in master encoder counts; it is never scaled. If you issue a PMESET (establish absolute position reference) command, any previously captured master encoder positions will be offset by the PMESET command value.

TPCMS Transfer Captured Master Cycle Position

| | | | |
|----------|--|---------|-----|
| Type | Transfer | Product | Rev |
| Syntax | <!>aTPCMSc | 6K | 5.0 |
| Units | a = axis # c = trigger input letter (A, B or M) for axis "a" (Reported value represents master counts, scalable by SCLMAS) | | |
| Range | n/a | | |
| Default | n/a | | |
| Response | 1TPCMSA *1TPCMSA+0 | | |
| See Also | CMDDIR, ENCCNT, ENCPOL, FOLMAS, INFNC, [PCMS], PSET, SCALE, SCLMAS, SFB, TPCMS, [TRIG], TRGLOT, TTRIG | | |

The TPCMS command transfers the captured position of the master within its current master cycle.

TPCMS (and TPMAS) is unique among position transfers, because its value rolls over to zero each time the entire master cycle length (FMCLEN) has been traveled. Thus, the captured TPCMS value is essentially a snap-shot of the position relative to the master cycle at the time of the capture.

The master must be assigned first (FOLMAS command) before this command will be useful.

For a complete discussion of master cycles, refer to the Following chapter in the *6K Series Programmer's Guide*.

| Trigger Input (Axis 1-4 "TRIGGERS/OUTPUTS" connector) * Axis | Dedicated | TPCMS Syntax | Trigger Input (Axis 5-8 "TRIGGERS/OUTPUTS" connector) * Axis | Dedicated | TPCMS Syntax |
|---|-----------|-----------------|---|-----------|-----------------|
| Pin 23, Trigger 1A | 1 | 1TPCMSA | Pin 23, Trigger 5A | 5 | 5TPCMSA |
| Pin 21, Trigger 1B | 1 | 1TPCMSB | Pin 21, Trigger 5B | 5 | 5TPCMSB |
| Pin 19, Trigger 2A | 2 | 2TPCMSA | Pin 19, Trigger 6A | 6 | 6TPCMSA |
| Pin 17, Trigger 2B | 2 | 2TPCMSB | Pin 17, Trigger 6B | 6 | 6TPCMSB |
| Pin 15, Trigger 3A | 3 | 3TPCMSA | Pin 15, Trigger 7A | 7 | 7TPCMSA |
| Pin 13, Trigger 3B | 3 | 3TPCMSB | Pin 13, Trigger 7B | 7 | 7TPCMSB |
| Pin 11, Trigger 4A | 4 | 4TPCMSA | Pin 11, Trigger 8A | 8 | 8TPCMSA |
| Pin 9, Trigger 4B | 4 | 4TPCMSB | Pin 9, Trigger 8B | 8 | 8TPCMSB |

* The number of trigger inputs available varies by product (refer to your product's *Installation Guide*).

To report an axis position captured with the MASTER TRIG input, use aPCMSM, where "a" can be any axis number.

About Position Capture: The master cycle position can be captured only by a trigger input that is defined as "trigger interrupt" input with the INFNCi-H command (see INFNC command). Each trigger input, when configured as a "trigger interrupt" input, is dedicated to capture the position of a specific axis (see table above). When a "trigger interrupt" input is activated, the master cycle position of the dedicated axis is captured and the position is available through the use of the PCMS operator and the TPCMS display command.

Position Capture Status, Longevity of Captured Position: Use the TTRIG and TRIG commands to ascertain if a trigger interrupt input has been activated. TTRIG displays the status as a binary report, and TRIG is an assignment/comparison operator for using the status information in a conditional expression (e.g., in an IF statement). Once the captured master cycle position value is reported with the TPCMS command, the TTRIG/TRIG status bit for that trigger input is cleared; but the position information remains available until it is overwritten by a subsequent position capture from the same trigger input.

Position Capture Accuracy: The master cycle position is interpolated; the capture accuracy is 50 μs multiplied by the velocity of the axis at the time the trigger input was activated.

Scaling and Position Offset: If scaling is enabled (SCALE1), the master cycle position is scaled by the distance scaling factor (SCLMAS). If scaling is not enabled (SCALE0), the value assigned will be actual counts from the commanded or encoder master source as selected with the FOLMAS command. If you issue a PSET (establish absolute position reference) command, any previously captured master cycle positions will be offset by the PSET command value.

TPE

Transfer Position of Encoder

| | | | |
|----------|--|----------------|------------|
| Type | Transfer | Product | Rev |
| Syntax | <!><a>TPE | 6K | 5.0 |
| Units | (Reported value represents encoder counts, scalable by SCLD) | | |
| Range | n/a | | |
| Default | n/a | | |
| Response | TPE: *TPE+0,+0,+0,+0,+0,+0,+0,+0 1TPE: *1TPE+0 | | |
| See Also | CMDDIR, ENCCNT, ENCPOL, ENCSND, [FB], [PE], PESET, PSET, SCALE, SCLD, SFB, TFB | | |

The Transfer Position of Encoder (TPE) command returns the current encoder position. If the encoder has been configured to receive step and direction input (ENCSND), the TPE command will report the position as counted from the step and direction signal.

Stepper axes: If the ENCCNT1 mode is enabled TPE reports the encoder position, but in ENCCNT0 mode (the factory default setting) the TPE report represents the commanded position.

| |
|---|
| UNITS OF MEASURE and SCALING: refer to page 16 or to the SCLD command. |
|---|

If you issue a PSET command, the encoder position value will be offset by the PSET command value. If you are using a stepper axis in the ENCCNT1 mode, use the PESET command instead.

TPER

Transfer Position Error

| | | | |
|----------|---|---------------------------------|------------|
| Type | Transfers | Product | Rev |
| Syntax | <!><a>TPER | 6K | 5.0 |
| Units | Reported value represents distance units (scalable by SCLD) | | |
| Range | Range of the reported value is $\pm 2,147,483,648$ | (applicable only to servo axes) | |
| Default | n/a | | |
| Response | TPER: *TPER+0,+0,+0,+0,+0,+0,+0,+0 1TPER: *1TPER+0 | | |
| See Also | CMDDIR, DRES, ENCPOL, ERES, [FB], [PC], [PE], [PER], SFB, SMPER, TANI, TAS, TFB, TPE, TPC | | |

The Transfer Position Error (TPER) command allows you to display the current position error of each axis. The error is displayed in feedback device counts and is scaled by the distance scaling factor (SCLD), if scaling is enabled with the SCALE1 command.

The position error is the difference between the commanded position and the actual position read by the feedback device ($TPER = TPC - TFB$). This error is calculated every sample period and can be displayed at any time using this command.

