

CHAPTER EIGHT

# Troubleshooting

## IN THIS CHAPTER

• Troubleshooting basics .....	222
• Solutions to common problems (problem/cause/remedy table).....	222
• Program debug tools	
- Status commands .....	226
- Error messages .....	232
- Trace mode.....	236
- Single-step mode.....	237
- Simulating programmable I/O activation.....	238
- Simulating analog input activation.....	240
- Motion Planner's Panel Gallery .....	240
• Technical support.....	241
• Operating system upgrades .....	241
• Product return procedure .....	241

# Troubleshooting Basics

---

When your system does not function properly (or as you expect it to operate), the first thing that you must do is identify and isolate the problem. When you have accomplished this, you can effectively begin to resolve the problem.

The first step is to isolate each system component and ensure that each component functions properly when it is run independently. You can have to dismantle your system and put it back together piece by piece to detect the problem. If you have additional units available, you can want to exchange them with existing components in your system to help identify the source of the problem.

Determine if the problem is mechanical, electrical, or software-related. Can you repeat or recreate the problem? Random events can appear to be related, but they are not necessarily contributing factors to your problem. You can be experiencing more than one problem. You must isolate and solve one problem at a time.

Log (document) all testing and problem isolation procedures. Also, if you are having difficulty isolating a problem, be sure to document all occurrences of the problem along with as much specific information as possible. You can need to review and consult these notes later. This will also prevent you from duplicating your testing efforts.

Once you isolate the problem, see the problem solutions contained in this chapter. If the problem persists, contact your local technical support resource (see *Technical Support* below).

## Electrical Noise

If you suspect that the problems are caused by electrical noise, see your 6K product's *Installation Guide* for help.

# Solutions to Common Problems

---

## NOTES

- Some hardware-related causes are provided because it is sometimes difficult to identify a problem as either hardware or software related.
- Refer to other sections of this manual for more information on controller programming guidelines, system set up, and general feature implementation. You can also see the command descriptions in the *6K Series Command Reference*. Refer to your product's *Installation Guide* for hardware-related issues.

Problem	Cause	Solution
Communication (Ethernet) errors.	1. Ethernet card not installed correctly.	1. Refer to the user instructions that came with your Ethernet card.
	2. Ethernet IP address conflict.	2. Change IP address with the <code>NTADDR</code> command.
	3. Connection to Ethernet port is compromised or miswired.	3. Refer to the connection instructions in the <i>Installation Guide</i> .
Communication (serial) not operative, or receive garbled characters.	1. Improper interface connections or communication protocol.	1. See troubleshooting section in your product's <i>Installation Guide</i> .
	2. COM port disabled.	2.a. Enable serial communication with the <code>E1</code> command. 2.b. If using RS-485, make sure the internal jumpers are set accordingly (see <i>Installation Guide</i> ). Make sure COM 2 port is enabled for sending 6K language commands (execute the <code>PORT2</code> and <code>DRPCHKØ</code> commands).
	3. In daisy chain, unit cannot be set to proper address.	3. Verify DIP switch settings (see <i>Installation Guide</i> ), verify proper application of the <code>ADDR</code> command.

Problem	Cause	Solution
Direction is reversed. (stepper axes only)	1. Direction connections to the drive are reversed.	1. Switch DIR+ with DIR- connection to drive.
	2. Phase of step motor reversed (motor does not move in the commanded direction).	2. Switch A+ with A- connection from drive to motor. SOFTWARE ALTERNATIVE: If the motor (and the encoder if one is used) is reversed, use the <code>CMDDIR1</code> command to reverse the polarity of both the commanded direction and the polarity of the encoder counts).
	3. Phase of encoder reversed (reported <code>TPE</code> direction is reversed).	3. Swap the A+ and A- connection at the ENCODER connector.
Direction is reversed, servo condition is stable. (servo axes only)	1. Command output (CMD) connections and feedback device connections or mounting are reversed.	1. Software remedy: Issue the <code>CMDDIR1</code> command to the affected axis. This reverses the polarity of the commanded direction and the feedback direction so that servo stability is maintained.  Hardware remedy: Switch CMD- with the CMD+ connection to drive or valve (if your drive or valve does not accept differential outputs this will not work). You will also have to change the feedback device wiring or mounting so that it counts in same direction as the commanded direction.
Direction is reversed, servo condition is <u>un</u> stable. (servo axes only)	1. Not tuned properly.	1. Refer to tuning instructions in your product's <i>Installation Guide</i> .
	2. Phase of encoder reversed or mounting of ANI input is such that it counts in the opposite direction as the commanded direction.	2. Software remedy for encoder feedback only: For the affected axis, issue <code>ENCPOL1</code> .  Hardware remedy: If using encoder feedback, swap the A+ and A- connections to the 6K product. If using ANI feedback, change the mounting so that the counting direction is reversed.
Distance, velocity, and accel are incorrect as programmed.	1. Incorrect resolution setting.	1.a. Stepper axes: Set the resolution on the to match the 6K product's <code>DRES</code> command setting (default <code>DRES</code> setting is 4,000 steps/rev).  1.b. Match the 6K product's <code>ERES</code> command setting (default <code>ERES</code> setting is 4,000 counts/rev) to match the post-quadrature resolution of the encoder.  <u>ERES values for Compumotor encoders:</u> Stepper axes: <ul style="list-style-type: none"> <li>• RE, -RC, -EC, &amp; -E Series Encoders:..... <code>ERES4000</code></li> <li>• HJ Series Encoders:..... <code>ERES2048</code></li> </ul> Servo axes (SM, N or J Series Servo Motors): <ul style="list-style-type: none"> <li>• SM/N/JxxxxD-xxxx:..... <code>ERES2000</code></li> <li>• SM/N/JxxxxE-xxxx:..... <code>ERES4000</code></li> </ul> Dynaserv (stepper and servo): <ul style="list-style-type: none"> <li>• DR10xxB ..... <code>ERES507904</code></li> <li>• DR1xxxE ..... <code>ERES614400</code></li> <li>• DR1xxxA ..... <code>ERES819200</code></li> <li>• DR5xxxB ..... <code>ERES278528</code></li> <li>• DR5xxxA ..... <code>ERES425894</code></li> <li>• DM10xxB ..... <code>ERES655360</code></li> <li>• DM1xxxA ..... <code>ERES1024000</code></li> <li>• DM1004x ..... <code>ERES655360</code></li> </ul>
	2. Pulse width too narrow. (stepper axes)	2. Set pulse width to drive specifications using the <code>PULSE</code> command. See page 47 for Parker drive <code>PULSE</code> settings.
	3. Wrong scaling values.	3. Check the scaling parameters ( <code>SCALE1</code> , <code>SCLA</code> , <code>SCLD</code> , <code>SCLV</code> , <code>SCLMAS</code> ) – see also page 48.
Erratic operation.	1. Electrical Noise.	1. Reduce electrical noise or move product away from noise source.
	2. Improper shielding.	2. Refer to the instructions in your product's <i>Installation Guide</i> .
	3. Improper wiring.	3. Check wiring for opens, shorts, & mis-wired connections.
Feedback device (encoder or ANI) counts missing.	1. Improper wiring.	1. Check wiring.
	2. Feedback device slipping.	2. Check and tighten feedback device coupling.
	3. Encoder too hot.	3. Reduce encoder temperature with heatsink, thermal insulator, etc.
	4. Electrical noise.	4a. Shield wiring. 4b. Use encoder with differential outputs.
	5. Encoder frequency too high.	5. Peak encoder frequency must be below 12 MHz post-quadrature. Peak frequency must account for velocity ripple.