Example:

```
TPC          ; Display the current commanded position for each axis:
              ; *TPC4000,4000,4000,4000 (setpoints displayed in steps)
TFB          ; Display the current actual position for each axis:
              ; *TFB4004,4005,4004,4003 (actual positions displayed in steps)
TPER         ; Display current position error of each axis:
              ; *TPER-4,-5,-4,-3 (error displayed in steps)
```

TPMAS Transfer Current Master Cycle Position

| | | | |
|----------|---|----------------|------------|
| Type | Following; Transfer | Product | Rev |
| Syntax | <! <a>tpmas< a="">tpmas<> | 6K | 5.0 |
| Units | Reported value represents master counts, scalable by SCLMAS. | | |
| Range | n/a | | |
| Default | n/a | | |
| Response | TPMAS *TPMAS0,0,0,0 1TPMAS *1TPMAS0 | | |
| See Also | FMCLEN, FMCNEW, FMCP, FOLMAS, FOLMD, [FS], MEPOL, [NMCY], [PMAS], SCALE, SCLMAS, TFS | | |

The TPMAS command transfers the current position of the master within its current master cycle. **The master must be assigned first (FOLMAS command) before this command will be useful.**

TPMAS is unique among position transfers, because master cycle position rolls over to zero each time the entire master cycle length (FMCLEN value) has been traveled.

If scaling is enabled (SCALE1), the value returned is scaled by the master scaling factor (SCLMAS). If scaling is disabled (SCALE0), the value returned is in master counts (encoder counts, commanded counts, or analog input counts).

For a complete discussion of master cycles, please refer to the Following chapter in the *Programmer's Guide*.

TPME Transfer Position of Master Encoder

| | | | |
|----------|--|----------------|------------|
| Type | Transfer | Product | Rev |
| Syntax | <! <a>tpme< a="">tpme<> | 6K | 5.0 |
| Units | Reported value represents master encoder counts. | | |
| Range | n/a | | |
| Default | n/a | | |
| Response | TPME *TPME+0 | | |
| See Also | MEPOL, MESND, [PCME], [PE], [PME], PMECLR, PMESET, TPCME | | |

Use the TPME command to display the current master encoder position. The master encoder is connected to the connector labeled “Master Encoder”. If you issue a PMESET command, the master encoder position value will be offset by the PMESET command value. The TPME value is always in encoder counts, it is never scaled.

TPROG Transfer Program Contents

| | | | |
|----------|---|----------------|------------|
| Type | Transfer | Product | Rev |
| Syntax | <! <a>tprog<t>< a="">tprog<t><> | 6K | 5.0 |
| Units | t = text (name of program) | | |
| Range | Text name of 6 characters or less | | |
| Default | n/a | | |
| Response | n/a | | |
| See Also | DEF, TDIR, TMEM | | |

The Transfer Program (TPROG) command displays the contents of the program specified. If there is no such program, then the error message *INVALID DATA will be generated. To see which programs have been created, use the TDIR command.

TPSHF Transfer Net Position Shift

| | | | |
|----------|--|----------------|------------|
| Type | Following; Transfer | Product | Rev |
| Syntax | <!><a>TPSHF | 6K | 5.0 |
| Units | (Reported value represents commanded counts, scalable by SCLD) | | |
| Range | n/a | | |
| Default | n/a | | |
| Response | TPSHF *TPSHF+0,+0,+0,+0,+0,+0,+0,+0 | | |
| See Also | FMCNEW, FMCP, FOLEN, FSHFC, FSHFD, [PSHF], SCALE, SCLD | | |

The TPSHF command transfers the net (absolute) follower axis position shift that has occurred since that last FOLEN1 command. The position returned will be the sum of all shifts performed on that axis, or axes, including decelerations due to limits, kill, or stop. The shift value is set to zero each time a new FOLEN1 command or a FOLMAS command (with a value other than zero) is issued.

If scaling is enabled (SCALE1), the PSHF value is scaled by the distance scaling factor (SCLD). If scaling is not enabled, the value is in commanded counts.

TPSLV Transfer Current Commanded Position of Follower Axis

| | | | |
|----------|--|----------------|------------|
| Type | Following; Transfer | Product | Rev |
| Syntax | <!><a>TPSLV | 6K | 5.0 |
| Units | (Reported value represents commanded counts, scalable by SCLD) | | |
| Range | n/a | | |
| Default | n/a | | |
| Response | TPSLV *TPSLV+0,+0,+0,+0,+0,+0,+0,+0 | | |
| See Also | FMCNEW, FMCP, [PSLV], SCALE, SCLD | | |

The TPSLV command transfers the current commanded position of the follower axis. The master must be assigned first (FOLMAS command) before this command will be useful.

If scaling is enabled (SCALE1), the PSLV value is scaled by the distance scaling factor (SCLD). If scaling is not enabled, the value is in commanded counts.

TRACE Program Trace Mode Enable

| | | | |
|----------|--|----------------|------------|
| Type | Program Debug Tool | Product | Rev |
| Syntax | <!>TRACE | 6K | 5.0 |
| Units | n/a | | |
| Range | b = 0 (disable), 1 (enable) or X (don't care) | | |
| Default | 0 | | |
| Response | TRACE: *TRACE0 | | |
| See Also | [,], [#], PORT, [SS], STEP, TRACEP, TRANS, TSS | | |

The Program Trace Mode Enable (TRACE) command enables program trace mode. When in program trace mode, all commands executed are or transferred out the Ethernet, RS-232 or RS-485 port, along with the program from which the command came.

Example:

```
DEF pick            ; Begin definition of program named pick
GO1100            ; Initiate motion on axes 1 and 2
IF(VAR1=5)        ; If variable 1 = 5 then do commands between IF and NIF
  GOTOpick1       ; Goto label pick1
  ELSE            ; Else part of IF command
  GOTOpick2       ; Goto label pick2
NIF               ; End IF command
$pick1            ; Label declaration for pick1
GO0011           ; Initiate motion on axes 3 and 4
BREAK            ; Break out of current subroutine or program
$pick2            ; Label declaration for pick2
GO1001           ; Initiate motion on axes 1 and 4
END               ; End program definition
TRACE1           ; Enable trace mode.
VAR1=5           ; Set variable 1 to 5
```

```

@LH0          ; Disable all limits
EOT13,10,0   ; Set End-of-Transmission characters to a carriage return
              ; and a line feed
RUNpick      ; Initiate program pick

```

After executing `RUN pick`, the following information will be placed in the output buffer, due to the trace mode being enabled. (Assume variable 1 = 5)

```

*PROGRAM=PICK COMMAND=GO1100
*PROGRAM=PICK COMMAND=IF(VAR1=5.0)
*PROGRAM=PICK COMMAND=GOTO PICK1
*PROGRAM=PICK COMMAND=$PICK1
*PROGRAM=PICK COMMAND=GO0011
*PROGRAM=PICK COMMAND=BREAK

```

TRACEP Program Flow Mode Enable

| Type | Program Debug Tool | Product | Rev |
|----------|---|---------|-----|
| Syntax | <!>TRACEP | 6K | 5.0 |
| Units | n/a | | |
| Range | b = 0 (disable), 1 (enable) or X (don't care) | | |
| Default | 0 | | |
| Response | TRACEP: *TRACEP0 | | |
| See Also | TRACE | | |

The Program Flow Mode Enable (`TRACEP`) command provides a debug tool to monitor the entry and exit of programs and their associated nest-levels.