Problem	Cause	Solution
Following problems — see page 199.		
Joystick mode: Motor does not move.	1. Joystick Release input not grounded.	1.a. If an input is not assigned the “Joystick Release” input function, do so with the <code>INFNCi-M</code> command (see page 86). 1.b. Ground the Joystick Release input.
	2. Improper wiring.	2. Check wiring for opens, shorts, and mis-wired connections.
LEDs:	<i>All other LED states indicate hardware errors.</i>	
“POWER” LED is off.	1. No power.	1. Check 24VDC power connection and restore power.
“POWER” LED is red.	1. General fault. 2. ENABLE input not grounded.	1. Reset the controller by one of these methods: <ul style="list-style-type: none"> <li>• Cycle power</li> <li>• Issue the <code>RESET</code> command</li> </ul> 2. Ground the ENABLE input.
An “AXIS” LED is on (red).	1. Drive was commanded to shut down ( <code>DRIVE0</code> ). If Disable Drive on Kill mode is enabled ( <code>KDRIVE1</code> ), a kill command or kill input will also disabled the drive. 2. Servo Axes: Maximum position error ( <code>SMPER</code> value) exceeded. Could be caused by disconnected or mismatched feedback device.	1. Re-enable the drive by sending a <code>DRIVE1</code> command to the affected axis. 2. (verify position error by checking to see if <code>TAS/TASF</code> bit 23 is set) Check feedback device connection and mounting and re-enable drive by sending <code>DRIVE1</code> command to the affected axis.
Motion does not occur.	1. “AXIS” LED is red, or “POWER” LED is off or red. 2. End-of-travel limits are active. 3. Step pulse too narrow for drive to recognize (stepper axes only). 4. Drive fault level incorrect. 5. Improper wiring. 6. ENABLE input is not grounded. 7. Load is jammed. 8. No torque from motor. 9. Maximum position error ( <code>SMPER</code> value) exceeded. (servo axes only) 10. Drive has activated the drive fault input.	1. See LED troubleshooting as noted above. 2.a. Move load off of limits or disable limits by sending the <code>LH0</code> command to the affected axis. 2.b. Software limits: Set <code>LSPOS</code> to a value greater than <code>LSNEG</code> . 3. Set pulse width to drive specifications using the <code>PULSE</code> command (see page 47). 4. Set drive fault level using the <code>DRFLVL</code> command (see page 46). 5. Check drive fault & limit connections. Stepper Axes: check step and direction connections. Servo Axes: check command and shutdown connections. 6. Ground the ENABLE input connection. 7. Remove power and clear jam. 8. See problem: <i>Torque, loss of.</i> 9. Check to see if <code>TAS/TASF</code> bit 23 is set, and issue the <code>DRIVE1</code> command to the axis that exceeded the position error limit. 10. Check to see if <code>TAS/TASF</code> bit 14 is set, and check the drive fault level ( <code>DRFLVL</code> ) — see page 46 for appropriate <code>DRFLVL</code> settings.
Power-up Program does not execute.	1. ENABLE input is not grounded. 2. <code>STARTP</code> program is not defined.	1. Ground the ENABLE input to GND and reset the product. 2. Check the response to the <code>STARTP</code> command. If no program is reported, define the <code>STARTP</code> program and reset (see page 13, or see the <code>STARTP</code> command description).
Program access denied: receive the message *ACCESS DENIED when trying to use the <code>DEF</code> , <code>DEL</code> , <code>ERASE</code> , <code>LIMFNC</code> , <code>INFNC</code> , or <code>MEMORY</code> commands.	1. Program security function has been enabled ( <code>INFNCi-Q</code> or <code>LIMFNCi-Q</code> ) and the program access input has not been activated	1.a. Activate the assigned program access input, perform your programming changes, then deactivate the program access input. 1.b. Refer to the instructions on page 89, or to the <code>INFNC</code> or <code>LIMFNC</code> command descriptions.
Program execution: the first time a program is run, the move distances are incorrect. Upon downloading the program the second time, move distances are correct.	1. Scaling parameters were not issued when the program was downloaded; or scaling parameters have been changed since the program was defined.	1. Issue the scaling parameters ( <code>SCALE1</code> , <code>SCLA</code> , <code>SCLD</code> , <code>SCLV</code> , <code>PSCLA</code> , <code>PSCLD</code> , <code>PSCLV</code> , <code>SCLMAS</code> ) before saving any programs.

Problem	Cause	Solution
Program execution: stops at the DRFEN1 command	1. DRFEN1 enables drive fault monitoring, but the drive fault level (DRFLVL) command is set incorrectly for the drive being used.	1. Issue the correct DRFLVL command for your drive (see the DRFLVL command or to page 46).
Runaway (SERVOS ONLY)	1. Direction connections reversed. (if encoder counts positive when turned clockwise or extended).	1. Switch CMD- with the CMD+ connection to drive or valve. <b>NOTE:</b> The CMD+/- Connection is not differential. Do not connect CMD+ to ground on your drive or valve.
Torque, loss of.	1. Improper wiring.	1. Check wiring to the drive, as well as other system wiring.
	2. No power to drive .	2. Check power to drive.
	3. Drive failed.	3. Check drive status.
	4. Drive faulted.	4. Check drive status.
	5. Shutdown issued to drive.	5. Re-enable drive by sending the DRIVE1 command to the affected axis.
Velocity & acceleration is incorrect as programmed.	See <i>Distance</i> problem noted above.	

**Program Debug Tools** After creating your programs, you can need to debug the programs to ensure that they perform as expected. The 6K controller provides several debugging tools. Detailed descriptions are provided on the following pages.

- **Status Commands:** Use the “Transfer” commands (e.g., TAS, TSS, TIN) to display various controller status information. Multi-Tasking: System (TSS) and Error (TER) status information is task specific; to check the status for a specific task, you must prefix the status command with the task identifier (e.g., 2%TSS to check the system status for task 2).
- **Error Messages:** You can enable the 6K controller to display error messages when it detects certain programming errors as you enter them or as the program is run. When the controller detects an error with a command, you can issue the TCMDER command to find out which command caused the error.
- **Trace mode:** Trace a program as it is executing.
- **Single-Step mode:** Step through the program one command at a time.
- **Simulate Programmable I/O Activation:** You can set the desired state of the 6K controller’s inputs and outputs via software commands.
- **Simulate Analog Input Activation:** Without an actual voltage present, you can simulate a specific voltage on the 6K controller’s analog input channels using the ANIEN command.
- **Motion Planner’s Panel Gallery:** Motion Planner’s Panel Gallery provides an assortment of test panels you can use to verify various system I/O and operating parameters.

## Status Commands

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

### Checking Specific Setup Parameters

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

Refer to page 44 for a list of most setup parameters and their respective commands.

⇒ **HINT:** To send a status command to the 6K product during program execution, prefix the command with an “!” (e.g., !TASF).

Below is a list of the status commands that are commonly used for diagnostics. Additional status commands are available for checking other elements of your application (see *List of All Status Commands* below). For more information on each status command, see the respective command description in the *6K Series Command Reference*.

### SPECIAL NOTATIONS

- \* The command has a binary report version (just leave the “F” off when you type it in—e.g., TAS). This is used more by experienced 6K programmers. Using the binary report command, you can check the status of one particular bit (e.g., The 2TAS.1 command reports “1” if axis 2 is moving or “0” if it is not moving.). In the binary report the bits are numbered left to right, 1 through *n*. A “1” in the binary report correlates to a “YES” in the full text report, and a “0” correlates to a “NO” in the full text report.
- † The command has an assignment/comparison operator that uses the bit status for conditional expressions and variable assignments. For example, the WAIT (2AS.1=b0) command pauses program execution until axis 2’s status bit 1 (2AS.1) reports a binary zero value (indicates that the axis is not-moving). See page 7 and page 25 for more information on using assignment and comparison operators in conditional expressions and variable assignments.

TASF	Reports axis-specific conditions.	* (TAS)	† (AS)
1. Axis is in motion (commanded)	17. Positive-direction software limit (LSPOS) encountered		
2. Direction is negative	18. Negative-direction software limit (LSNEG) encountered		
3. Accelerating (n/a to deceleration)	19. RESERVED		
4. At velocity	20. RESERVED		
5. Home Successful (HOM)	21. RESERVED6. In absolute positioning mode (MA)		22.
RESERVED7. In continuous positioning mode (MC)	23. Position error limit is exceeded (SMPER) — servos		
8. In Jog Mode (JOG)	24. Load is within Target Zone (STRGTD & STRGTV)		
9. In Joystick Mode (JOY)	25. Target Zone timeout occurred (STRGTT) — servos		
10. RESERVED	26. Motion suspended, pending GOWHEN		
11. RESERVED	27. RESERVED		
12. Stall detected (ESTALL) — steppers	28. Registration move occurred since last GO		
13. Drive shutdown occurred	29. GOWHEN error		
14. Drive fault occurred (enable DRFEN1 first)	30. Pre-emptive (OTF) GO or Registration profile not possible		
15. Positive-direction hardware limit hit	31. Executing profile		
16. Negative-direction hardware limit hit	32. RESERVED		

**TASXF**

Reports extended axis-specific conditions.