Example:

```

DEF PICK1
GOSUB PICK2
GOTO PICK3
END

DEF PICK2
GOSUB PICK4
END

DEF PICK3
END

DEF PICK 4
END

>TRACEP1
>PICK1
*INITIATE PROGRAM: PICK1 NEST=1
*INITIATE PROGRAM: PICK2 NEST=2
*INITIATE PROGRAM: PICK4 NEST=3
*END:          PROGRAM NOW: PICK2 NEST=2
*END:          PROGRAM NOW: PICK1 NEST=1
*INITIATE PROGRAM: PICK3 NEST=1
*END:          PROGRAM EXECUTION TERMINATED

```

TRANS Translation Mode Enable

| | | | |
|----------|---|----------------|------------|
| Type | Program Debug Tool | Product | Rev |
| Syntax | <!>TRANS | 6K | 5.0 |
| Units | n/a | | |
| Range | b = 0 (disable), 1 (enable) or X (don't care) | | |
| Default | 0 | | |
| Response | TRANS: *TRANS0 | | |
| See Also | [#], [SS], STEP, TSS | | |

The Translation Mode Enable (TRANS) command enables the program translation mode, in which all commands processed by the 6K Series product are echoed back in their binary format (hex representation of the binary equivalent), and are not executed. The first byte (first two characters) of the response represents the command's memory requirement. The remaining bytes represent the actual command.

Example:

```
TRANS1          ; Enable translation mode
A10,20,1,1     ; Translate acceleration command A10,20,1,1. Response displayed
               ; is: 13 01 00 00 01 86 A0 00 03 0D 40 00 00 27 10 00 00 27 10.
               ; Note that 13 hex represents a command memory requirement of 19
               ; bytes.
GO1100         ; Translate initiate motion command GO1100. Response displayed
               ; is: 07 07 03 01 01 00 00. Note that 07 hex represents a
               ; command memory requirement of 7 bytes.
GO0011         ; Translate initiate motion command GO0011. Response displayed
               ; is: 07 07 03 00 00 01 01. Note that 07 hex represents a
               ; command memory requirement of 7 bytes.
```

TREV Transfer Revision Level

| | | | |
|----------|--|----------------|------------|
| Type | Transfer | Product | Rev |
| Syntax | <!>TREV | 6K | 5.0 |
| Units | n/a | | |
| Range | n/a | | |
| Default | n/a | | |
| Response | TREV: *TREV92-016740-01-5.0 (response varies by product) | | |
| See Also | RESET | | |

The Transfer Revision Level (TREV) command provides the current revision of the product's firmware. It also reports any options that have been installed. Options can be ordered through your local ATC or distributor.

TRGFN Trigger Functions

| | | | |
|----------|---|---------|-----|
| Type | Inputs; Following; Motion | Product | Rev |
| Syntax | <!><@>aTRGFNcbb | 6K | 5.0 |
| Units | a = axis # c = trigger input letter for axis "a" 1 st b = bit to select Conditional GO (GOWHEN) function 2 nd b = bit to select Start New Master Cycle (FMCNEW) function | | |
| Range | a = 1-8 (product dependent) c = A, B, or M (M is master trigger, "TRG-M") b = 0 (disable function), 1 (enable function), or X (leave unchanged) | | |
| Default | a = 1, c = A; b = 0 | | |
| Response | 1TRGFN *1TRGFNA00 | | |
| See Also | [AS], ERROR, ERRORP, FMCNEW, GOWHEN, INFNC, [SS], TAS, TRGLOT, TSS, [TRIG], TTRIG | | |

Use the TRGFN command to assign certain command functions to the onboard trigger inputs. *Note that the number of trigger inputs available varies by product — see page 6.*

| Trigger Input (Axis 1-4 "Triggers/Outputs" connector) * Axis | Dedicated Axis | TRGFN Syntax | Trigger Input (Axis 5-8 "Triggers/Outputs" connector) * | Dedicated Axis | TRGFN Syntax |
|---|----------------|--------------|--|----------------|--------------|
| Pin 23, Trigger 1A | 1 | 1TRGFNA | Pin 23, Trigger 5A | 5 | 5TRGFNA |
| Pin 21, Trigger 1B | 1 | 1TRGFNB | Pin 21, Trigger 5B | 5 | 5TRGFNB |
| Pin 19, Trigger 2A | 2 | 2TRGFNA | Pin 19, Trigger 6A | 6 | 6TRGFNA |
| Pin 17, Trigger 2B | 2 | 2TRGFNB | Pin 17, Trigger 6B | 6 | 6TRGFNB |
| Pin 15, Trigger 3A | 3 | 3TRGFNA | Pin 15, Trigger 7A | 7 | 7TRGFNA |
| Pin 13, Trigger 3B | 3 | 3TRGFNB | Pin 13, Trigger 7B | 7 | 7TRGFNB |
| Pin 11, Trigger 4A | 4 | 4TRGFNA | Pin 11, Trigger 8A | 8 | 8TRGFNA |
| Pin 9, Trigger 4B | 4 | 4TRGFNB | Pin 9, Trigger 8B | 8 | 8TRGFNB |

"Master Trigger" (TRIG-M) trigger: syntax is TRGFNM

* The number of trigger inputs available varies by product (refer to your product's *Installation Guide*).

NOTE

The trigger input used in this command must first be defined as a *Trigger Interrupt* input with the INFNCi-H command.

- **"Conditional GO"** Function (aTRGFNc1x): Suspend execution of the next start-motion command until the specified trigger input goes active. Start-motion commands are:
 - GO (standard command to begin motion)
 - GOL (begin linear interpolated motion)
 - FGADV (begin geared advance – for Following motion)
 - FSHFC (begin continuous shift – for Following motion)
 - FSHFD (begin preset shift – for Following motion)

Axis status bit #26 (reported with TASF, TAS, or AS) is set to one (1) when there is a pending "Conditional GO" condition initiated by a TRGFN command; this bit is cleared when the trigger is activated or when a stop command (S) or a kill command (K) is issued. If you need execution to be triggered by other factors (e.g., input state, master position, encoder position, etc.) use the GOWHEN command.

- **"New Master Cycle"** Function (aTRGFNcx1): This is equivalent to executing the FMCNEW command. When the specified trigger input goes active, the controller begins a new Following master cycle. For more information on master cycles, refer to the Following chapter in the *Programmer's Guide*.

These trigger functions are cleared once the function is complete. To use the trigger to perform a GOWHEN function again, the TRGFN command must be given again.

TRGFN in Compiled Motion: When used in a compiled program, a aTRGFNc1xx (GOWHEN function) command will pause the profile in progress (motion continues at constant velocity) until the trigger is

activated to execute the next move profile. When used in a compiled profile, the TRGFN command consumes one segment of compiled memory. When used in a compiled Following profile, the TRGFN command is ignored on the reverse Following profile (i.e., when the master is moving in the opposite direction of that specified in the FOLMAS command).

Trigger Interrupt Status: The status of a trigger interrupt event is reported with the TTRIG and TRIG commands

Example: (refer also to the FOLEN examples)

```
1TRGFNBx1      ; When trigger 1B goes active, axis 1 will begin a
                ; new master cycle
2TRGFNB1x      ; When trigger 2B goes active, axis 2 will execute the move
                ; commanded with the GO command.
GO01           ; The move on axis 2 is commanded, but will not execute until
                ; trigger 2B goes active.
```

TRGLOT Trigger Interrupt Lockout Time

| | | | |
|----------|--|----------------|------------|
| Type | Input | Product | Rev |
| Syntax | <!>TRGLOT<r> | 6K | 5.0 |
| Units | r = time in milliseconds | | |
| Range | 0-250 | | |
| Default | 24 | | |
| Response | TRGLOT: *TRGLOT24 | | |
| See Also | INDEB, INFNC, RE, REG, TIN, TRGFN, [TRIG], TTRIG | | |

The TRGLOT command configures the amount of time in which all “trigger interrupt” inputs (all trigger 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, but prevents subsequent bouncing of the input from causing a false position capture, registration move, or TRGFN event. The lockout time affects those triggers configured as H (trigger interrupt) with the INFNC command during those interrupt actions (registration, position capture, etc.).