\* (TASK)

† (ASX)

1. RESERVED
2. RESERVED
3. RESERVED
4. Drive Fault input is active (hardware state is always recognized, regardless of DRIVE and DRFEN)
5. Encoder failure detected (encoder failure detection must first be enabled with the EFAIL1 command)
6. Z-Channel input (on the encoder connector) is active.
7. Drive Stall Active

**TSTAT**

Reports general system setup and current conditions.

Sample response for the 6K8:

```

*6K8 (8-axis controller)
*6K revision: 92-XXXXXX-01-5.0 6K 92-XXXXXX-XX-NOP2.5 DSP
*Ethernet address: xxxxxxxxxx; 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): 0000_0000_00
*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

```

<b>TSSF</b>	Reports current system conditions. * (TSS) † (SS) <u>Multi-tasking</u> : Each task has its own system status; therefore, to check the system status for a specific task, prefix the TSSF command (e.g., 2%TSSF).
1. System is ready 2. RESERVED 3. Executing a Program 4. Last command was immediate 5. In ASCII Mode 6. In Echo Mode (ECHO) 7. Defining a Program (DEF) 8. In Trace Mode (TRACE, TRACEP) 9. In Step Mode (STEP) 10. RESERVED 11. Command error (check with TCMER) 12. Break Point Active (BP) 13. Pause Active (PS or pause input) 14. Wait Active (WAIT) 15. Monitoring On Conditions (ONCOND) 16. Waiting for Data (READ)	17. Loading Thumbwheel Data (TW operator) 18. In External Program Select Mode (INSELP) 19. Dwell in Progress (T command) 20. Waiting for RP240 Data (DREAD or DREADF) 21. RP240 Connected 22. Non-volatile Memory Error 23. Gathering servo data 24. RESERVED 25. RESERVED 26. RESERVED 27. RESERVED 28. RESERVED 29. Compiled memory partition is 75% full 30. Compiled memory partition is 100% full 31. Compile operation (PCOMP) failed 32. RESERVED

<b>TINOF</b>	Reports the status of the ENABLE input. * (TINO) † (INO)
1-5. RESERVED 6. ENABLE input OK (motion not inhibited) 7-8. RESERVED	

<b>TFSF</b>	Reports Following Mode conditions (details on page 167). * (TFS) † (FS)
1. Slave is in a Following ratio move 2. Current ratio is negative 3. Slave is changing ratio 4. Slave at ratio (constant non-zero ratio) 5. FOLMAS Active 6. Following Mode enabled (FOLEN) 7. Master is moving 8. Master direction is negative 9. OK to Shift 10. Shifting now 11. FSHFC-based shift move is in progress 12. Shift direction is negative 13. Master cycle trigger input is pending 14. Mas cycle length (FMCLN) given 15. Master cycle position is negative 16. Master cycle number is > 0	17. Master Position Prediction Mode enabled (FPPEN) 18. Master Position Filtering Mode enabled (FFILT) 19. RESERVED 20. RESERVED 21. RESERVED 22. RESERVED 23. OK to do a geared advance (FGADV) move 24. Geared advance (FGADV) move is underway 25. RESERVED 26. Following profile is limited (FMAXA/FMAXV) 27. RESERVED 28. RESERVED 29. RESERVED 30. RESERVED 31. RESERVED 32. RESERVED

## TERF

Reports error conditions. \*\*

\* (TER) † (ER)

Multi-tasking: Each task has its own error status; therefore, to check the error status for a specific task, prefix the TERF command (e.g., 2%TERF).

1. Stall detected. 1st: Enable Stall Detection (ESTALL).
2. Hardware end-of-travel limit encountered. 1st: Enable hard limits (LH).
3. Software end-of-travel limit encountered. 1st: Enable hard limits (LH).
4. Drive Fault input is active. 1<sup>st</sup>: enable (DRFEN) & set fault level (DRFLVL).
5. Stop or Kill Issued
6. A programmable input, defined as a “kill” input, is active.
7. A programmable input, defined as a “user fault” input, is active.
8. A programmable input, defined as a “stop” input, is active.
9. ENABLE input not grounded.
10. Pre-emptive (OTF) GO or Registration profile not possible.
11. Target Zone settling timeout period (STRGTT) is exceeded. — servo only
12. Maximum position error (SMPER value) is exceeded. — servo only
13. RESERVED
14. GOWHEN condition already true.
15. RESERVED
16. Bad command detected (use TCMDEP to identify the bad command).
17. Encoder failure detected (1<sup>st</sup>: enable failure detection with EFAIL1).
18. Expansion I/O brick is disconnected or has lost power.
19. Option Card Fault
20. RESERVED
21. RESERVED
22. Ethernet Failure
23. Client Connect Error
24. Client Polling Error
- 25–32. RESERVED

\*\* The error condition will not be reported until you enable the respective error-checking bit with the ERROR command (for details, see page 30 or the ERROR command description).

**NOTE:** When the error-checking bit is enabled and the error occurs, the controller will branch to the “error” program that you assigned with the ERRORP command.

## Other status commands commonly used for diagnostics:

TDIR.....	Identifies the name and number of all programs residing in the 6K product's memory. Also reports percent of available memory for programs and compiled path segments.
TCMDER....	Identifies the bad command that caused the error prompt (?). (see page 235 for details)
TEX.....	Execution status (and line of code) of the current program in progress. Task specific.
TIN.....	Binary report of all programmable and trigger inputs ("1" = active, "Ø" = inactive). <i>INFNC</i> also reports the state and programmed function of each input. (see page 76 for bit assignments)
TOUT.....	Binary report of all programmable and auxiliary outputs ("1" = active, "Ø" = inactive). <i>OUTFNC</i> also reports the state and programmed function of each output. (see page 76 for bit assignments)
TLIM.....	Binary report of all limit inputs ("1" = active, "Ø" = inactive). <i>LIMFNC</i> also reports the state and programmed function of each limit input. (see page 76 for bit assignments)
TIO.....	Reports current contents on all expansion I/O bricks connected to the 6K controller. Includes current state and function of the digital inputs and outputs, as well as voltage of analog inputs.
TPER.....	(servo axes) Reports the difference between the commanded position and the actual position as measure by the feedback device.
TPC.....	Current commanded position.
TPE.....	Current position of the encoder.
TFB.....	(servo axes) Current position of the feedback device selected with the last <i>SFB</i> command.
TPMAS.....	Current position of the Following master axis.
TPSLV.....	Current position of the Following slave axis.
TNMCY.....	Current master cycle number.
TSCF.....	Full-text report of the current "Controller Status" register.
TNT.....	Report the current Ethernet status.

### List of All Status Commands

#### SPECIAL NOTATIONS

- \* The command responds with a binary report. This is used more by experienced 6K programmers. Using the bit select operator (.), you can check the status of one particular bit (e.g., The *2TAS.1* command reports "1" if axis 2 is moving or "Ø" if it is not moving.). In the binary report, the bits are numbered left to right, 1 through *n*. A "1" in the binary report correlates to a "YES" in the full text report, and a "Ø" correlates to a "NO" in the full text report.
- Δ The command has a full-text report version (just add an "F" when you type it in—e.g., *TASF*). This makes it easier to check status information without having to look up the purpose of each status bit. (see full-text descriptions on pages 226-230)
- † The command has an assignment/comparison operator that uses the bit status for conditional expressions and variable assignments. For example, the *WAIT (2AS.1=bØ)* pauses progress execution until axis 2's status bit 1 (*2AS.1*) reports a binary zero value (indicates that the axis is not-moving). See page 7 and page 25 for more information on using assignment and comparison operators in conditional expressions and variable assignments.

## COMMAND STATUS SUBJECT

---

TANI	.....	Voltage of ANI inputs ( <i>servo products with ANI option</i> ) †
TANO	.....	report the value of an analog output channel †
TAS	.....	Binary Report of Axis Status * Δ †
TASX	.....	Binary Report of Axis Status – extended * Δ †
TCMDER	.....	Command Error (view command that caused the error prompt)
TDAC	.....	Digital-to-Analog (DAC) Voltage (Servos) †
TDIR	.....	Program Directory and Available Memory
TDPTR	.....	Data Pointer Status †
TER	.....	Error Status * Δ †
TEX	.....	Program Execution Status †
TFB	.....	Position of Selected Feedback Devices †
TFS	.....	Binary Report of Following Status * Δ †
TGAIN	.....	Current Value of Active Servo Gains
TIN	.....	Binary Report of Status of Programmable Inputs * †
TINO	.....	Status of the ENABLE input (bit 6) * Δ †
TIO	.....	Report of all I/O on expansion I/O bricks
TLABEL	.....	Defined Labels (names of)
TLIM	.....	Binary Report of Hardware Status of All Limit Inputs †
TMEM	.....	Memory Usage (partition and available memory)
TNMCY	.....	Master Cycle Number †
TNT	.....	Reports the current Ethernet conditions
TNTMAC	.....	Ethernet Address
TOUT	.....	Binary Report of Status of Programmable Outputs * †
TPANI	.....	Position of ANI Inputs †
TPC	.....	Commanded Position †
TPCC	.....	Captured Commanded Position †
TPCE	.....	Captured Encoder Position †
TPCME	.....	Captured Master Encoder Position †
TPCMS	.....	Captured Master Cycle Position †
TPE	.....	Position of Encoder †
TPER	.....	Position Error †
TPMAS	.....	Position of Master Axis †
TPME	.....	Position of Master Encoder †
TPROG	.....	Contents of a Program
TPSHF	.....	Net Position Shift †
TPSLV	.....	Current Commanded Position of the Slave Axis †
TREV	.....	Firmware Revision Level
TSC	.....	Current "Controller Status" register.
TSCAN	.....	Scan Time of Last PLC Program * Δ †
TSGSET	.....	Servo Gain Sets
TSEG	.....	Number of Free Segment Buffers †
TSS	.....	Binary Report of System Status * Δ †
TSTAT	.....	Statistics
TSTLT	.....	Settling Time
TSWAP	.....	Identify Currently Active Tasks (in multi-tasking) * †
TTASK	.....	Task Number of the program that executes this command †
TTIM	.....	Time Value †
TTRIG	.....	Status of "Trigger Interrupt" Activation * †
TUS	.....	User Status * †
TVEL	.....	Current Commanded Velocity †
TVELA	.....	Current Actual Velocity †
TVMAS	.....	Current Velocity of the Master Axis †

## Error Messages

Depending on the error level setting (set with the `ERRLVL` command), when a programming error is created, the 6K controller will respond with an error message and/or an error prompt. A list of all possible error messages is provided in a table below. The default error prompt is a question mark (?), but you can change it with the `ERRBAD` command if you wish.

At error level 4 (`ERRLVL4`—the factory default setting) the 6K controller responds with both the error message and the error prompt. At error level 3 (`ERRLVL3`), the 6K controller responds with only the error prompt.