The TRGLOT setting overrides the existing INDEB setting for only the trigger inputs that are assigned the “Trigger Interrupt” function.

Example:

```
INFNC1-H      ; Assign trigger 1A as a "trigger interrupt" input
TRGLOT40      ; Set lockout time for all "trigger interrupt" inputs
                ; to be 40 milliseconds
```

[TRIG] Trigger Interrupt Status

| | | | |
|----------|--|----------------|------------|
| Type | Assignment or Comparison | Product | Rev |
| Syntax | See below | 6K | 5.0 |
| Units | n/a | | |
| Range | n/a | | |
| Default | n/a | | |
| Response | n/a | | |
| See Also | INFNC, [PCC], [PCE], [PCME], [PCMS], TAS, TPCC, TPCE, TPCME, TPCMS, TRGFN, TTRIG | | |

Use the TRIG operator to assign the Trigger Interrupt status bits to a binary variable (VARB), or to make a comparison against a binary or hexadecimal value. To make a comparison against a binary value, the letter b (b or B) must be placed in front of the value. The binary value itself must only contain ones, zeros, or Xs (1, Ø, X, x). To make a comparison against a hexadecimal value, the letter h (h or H) must be placed in front of the value. The hexadecimal value itself must only contain the letters A through F, or the numbers Ø through 9.

Syntax: VARBn=TRIG where “n” is the binary variable number, or TRIG can be used in an expression such as IF (TRIG=b11Ø1), or IF (TRIG=h7F)

Each Trigger Interrupt status bit indicates whether a “trigger interrupt” input has been activated to capture a position, initiate a registration move, or execute a TRGFN function. “Trigger Interrupt” inputs are onboard trigger inputs that have been assigned the trigger interrupt function with the `INFNCi-H` command.

Each TTRIG bit is cleared when the captured position value is read with the `PCC`, `PCE`, `PCME`, `PCMS`, `TPCC`, `TPCE`, `TPCME`, or `TPCMS` commands, but the position information is still available from the respective register until it is overwritten by a subsequent position capture by the same trigger input.

The function of each status bit are shown in the table below (bits are numbered from left to right). A bit that is set (“1”) indicated the trigger interrupt has occurred, a “0” indicates no trigger interrupt.

| TTRIG bit # | Trigger Input (Axis 1-4 “Triggers/Outputs” connector) * | Dedicated Axis | TTRIG bit # | Trigger Input (Axis 5-8 “Triggers/Outputs” connector) * | Dedicated Axis |
|-------------|---|----------------|-------------|---|----------------|
| 1 | Pin 23, Trigger 1A | 1 | 9 | Pin 23, Trigger 5A | 5 |
| 2 | Pin 21, Trigger 1B | 1 | 10 | Pin 21, Trigger 5B | 5 |
| 3 | Pin 19, Trigger 2A | 2 | 11 | Pin 19, Trigger 6A | 6 |
| 4 | Pin 17, Trigger 2B | 2 | 12 | Pin 17, Trigger 6B | 6 |
| 5 | Pin 15, Trigger 3A | 3 | 13 | Pin 15, Trigger 7A | 7 |
| 6 | Pin 13, Trigger 3B | 3 | 14 | Pin 13, Trigger 7B | 7 |
| 7 | Pin 11, Trigger 4A | 4 | 16 | Pin 11, Trigger 8A | 8 |
| 8 | Pin 9, Trigger 4B | 4 | 16 | Pin 9, Trigger 8B | 8 |
| | | | 17 | Master Trigger (“TRG-M”) | Master Encoder |

* The number of trigger inputs available varies by product (refer to your product’s *Installation Guide*).

TSCAN Transfer Scan Time of PLCP Program

| | | | |
|----------|--|----------------|------------|
| Type | Transfer; PLC Scan Program | Product | Rev |
| Syntax | <!>TSCAN | 6K | 5.0 |
| Units | Response is in increments of 2 milliseconds | | |
| Range | 2 ms - unlimited | | |
| Default | n/a | | |
| Response | TSCAN *TSCAN4 (4 ms corresponds to 2 system updates) | | |
| See Also | SCANP, PLCP, EXE | | |

The TSCAN command reports the duration it takes the last PLCP program to be scanned completely. A compiled PLCP program is launched into Scan mode using the SCANP command. During each 2 ms system update, the PLCP program is scanned an allotted 0.5 ms window. If the PLCP program requires more than 0.5 ms to be scanned, the program will be paused and then resumed at the next system update. The value reported by the TSCAN command is in multiples of the 2 ms system update period.

Example:

```
SCANP PLCP1 ; Start execution of compiled PLCP program PLCP1 in Scan mode
TSCAN      ; Report the duration of the scan program in multiples of the
           ; 2 millisecond system update period. An example response might
           ; be *TSCAN4 (indicates that 2 system update periods, a total of
           ; 4 milliseconds was required to scan the last PLCP program).
```

TSEG Transfer Number of Free Segment Buffers

| | | | |
|----------|--|----------------|------------|
| Type | Compiled Motion; Transfer | Product | Rev |
| Syntax | <!>TSEG | 6K | 5.0 |
| Units | n/a | | |
| Range | n/a | | |
| Default | n/a | | |
| Response | TSEG: *TSEG258 | | |
| See Also | MEMORY, TDIR, TMEM, [SEG], [SS], TSS, TSSF | | |

The Transfer Number of Free Segment Buffers (TSEG) command returns the number of free segment buffers in compiled memory.

System status bit (see TSSF, TSS, and SS) 29 to set when the compiled memory is 75% full, and bit 30 is set if the compiled memory is 100% full.

TSGSET Transfer Servo Gain Set

| | | | |
|----------|--|----------------|----------------------------------|
| Type | Transfer | Product | Rev |
| Syntax | <!>TSGSETi | 6K | 5.0 |
| Units | i = gain set identification number (see SGSET command) | | |
| Range | 1-5 | | (application to servo axes only) |
| Default | n/a | | |
| Response | (see examples below) | | |
| See Also | SFB, SGAF, SGENB, SGI, SGILIM, SGP, SGSET, SGV, SGVF, SOFFS, TGAIN | | |

This command allows you to display any of the 5 gain sets that you saved with the SGSET command. Up to 5 gain sets can be saved.

| |
|-------------|
| NOTE |
|-------------|

The tuning gains in a given gain set are specific to the feedback source that was in use (selected with the last SFB command) at the time the gains were established with the respective gain commands (SGI, SGP, etc.).