Error Response	Possible Cause
ACCESS DENIED	Program security feature enabled, but program access input ( <code>INFNCi-Q</code> ) not activated.
ALREADY DEFINED FOR THUMBWHEELS	Attempting to assign an I/O function to an I/O that is already defined as a thumbwheel I/O.
alternative task not allowed	Attempting to execute a <code>LOCK</code> command directed to another task.
AXES NOT READY	Compiled Profile path compilation error.
COMMAND NOT IMPLEMENTED	Command is not applicable to the 6K Series product.
COMMAND NOT ALLOWED IN PROGRAM	Command is not allowed inside a program definition (between <code>DEF</code> and <code>END</code> ).
COMMAND/drive mismatch	The command (or $\geq$ one field in the command) is not appropriate to the <code>AXSDEF</code> configuration (e.g., attempting to execute a servo tuning command on a stepper axis)
ERROR: MOTION ENDS IN NON-ZERO VELOCITY - AXIS N	Compiled Motion: The last <code>GOBUF</code> segment within a <code>PLOOP/PLM</code> loop does not end at zero velocity, or there is no final <code>GOBUF</code> segment placed outside the loop.
EXCESSIVE PATH RADIUS DIFFERENCE	Contouring path compilation error.
FOLMAS NOT SPECIFIED	No <code>FOLMAS</code> for the axis is currently specified. It will occur if <code>FMCNEW</code> , <code>FSHFC</code> , or <code>FSHFD</code> commands are executed and no <code>FOLMASØ</code> command was executed, or <code>FOLMAS0</code> was executed.
INCORRECT AXIS	Axis specified is incorrect.
INCORRECT BRICK NUMBER	Attempted to execute a command that addresses an I/O brick that is not connected to your 6K controller.
INCORRECT DATA	Incorrect command syntax. Following: Velocity ( <code>v</code> ), acceleration ( <code>A</code> ) or deceleration ( <code>AD</code> ) command is zero (used by <code>FSHFC</code> & <code>FSHFD</code> ).
INPUT(S) NOT DEFINED AS JOYSTICK INPUT	Attempted to execute <code>JOYCDB</code> , <code>JOYCTR</code> , <code>JOYEDB</code> , or <code>JOYZ</code> before executing <code>JOYAXH</code> or <code>JOYAXL</code> to assign the analog input to an axis.
INSUFFICIENT MEMORY	Not enough memory for the user program or compiled profile segments. This can be remedied by reallocating memory (see <code>MEMORY</code> command description).
INVALID COMMAND	Command is invalid because of existing conditions

<b>Error Response</b>	<b>Possible Cause</b>
INVALID CONDITIONS FOR COMMAND	<p>System not ready for command (e.g., LN command issued before the L command).</p> <p>Following (these conditions can cause an error during Following):</p> <ul style="list-style-type: none"> <li>• The FOLMD value is too small to achieve the preset distance and still remain within the FOLRN/FOLRD ratio.</li> <li>• A phase shift cannot be performed: <ul style="list-style-type: none"> <li>FSHFD .... Error if already shifting or performing other time based move.</li> <li>FSHFC .... Error if currently executing a FSHFD move, or if currently executing another FSHFC move in the opposite direction.</li> </ul> </li> <li>• The FOLEN1 command was given while a profile was suspended by a GOWHEN.</li> </ul>
INVALID CONDITIONS FOR S_CURVE ACCELERATION—FIELD n	Average (AA) acceleration or deceleration command (e.g., AA, ADA, HOMAA, HOMADA, etc.) with a range that violates the equation $\frac{1}{2}A \leq AA \leq A$ (A is the maximum accel or decel command—e.g., A, AD, HOMA, HOMAD, etc.)
INVALID DATA	<p>Data for a command is out of range.</p> <p>Following (these conditions can cause an error during Following):</p> <ul style="list-style-type: none"> <li>• The parameter supplied with the command is valid. <ul style="list-style-type: none"> <li>FFILT .... Error if: smooth number is not 0-4</li> <li>FMCLEN .. Error if: master steps &gt; 999999999 or negative</li> <li>FMCP..... Error if: master steps &gt; 999999999 or &lt; -999999999</li> <li>FOLMD .... Error if: master steps &gt; 999999999 or negative</li> <li>FOLRD .... Error if: master steps &gt; 999999999 or negative</li> <li>FOLRN .... Error if: follower steps &gt; 999999999 or negative</li> <li>FSHFC .... Error if: number is not 0-3</li> <li>FSHFD .... Error if: follower steps &gt; 999999999 or &lt; -999999999</li> <li>GOWHEN .. Error if: position &gt; 999999999 or &lt; -999999999</li> <li>WAIT..... Error if: position &gt; 999999999 or &lt; -999999999</li> </ul> </li> <li>• Error if a GO command is given in the preset positioning mode (MCØ) and: <ul style="list-style-type: none"> <li>FOLRN = zero</li> <li>FOLMD = zero, or too small</li> <li>(see Following chapter on page 166)</li> </ul> </li> </ul>
INVALID FOLMAS SPECIFIED	Following: An illegal master was specified in FOLMAS. A follower can never use its own commanded position or feedback source as its master.
INVALID RATIO	Following: Error if the FOLRN:FOLRD ratio after scaling is > 127 when a GO is executed
INVALID TASK IDENTIFIER	Attempting to launch a PEXE or EXE command into the supervisor task (task 0).
MASTER, SLAVE DISTANCE MISMATCH	Attempting a preset Following move with a FOLMD value that is too small.
LABEL ALREADY DEFINED	Defining a program or label with an existing program name or label name
MAXIMUM COMMAND LENGTH EXCEEDED	Command exceeds the maximum number of characters
MAXIMUM COUNTS PER SECOND EXCEEDED	Velocity value is greater than 1,600,000 counts/sec

<b>Error Response</b>	<b>Possible Cause</b>
MOTION IN PROGRESS	Attempting to execute a command not allowed during motion (see <i>Restricted Commands During Motion</i> on page 17).  Following: The FOLEN1 command was given while that follower was moving in a non-Following mode.
NEST LEVEL TOO DEEP	IFs, REPEATs, WHILEs, or GOSUBs nested greater than 16 levels (for each type)
NO MOTION IN PROGRESS	Attempting to execute a command that requires motion, but motion is not in progress
NO PATH SEGMENTS DEFINED	Compiled Profile compilation error
NO PROGRAM BEING DEFINED	END command issued before a DEF command
NOT ALLOWED IF SFBØ	Changes to tuning commands (SGILIM, SGAF, SGI, SGP, SGV, and SGVF) and SMPER are not allowed if SFBØ is selected
NOT ALLOWED IN PATH	Compiled Profile path compilation error
NOT DEFINING A PATH	Executing a compiled profile or contouring path command while not in a path
NOT VALID DURING FOLLOWING MOTION	A GO command was given while moving in the Following mode (FOLEN1) and while in the preset positioning mode (MCØ).
NOT VALID DURING RAMP	A GO command was given while moving in a Following ramp and while in the continuous positioning mode (MC1). Following status (FS) bit 3 will be set to 1. A FOLEN command was given during one of these conditions: <ul style="list-style-type: none"> <li>• During a shift (FSHFC or FSHFD)</li> <li>• During a change in ratio (FOLRN/FOLRD)</li> <li>• During deceleration to a stop</li> </ul>
OUTPUT BIT USED AS OUTFNC	Attempted to change an output that is not an OUTFNCi-A output.
PATH ALREADY MOVING	Compiled Profile path compilation error
PATH NOT COMPILED	Attempting to execute a individual axis profile or a multiple axis contouring path that has not been compiled
PATH RADIUS TOO SMALL	Contouring path compilation error
PATH RADIUS ZERO	Contouring path compilation error
PATH VELOCITY ZERO	Contouring path compilation error
STRING ALREADY DEFINED	A string (program name or label) with the specified name already exists
STRING IS A COMMAND	Defining a program or label that is a command or a variant of a command
SYSTEM UPDATE OVERRUN, USE SYSPER4	The system update service has overrun the time allotted with the default 2-millisecond period. Use the SYSPER4 command to increase the system update period to 4 milliseconds.

Error Response	Possible Cause
UNDEFINED LABEL	Command issued to product is not a command or program name
WARNING: POINTER HAS WRAPPED AROUND TO DATA POINT 1	During the process of writing data (DAT TCH) or recalling data (DAT), the pointer reached the last data element in the program and automatically wrapped around to the first datum in the program
WARNING: ENABLE INPUT INACTIVE	ENABLE input is no longer connected to ground (GND)
WARNING: DEFINED WITH ANOTHER TW/PLC	Duplicate I/O in multiple thumbwheel definitions

## Identifying Bad Commands

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

### Using Motion Planner:

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

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

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

### Example Error Scenario

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

```

DEL badprg      ; Delete a program before defining and downloading
DEF badprg      ; Begin definition of program called badprg
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 < 16, increment the counter by 1
NIF             ; End IF statement
END             ; End programming of program called badprg

```

2. Using Motion Planner's terminal emulator, download the program to the 6K Series product. Notice that an error response identifies the bad command as an "INCORRECT DATA" item and displays it:

```

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

```

## Trace Mode

You can use the Trace mode to debug a program. The Trace mode allows you to track, command-by-command, the entire program as it runs. The 6K controller will display all of the commands as they are executed. **NOTE:** Program tracing is also available on the RP240 display (see page 111).

The example below demonstrates the Trace mode.

**Step 1** Create a program called prog1:

```
DEF prog1          ; Begin definition of program prog1
A10                ; Acceleration is 10
AD10               ; Deceleration is 10
V5                 ; Velocity is 5
L3                 ; Loop 3 times
GOSUB prog3        ; Gosub to program 3 (prog3)
LN                 ; End the loop
END                ; End definition of program prog1
```

**Step 2** Create a program prog3:

```
DEF prog3          ; Begin definition of program prog3
D50000             ; Sets the distance to 50,000
GO1                ; Initiates motion
END                ; End definition of program prog3
```

**Step 3** Enable the Trace Mode:

```
TRACE1            ; Enables the Trace mode
```

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

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

The response will be:

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

The format for the Trace mode display is:

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

**Step 5** Exit the Trace Mode.

```
TRACE0            ; Disables the Trace mode
```

## Tracing Program Flow

Using the TRACEP command, you can monitor the entry and exit of programs and their associated nest levels.

For example, let's assume these four programs are defined:

```
DEF PICK1
GOSUB PICK2
GOTO PICK3
END
```

```
DEF PICK2
GOSUB PICK4
END
```

```
DEF PICK3
END
```

```
DEF PICK 4
END
```

Now we'll enable the TRACEP mode and launch the calling program (PICK1) to start tracing the program flow:

```
>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
>TRACEP0
```

## Single-Step Mode

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

**Step 1** Enable the Single-Step Mode:

```
STEP1          ; Enables Single Step Mode
```

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

```
TRACE1        ; Enables the Trace mode
RUN prog1     ; Run program called prog1
```

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

```
!#           ; Executes one command
```

The response will be:

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

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

```
!#3          ; Executes three commands
```

The response will be:

```
*PROGRAM=PROG1          COMMAND=AD10.0000
*PROGRAM=PROG1          COMMAND=V5.0000
*PROGRAM=PROG1          COMMAND=L3
```

To complete the sequence, use the # sign until all the commands are completed (!#16 would complete the example).

**Step 5** To exit Single-Step mode, type:

```
STEP0        ; Disables Single Step Mode
```

## Breakpoints

The Break Point (BP) command allows the programmer to set a place in the program where command processing will halt and a message will be transmitted to the PC. There are 32 break points available, BP1 to BP32, all transmitting the message \*BREAKPOINT NUMBER x<cr> where x is the break point number.