Example:

```
SGP5,5,10,10      ; Set the gain for the proportional gain
SGI.1,.1,0,0      ; Set the gain for the integral gain
SGV50,60,0,0      ; Set the gain for the velocity gain
SGVF5,6,10,11     ; Set the gain for the velocity feedforward gain
SGAF0,0,0,0       ; Set the gain for the acceleration feedforward gain
SGSET3            ; Assign the SGP, SGI, SGV, SGVF, & SGAF gains to servo gain set 3
SGP75,75,40,40    ; Set the gain for the proportional gain
SGI5,5,5,7        ; Set the gain for the integral gain
SGV1,.45,2,2      ; Set the gain for the velocity gain
SGVF0,8,0,9       ; Set the gain for the velocity feedforward gain
SGAF18,20,22,24   ; Set the gain for the acceleration feedforward gain
SGSET1            ; Assign the SGP, SGI, SGV, SGVF, & SGAF gains to servo gain set 1
SGENB1,3,3,1      ; Enable gain set 1 on axis 1 & 4 and enables gain set 3 on
                  ; axis 2 & 3
TSGSET1           ; Display gain set 1.  Response should be:
                  ; *SGP75,75,40,40
                  ; *SGI5,5,5,7
                  ; *SGV1,.45,2,2
                  ; *SGVF0,8,0,9
                  ; *SGAF18,20,22,24
TSGSET3           ; Display gain set 3.  Response should be:
                  ; *SGP5,5,10,10
                  ; *SGI.1,.1,0,0
                  ; *SGV50,60,0,0
                  ; *SGVF5,6,10,11
                  ; *SGAF0,0,0,0
```

TSKAX Task Axis

| | | | |
|----------|---|----------------|------------|
| Type | Multi-Tasking | Product | Rev |
| Syntax | i%TASKAX<a1>,<a2> | 6K | 5.0 |
| Units | i = task number a1 = first axis associated with the task a2 = last axis associated with the task Range of axes from a1 - a2, where a1 ≤ a2 | | |
| Range | For i, 0-10 For a1 and a2, 0-n, where n = number of axes on the product | | |
| Default | a1 = 1 a2 = n | | |
| Response | n/a | | |
| See Also | %, TTASK, [TASK], TSWAP, [SWAP], TSKTRN | | |

The Task Axis command (TSKAX) allows you to specify the axes associated with a task. The default condition in multi-tasking is that each task is associated with all controller axes. This means, for example, that when an axis being used in a task hits an end-of-travel limit, program execution will be killed within that task, and in all other tasks, because they all share that axis. The TSKAX command is used to assign a

set of axes to a given task to allow a multi-axis controller to be used as more than one independent program execution environment.

The `TSKAX` command allows you to assign axes to specific tasks, thus constraining task response and control to a smaller set of axes. A task is allowed to control only its associated axes. This axis association covers all interaction between axes commands, conditions or inputs and task program flow. For example, if a 6K controller is controlling two independent machines that do not share common axes, with control of each machine as a separate task, a limit hit by an axis in one machine can kill the task running that machine, but will not kill the task running the other machine.

The `TSKAX` command allows you to specify the first and last axis numbers associated with the task. Thus, the axes associated with a task will always be consecutive. As a demonstration, the `TSKAX` commands in the example below will associate axes 1, 2 and 3 with Task1, axes 4, 5 and 6 with Task2, and axes 7 and 8 with Task3. If axis 3 in Task1 hits a limit, program execution in Task1 will be killed, but Task2 and Task3 can continue to run because they are independent and do not share axis 3. Task1 may change motion parameters and start motion on only axes 1, 2, and 3.

Example:

```
DEF main          ; Begin definition of program called "main"
1%TSKAX1,3       ; Associate axes 1, 2 and 3 to Task1
2%TSKAX4,6       ; Associate axes 4, 5 and 6 to Task2
3%TSKAX7,8       ; Associate axes 7 and 8 to Task3
1%move1         ; Execute stored program "move1" in Task1
2%inout         ; Execute stored program "inout" in Task2
3%fill          ; Execute stored program "fill" in Task3
END
```

It is also possible to eliminate axis association for a task altogether with the `TSKAX0,0` command. This would be appropriate for a task that is not involved in motion control, but may control I/O or start other tasks.

TSKTRN

Task Turns Before Swapping

| | | | |
|----------|---|----------------|------------|
| Type | Multi-Tasking | Product | Rev |
| Syntax | <code>i₁%TSKTRNi₂</code> | 6K | 5.0 |
| Units | <code>i₁</code> = task number <code>i₂</code> = number of turns before task swap | | |
| Range | <code>i₁</code> = 0-10 <code>i₂</code> = 0-10,000 | | |
| Default | <code>i₁</code> = 0 <code>i₂</code> = 1 | | |
| Response | n/a | | |
| See Also | <code>%, LOCK, TTASK, [TASK], TSWAP, [SWAP], TSKAX</code> | | |

Use the `TSKTRN` command to set the relative amount of processing time a task will get. Under default multi-tasking operation, all active tasks have an equal share of processing time; that is, each task executes one “turn” and then “swaps” control to the next active task. (A “turn” is the execution of a command, or a portion of a complex command such as those for contouring and math and trig operators.)

For example, if Task2 issued a `TSKTRN6` command, while the other tasks stayed at `TSKTRN1`, Task2 would execute 6 commands (or portions of long commands) before relinquishing control to another task.

The `TSKTRN` value for a task may be changed at any time, allowing a task to increase its weight for an isolated section of program commands.

TSS

Transfer System Status

| | | | |
|----------|--|----------------|------------|
| Type | Transfer | Product | Rev |
| Syntax | <!><%>TSS<.i> | 6K | 5.0 |
| Units | i = system status bit number | | |
| Range | 1-32 | | |
| Default | n/a | | |
| Response | TSS: *TSS1000_1000_0000_0000_0000_0000_0000_0000 TSS.1: *1 (status of Task 0 status bit #1—system is ready) | | |
| See Also | PORT, TAS, TCMDER, TRGFN, TSTAT, [TRIG], TTRIG | | |

The Transfer System Status (TSS) command provides information on the 32 system status bits. The TSS status command reports a binary bit report. If you would like to see a more descriptive text-based report, use the TSSF command description.

Response for TSS (b can equal Ø, 1, X, or x): *TSSbbbb_bbbb_bbbb_bbbb_bbbb_bbbb_bbbb_bbbb
^ ^
 Bit #1 Bit #32

MULTI-TASKING

If you are using multi-tasking, be aware that each task has its own system status register. Therefore, to check a specific task's system status, you must prefix the TSS command with the task identifier (e.g., 2%TSS to check system status for Task 2). If no task identifier is given, the TSS response is for the task supervisor (Task 0).

| BIT (Left to Right) | Function (1 = yes, Ø = no) | BIT (Left to Right) | Function (1 = yes, Ø = no) |
|---------------------|---|---------------------|---|
| 1 | System Ready (fully powered up and ready to receive commands) | 17 | Loading Thumbwheel Data ([TW]) |
| 2 | Reserved | 18 | External Program Select Mode (INSELP) |
| 3 | Executing a Program | 19 | Dwell in Progress (T command) |
| 4 | Immediate Command (set if last command was immediate) | 20 | Waiting for RP240 Data—[DREAD] or [DREADF] |
| 5 | In ASCII Mode | 21 | RP240 Connected— current PORT setting only |
| 6 | In Echo Mode — current PORT setting only | 22 | Non-volatile Memory Error |
| 7 | Defining a Program | 23 | Servo data gathering transmission in progress (servo axes only) |
| 8 | In Trace Mode | 24 | Reserved |
| 9 | In Step Mode | 25 | RESERVED |
| 10 | In Translation Mode (must use fast status area to see) | 26 | RESERVED |
| 11 | Command Error Occurred (bit is cleared when TCMDER is issued) | 27 | RESERVED |
| 12 | Break Point Active (BP) | 28 | RESERVED |
| 13 | Pause Active | 29 | Compiled memory is 75% full |
| 14 | Wait Active (WAIT) | 30 | Compiled memory is 100% full |
| 15 | Monitoring On Condition (ONCOND) | 31 * | Compile operation failed (PCOMP) ** |
| 16 | Waiting for Data (READ) | 32 | RESERVED |

* Bit #31: failed PCOMP compile is cleared on power up, RESET, or after successful compile. Possible causes include:

- Errors in profile design (e.g., change direction while at non-zero velocity; distance & velocity equate to < 1 count per system update; preset move profile ends in non-zero velocity)
- Profile will cause a Following error (see TFSF, TFS, or FS command descriptions)
- Out of memory (see TSS bit #30)
- Axis already in motion at the time of the PCOMP command
- Loop programming errors (e.g., no matching PLOOP or PLN; more than 4 embedded PLOOP/END loops)
- PLCP program contains invalid commands.

TSSF

Transfer System Status (full-text report)

| | | | |
|----------|---|----------------|------------|
| Type | Transfer | Product | Rev |
| Syntax | <!><%>TSSF | 6K | 5.0 |
| Units | n/a | | |
| Range | n/a | | |
| Default | n/a | | |
| Response | TSSF: (see example below) | | |
| See Also | PORT, [SS], TAS, TCMER, TRGFN, TSS, TSTAT | | |

The TSSF command returns a text-based status report of all axes. This is an alternative to the binary report (TSS).

MULTI-TASKING

If you are using multi-tasking, be aware that each task has its own system status register. Therefore, to check a specific task's system status, you must prefix the TSSF command with the task identifier (e.g., 2%TSSF to check system status for Task 2). If no task identifier is given, the TSSF response is for the task supervisor (Task 0).

Example TSSF response:

```
*TSSF
*System Ready          YES      Thumbwhl Data Load  NO
*RESERVED              NO       Ext Prog Sel Mode   NO
*Program Executing    NO       Time Command        NO
*Immediate Comm Last  NO       Waiting RP240 Data  NO
*
*ASCII Mode           YES      RP240 Connected     NO
*Echo Mode            YES      Memory Error        NO
*Defining a Program   NO       Servo Data Transfer NO
*Trace Mode           NO       RESERVED            NO
*
*Step Mode            NO       RESERVED            NO
*FS Translate Mode    NO       RESERVED            NO
*Command Error        NO       RESERVED            NO
*Break Point Active   NO       RESERVED            NO
*
*Pause Active         NO       Comp Mem Near Full  NO
*Wait Active          NO       Compiled Mem Full   NO
*Checking On Conds    NO       Compile Failed      NO
*Waiting for Data     NO       Reserved            NO
```

TSTAT Transfer Statistics

| | | | |
|----------|---|---------|-----|
| Type | Transfer | Product | Rev |
| Syntax | <!>TSTAT | 6K | 5.0 |
| Units | n/a | | |
| Range | n/a | | |
| Default | n/a | | |
| Response | TSTAT: (See below) | | |
| See Also | NTADDR, TAS, TDIR, TER, TFB, TIN, TIO, TLIM, TOUT, TPC, TPE, TREV, TSKAX, TSWAP, TSS, TTIM, TUS, TVEL | | |

The following is an example (**NOTE:** The response for each 6K Series product will vary slightly.):

```
*6K8 (8-axis controller)
*6K revision: 92-XXXXXX-01-5.0 6K 92-XXXXXX-XX-NOP2.5 DSP
*Ethernet address: xxxxxxxxxxxx; IP address: 192.168.10.30
*Axis definition: Servo,Servo,Servo,Servo,Stepper,Stepper,Stepper,Stepper
*Power-up program assignment (STARTP): SETUP
*ENABLE input OK: Yes
*Drive status (DRIVE): 0000_0000
*Drive Fault input states (ASX.4 for each axis): 0000_0000
*Drive Fault input checking - enabled (DRFEN1): 0000_0000
*Drive resolution (DRES): -, -, -, -, 25000, 25000, 25000, 25000
*Encoder resolution (ERES): 4000, 4000, 4000, 4000
*Encoder failure detection enabled (EFAIL1): 0000_0000
*Hard Limit enable: LH3, 3, 3, 3, 3, 3, 3, 3
*Soft Limit enable: LS0, 0, 0, 0, 0, 0, 0, 0
*Current Motion Attributes:
* Scaling enabled (SCALE1): 0
* Acceleration scaler (SCLA): 4000, 4000, 4000, 4000, 4000, 4000, 4000, 4000
* Distance scaler (SCLD): 1, 1, 1, 1, 1, 1, 1, 1
* Velocity scaler (SCLV): 4000, 4000, 4000, 4000, 4000, 4000, 4000, 4000
* Continuous/Preset (MCL/MC0) positioning mode: 0, 0, 0, 0, 0, 0, 0, 0
* Absolute/Incremental (MAL/MA0) positioning mode: 0, 0, 0, 0, 0, 0, 0, 0
* Feedback position (TFB or TPE): +0, +0, +0, +0, -, -, -, -
* Commanded position (TPC): +0, +0, +0, +0, +0, +0, +0, +0
* A10.0000, 10.0000, 10.0000, 10.0000, 10.0000, 10.0000, 10.0000, 10.0000
* AA10.0000, 10.0000, 10.0000, 10.0000, 10.0000, 10.0000, 10.0000, 10.0000
* AD10.0000, 10.0000, 10.0000, 10.0000, 10.0000, 10.0000, 10.0000, 10.0000
* ADA10.0000, 10.0000, 10.0000, 10.0000, 10.0000, 10.0000, 10.0000, 10.0000
* V1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000
* D+4000, +4000, +4000, +4000, +4000, +4000, +4000, +4000
*I/O Status:
* Onboard limit inputs:
* Hardware state (TLIM): 000_000_000_000_000_000_000_000
* Prog. function (LIMFNC): RST_RST_RST_RST_RST_RST_RST_RST
* Onboard trigger inputs:
* Hardware state (TIN): 0000_0000_0000_0000_0
* Prog. function (INFNC): AAAA_AAAA_AAAA_AAAA_A
* Onboard digital outputs:
* Hardware state (TOUT): 000_000
* Prog. function (OUTFNC): AAA_AAA
* Expansion I/O bricks: See TIO response
*Axis Status (see TASF for full text report of all axes):
* Axis 1 (1TAS): 0010_0000_0000_1000_0000_0001_0000_0000
* Axis 2 (2TAS): 0010_0000_0000_1000_0000_0001_0000_0000
* Axis 3 (3TAS): 0010_0000_0000_1000_0000_0001_0000_0000
* Axis 4 (4TAS): 0010_0000_0000_1000_0000_0001_0000_0000
* Axis 5 (5TAS): 0010_0000_0000_1000_0000_0001_0000_0000
* Axis 6 (6TAS): 0010_0000_0000_1000_0000_0001_0000_0000
* Axis 7 (7TAS): 0010_0000_0000_1000_0000_0001_0000_0000
* Axis 8 (8TAS): 0010_0000_0000_1000_0000_0001_0000_0000
*System Status (This is Task 0 status if using multi-tasking.):
* Assoc. axes (TSKAX): 1, 2, 3, 4, 5, 6, 7, 8
* System status (TSSF): 1000_1100_0000_0000_0000_0100_0000_0000
* Error checking (ERROR): 1000_0100_1000_0001_0000_0000_0000_0000
* Error status (TERF): 0000_0000_0000_0000_0000_0000_0000_0000
* Error program (ERRORP): ERRPRG
* On conditions (ONCOND): 0000
*Multi-Tasking Status:
* Currently active tasks (TSWAP): 1100_0000_00
* Task 1:
* Assoc. axes (1%TSKAX): 1, 2, 3, 4
* System status (1%TSSF): 1000_1100_0000_0000_0000_0100_0000_0000
* Error checking (1%ERROR): 1000_0100_1000_0001_0000_0000_0000_0000
* Error status (1%TERF): 0000_0000_0000_0000_0000_0000_0000_0000
* Error program (1%ERRORP): ERRPRG
* On conditions (1%ONCOND): 0000
* Task 2:
* Assoc. axes (2%TSKAX): 5, 6, 7, 8
* System status (2%TSSF): 1000_1100_0000_0000_0000_0100_0000_0000
* Error checking (2%ERROR): 1000_0100_1000_0001_0000_0000_0000_0000
* Error status (2%TERF): 0000_0000_0000_0000_0000_0000_0000_0000
* Error program (2%ERRORP): ERRPRG
* On conditions (2%ONCOND): 0000
*Following Conditions:
* Master-Follower assignment (FOLMAS): +0, +0, +0, +0, +0, +0, +0, +0
* Master scaling (SCLMAS): 4000, 4000, 4000, 4000, 4000, 4000, 4000, 4000
* Following status (TFSF): 0000_0000_0000_0000_0000_0000_0000_0000
```