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

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

### Example

```
DEF prog1                ; Begin definition of program named prog1
D50000,1000              ; Set distance to 50000 units on axis 1, and 1000 units
                        ; on axis 2
MA1100                  ; Absolute mode for axes 1 and 2
GO1100                  ; Initiate motion on axes 1 and 2
IF(1PC>40000)           ; Compare axis 1 commanded position to 40000
BP1                      ; If the motor position is > 40000 units, set break
                        ; point 1
NIF                      ; End IF statement
D80000,2000             ; Set distance to 80000 units on axis 1, and 2000 units
                        ; on axis 2
GO1100                  ; Initiate motion on axes 1 and 2

BP2                      ; Set break point 2
END                      ; End program definition
RUN prog1               ; Execute program prog1
```

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

## Simulating I/O Activation

If your application has inputs and outputs that integrate the 6K controller with other components in your system, you can simulate the activation of these inputs and outputs so that you can run your programs without activating the rest of your system. Thus, you can debug your program independent of the rest of your system.

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

### NOTE

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

The OUTEN command has no effect on the onboard outputs when they are configured as *output-on-position* outputs with the OUTFNCi-H command.

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

## Outputs

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

### Step 1

Display the state of the outputs with the TOUT command:

```
TOUT          ; Displays the state of the outputs
```

The response will be:

```
*TOUT0000_0000_0000_0000_0000_0000_00
```

Display the function of the outputs with the OUTFNC command:

```
OUTFNC        ; Displays the state of the outputs
```

The response will be:

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

### Step 2

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

```
OUTEN1111     ; Disable outputs 1-4, leave them in ON state
OUTFNC        ; Displays the state of the outputs
```

The response will be:

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

### Step 3

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

```
OUT1010       ; Activates outputs 1 and 3, deactivates outputs 2 and 4
```

Display the state of the outputs with the OUTFNC command.

```
OUTFNC        ; Displays the state of the outputs
```

The response will be:

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

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

### Step 4

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

```
OUTENEEEE     ; Re-enables outputs 1-4
```

## Inputs

The steps below describe the use and function of the `INEN` command. You can use it to cause an input state to occur. The inputs will not actually be in this state but the 6K controller treats them as if they are in the given state and will use this state to execute its program.

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

```
INFNC1-A      ; Onboard input 1 is has no function
INFNC2-A      ; Onboard input 2 is has no function
INLVL00       ; Set input 1 and 2 active level to low
DEF prog8     ; Begin definition of program prog8
A100          ; Acceleration is set to 100
AD100         ; Deceleration is 100
V5            ; Velocity is 5
D25000        ; Distance is 25,000
WAIT(IN=b11)  ; Waits for the input state to be 11
GO1           ; Initiate motion
END           ; End definition of program prog8
```

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

```
TRACE1        ; Enables the trace mode
```

**Step 3** Execute the program:

```
RUN prog8     ; Runs program prog8
```

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

```
!INEN11      ; Disables inputs 1 and 2, leaving them in the ON state
```

The motor will now move for 25000 steps.

**Step 5** Deactivate the input simulation:

```
INENEE       ; Re-enables inputs 1 and 2
```

## Simulating Analog Input Channel Voltages

Without actually applying any voltage, you can test any command or function that references the voltage on an analog input (located on an expansion I/O brick). For example, `2ANIEN.1=1.2,1.6,1.8` overrides I/O brick 2's hardware analog input 1 through 3 as follows: 1.2V on input 1, 1.6V on input 2, and 1.8V on input 3.

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

## Motion Planner's Panel Gallery

Motion Planner's Panel Gallery provides diagnostic panels for testing your programs and monitor the following:

- I/O (programmable I/O, analog I/O, limits)
- Motion (motor and feedback device position, velocity)
- Status (axis, system, interrupt, user-defined, Following)
- Terminal (direct communication with the product, to check for error messages, etc.)

# Technical Support

---

For solutions to your questions about implementing 6K product software features, first look in this manual. Other aspects of the product (command descriptions, hardware specs, I/O connections, graphical user interfaces, etc.) are discussed in the respective manuals listed in *Reference Documentation* on page ii.

If you cannot find the answer in this documentation, contact your local Automation Technology Center (ATC) or distributor for assistance.

If you need to talk to our in-house application engineers, please contact us at the numbers listed on the inside cover of this manual. (The phone numbers are also provided when you issue the `HELP` command to the 6K controller.) **NOTE**

## Operating System Upgrades

---

You can obtain an upgraded 6K operating system from our web site at [www.compumotor.com](http://www.compumotor.com). Instructions for downloading and upgrading the operating system are provided on the website.

## Product Return Procedure

---

If you must return your 6K Series product to affect repairs or upgrades, use this procedure:

- Step 1* Get the serial number and the model number of the defective unit, and a purchase order number to cover repair costs in the event the unit is determined by the manufacturers to be out of warranty.
- Step 2* Before you return the unit, have someone from your organization with a technical understanding of the 6K Series product and its application include answers to the following questions:
- What is the extent of the failure/reason for return?
  - How long did it operate?
  - Did any other items fail at the same time?
  - What was happening when the unit failed (e.g., installing the unit, cycling power, starting other equipment, etc.)?
  - How was the product configured (in detail)?
  - What, if any, cables were modified and how?
  - With what equipment is the unit interfaced?
  - What was the application?
  - What was the system environment (temperature, enclosure, spacing, unit orientation, contaminants, etc.)?
  - What upgrades, if any, are required (hardware, software, user guide)?
- Step 3* Call for a return authorization. Refer to the Technical Support phone numbers provided on the inside cover of this manual. The support personnel will also provide shipping guidelines.