TSTLT Transfer Settling Time

| | | | |
|----------|--|----------------|---------------------------------|
| Type | Transfer | Product | Rev |
| Syntax | <!><a>TSTLT | 6K | 5.0 |
| Units | Reported value represents milliseconds | | |
| Range | n/a | | (applicable only to servo axes) |
| Default | n/a | | |
| Response | TSTLT: *TSTLT502,483,344,249,299,443,534,674 1TSTLT: *1TSTLT502 | | |
| See Also | STRGTD, STRGTE, STRGTT, STRGTV | | |

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

*** For a more information on target zone operation, refer to the *Programmer's Guide*.

TSWAP Transfer Current Active Tasks

| | | | |
|----------|---|----------------|------------|
| Type | Transfer | Product | Rev |
| Syntax | <!>TSWAP | 6K | 5.0 |
| Units | n/a | | |
| Range | Binary response status of tasks (0 = inactive, 1 = inactive). 10-bit pattern represents tasks 1-10 from left to right. | | |
| Default | n/a | | |
| Response | TSWAP: *TSWAP1001_0000_00 (tasks 1 and 4 are active) TSWAP.3: *0 (task 3 is inactive) | | |
| See Also | %, [SS], [TASK], TSKAX, TSS | | |

The Transfer Task Swap command (TSWAP) reports a binary bit pattern indicating the tasks that are currently active. Note that TSWAP only indicates of a task is active; to ascertain exactly what activity the task has at a given time, use the system status (SS or TSS commands).

TSWAP's binary 10-bit pattern represents tasks 1-10, from left to right. A "1" indicates that the task is active, and a "0" indicates that the task is inactive. To check the status of only one task, you may use the bit select (.) operator. For example, TSWAP.3 checks the status of Task3 only.

The "Task Supervisor", represented by task Ø, is always active and is therefore not included in the SWAP and TSWAP status.

TTASK Transfer Task Number

| | | | |
|----------|---|----------------|------------|
| Type | Transfer | Product | Rev |
| Syntax | <!>TTASK | 6K | 5.0 |
| Units | Reported value is the number of the controlling task. | | |
| Range | 0-10 | | |
| Default | n/a | | |
| Response | TTASK: *TTASK2 (Task 2 executed the TTASK command) | | |
| See Also | %, [TASK] | | |

Use the TTASK command to the display the task number of the task which executed the command. This could be used for diagnostic purposes, as a way to indicate which task is executing a given section of program.

TTIM Transfer Timer

| | | | |
|----------|--|---------|-----|
| Type | Transfer | Product | Rev |
| Syntax | <!><%>TTIM | 6K | 5.0 |
| Units | Reported value represents milliseconds | | |
| Range | Maximum count is 999,999,999 (approx. 11 days, 13 hours) | | |
| Default | n/a | | |
| Response | TTIM: *TTIM64000 | | |
| See Also | T, [TIM], TIMINT, TIMST, TIMSTP | | |

The Transfer Timer (TTIM) command returns the current value of the timer in milliseconds. The timer is started with the TIMST command, and stopped with the TIMSTP command.

Multi-Tasking: Each task has its own timer.

TTRIG Transfer Trigger Interrupt Status

| | | | |
|----------|---|---------|-----|
| Type | Transfer, Inputs | Product | Rev |
| Syntax | <!>TTRIG | 6K | 5.0 |
| Units | n/a | | |
| Range | n/a | | |
| Default | n/a | | |
| Response | TTRIG *TTRIG0000_0000_0000_0000_0 | | |
| See Also | INFNC, [PCC], [PCE], [PCME], [PCMS], TAS, TPCC, TPCE, TPCME, TPCMS, TRGFN, [TRIG] | | |

Use the TTRIG command to check whether a “trigger interrupt” input has been activated to capture a position, initiate a registration move, or execute a TRGFN function. “Trigger Interrupt” inputs are onboard trigger inputs that have been assigned the trigger interrupt function with the INFNCi-H command.

Each TTRIG bit is cleared when the captured position value is read with the PCC, PCE, PCME, PCMS, TPCC, TPCE, TPCME, or TPCMS commands, but the position information is still available from the respective register until it is overwritten by a subsequent position capture by the same trigger input.

The functions of each bit in the binary report are shown in the table below (bits are numbered from left to right). A bit that is set (“1”) indicated the trigger interrupt has occurred, a “0” indicates no trigger interrupt.

| TTRIG bit # | Trigger Input (Axis 1-4 “Triggers/Outputs” connector) * | Dedicated Axis | TTRIG bit # | Trigger Input (Axis 5-8 “Triggers/Outputs” connector) * | Dedicated Axis |
|-------------|---|----------------|-------------|---|----------------|
| 1 | Pin 23, Trigger 1A | 1 | 9 | Pin 23, Trigger 5A | 5 |
| 2 | Pin 21, Trigger 1B | 1 | 10 | Pin 21, Trigger 5B | 5 |
| 3 | Pin 19, Trigger 2A | 2 | 11 | Pin 19, Trigger 6A | 6 |
| 4 | Pin 17, Trigger 2B | 2 | 12 | Pin 17, Trigger 6B | 6 |
| 5 | Pin 15, Trigger 3A | 3 | 13 | Pin 15, Trigger 7A | 7 |
| 6 | Pin 13, Trigger 3B | 3 | 14 | Pin 13, Trigger 7B | 7 |
| 7 | Pin 11, Trigger 4A | 4 | 16 | Pin 11, Trigger 8A | 8 |
| 8 | Pin 9, Trigger 4B | 4 | 16 | Pin 9, Trigger 8B | 8 |
| | | | 17 | Master Trigger (“TRG-M”) | Master Encoder |

* The number of trigger inputs available varies by product (refer to your product’s *Installation Guide*).

Example

```
COMEXC1 ; Continuous command execution
INFNC1-H ; Define trigger 1A is a trigger interrupt input
1REGA10000 ; Registration move on axis 1 on trigger 1A event
WAIT(TRIG.1=B1) ; Wait for trigger 1A event to occur
WRITE"TRIGGER 1A OCCURRED" ; Display message
TTRIG ; Get report back (display to monitor)
; response should be: *TTRIG1000_0000_0000_0000_0
```

TUS Transfer User Status

| | | | |
|----------|---|----------------|------------|
| Type | Transfer | Product | Rev |
| Syntax | <!>TUS<.i> | 6K | 5.0 |
| Units | i = user status bit number | | |
| Range | 1 - 16 | | |
| Default | n/a | | |
| Response | TUS: *TUS1111_0000_1111_0000 TUS.4: *1 (user status bit 4 is reported) | | |
| See Also | INDUSE, INDUST, [US] | | |

The Transfer User Status (TUS) command returns the current bit pattern for the user status word. All 16 bits of the user status word are defined with the INDUST command. Each bit can correspond to an axis status bit, a system status bit, or an input.

Example:

```
INDUSE1          ; Enable user status
INDUST1-5A       ; User status bit 1 defined as axis 1 status bit 5
INDUST2-3F       ; User status bit 2 defined as axis 6 status bit 3
3INDUST3-5J      ; User status bit 3 defined as input 5 on I/O brick 3
INDUST4-1K       ; User status bit 4 defined as interrupt status bit 1
2%INDUST16-2I    ; User status bit 16 defined as system status bit 2 for task 2
TUS              ; Return the state of the user status word
```

TVEL Transfer Current Commanded Velocity

| | | | |
|----------|---|----------------|------------|
| Type | Transfer | Product | Rev |
| Syntax | <!><a>TVEL | 6K | 5.0 |
| Units | Reported value is in units/sec (scalable by SCLV) | | |
| Range | n/a | | |
| Default | n/a | | |
| Response | TVEL: *TVEL23.3450,23.0000,45.7800,456.7800 ... 1TVEL: *1TVEL23.3450 | | |
| See Also | ERES, SCALE, SCLV, TVELA, V, [VEL] | | |

The TVEL value represents the current commanded velocity. It is not the programmed velocity (v). If scaling is enabled (SCALE1), the TVEL value is scaled by the velocity scaling factor (SCLV).

Stepper Axes: If scaling is disabled (SCALE \emptyset), the value is measured in revolutions/sec (actual velocity in commanded counts/sec divided by the drive resolution DRES value).

Servo Axes: If scaling is disabled (SCALE \emptyset), the value is measured in encoder revs/sec or ANI volts/sec.

TVELA Transfer Current Actual Velocity

| | | | |
|----------|--|----------------|------------|
| Type | Transfer | Product | Rev |
| Syntax | <!><a>TVELA | 6K | 5.0 |
| Units | Reported value is in units/sec | | |
| Range | n/a | | |
| Default | n/a | | |
| Response | TVELA: *TVELA+1.55,-3.25,-5.55,+2.30 1TVELA: *TVELA+1.55 | | |
| See Also | ENCCNT, SCALE, SCLV, SFB, TVEL, V, [VEL], [VELA] | | |

The Transfer Current Actual Velocity (TVELA) command reports the current velocity as derived from the feedback device. The sign determines the direction of motion. You can use the TVELA command at all times; therefore, even if no motion is being commanded, TVELA will still report a non-zero value as it detects the servoing action.

Units of Measure:

Steppers: The velocity is always revs/sec (actual velocity in counts/sec multiplied by the ERES value if in ENCCNT1 mode, or multiplied by DRES if in ENCCNT0 mode).

Servos: If scaling is enabled (SCALE1), the velocity value will be scaled by the velocity scaling factor (SCLV). If scaling is not enabled (SCALE0), the value returned will be in encoder revs/sec or ANI volts/sec.

Example:

```
TVELA                    ; Reports the current actual velocity; since no motion is  
                         ; commanded, the servoing velocities are reported.  
                         ; Example response is:    *TVELA+0.0097,-0.0027,+0.0103,-0.0044
```

TVMAS Transfer Current Master Velocity

| | | | |
|----------|---|----------------|------------|
| Type | Following and Transfer | Product | Rev |
| Syntax | <!><a>TVMAS | 6K | 5.0 |
| Units | n/a | | |
| Range | n/a | | |
| Default | n/a | | |
| Response | TVMAS *TVMAS+0,+0,+0,+0,+0,+0,+0,+0 1TVMAS *1TVMAS0 | | |
| See Also | FFILT, FOLMAS, SCALE, SCLMAS, V, [VMAS] | | |

The TVMAS command transfers the current velocity of the master. The master must be assigned first (FOLMAS command) before this command will be useful.

The precision of the reported TVMAS value is dependent upon the FFILT filter value (details are provided in the “Master Position Filtering” section in the Following chapter of the *Programmer's Guide*).

If scaling is enabled (SCALE1), the value returned is scaled by the master scaling factor (SCLMAS). If scaling is disabled (SCALE0), the value returned is in counts/sec.

[TW]**Thumbwheel Assignment**

| | | | |
|----------|---|----------------|------------|
| Type | Assignment or Comparison | Product | Rev |
| Syntax | TWi (See below for examples) | 6K | 5.0 |
| Units | i = sets used by INPLC, INSTW, OUTPLC and OUTTW | | |
| Range | 1-8 | | |
| Default | n/a | | |
| Response | n/a | | |
| See Also | INPLC, INSTW, OUTPLC, OUTTW, [SS], TSS | | |

The Thumbwheel Assignment (TW) command, executed from within another command, reads data from a parallel device and loads it into the command field the TW command is occupying. Rule of Thumb for command value substitutions: If the command syntax shows that the command field requires a real number (denoted by <r>) or an integer value (denoted by <i>), you can use the TW substitution (e.g., V2, (TW)).

The value of the TW command designates which input and output set to use. TW values 1-4 correspond to INSTW and OUTTW sets 1 - 4, respectively. TW values 5-8 correspond to INPLC and OUTPLC sets 1 - 4, respectively.

The TW command can be used as a variable assignment (VAR1=TW2) or in another command (e.g., A10, (TW2), 10, 1). However, the TW command cannot be used in an expression such as VAR4=1 + TW2 or IF(TW2<8).

For more information on interfacing thumbwheels, refer to your product's *Installation Guide*.

Example:

```
INSTW2,1-4,5      ; Set INSTW set 2 as BCD digits on onboard inputs 1-4, with
                  ; input 5 as the sign bit
OUTTW2,1-3,4,50   ; Set OUTTW set 2 as output strobes on onboard outputs 1-3,
                  ; with output 4 as the output enable bit, and strobe time
                  ; of 50 milliseconds
A(TW2)            ; Read data into axis 1 acceleration using INSTW set 2 and
                  ; OUTTW set 2 as the data configuration
```