

CHAPTER FOUR

Product Control Options

IN THIS CHAPTER

This chapter explains various options for controlling your 6K product:

- Safety features 100
- Overview of product control options 101
- Programmable I/O (*switches, thumbwheels, PLCs, PLC Scan Mode, etc.*) 102
- RP240 remote operator panel 107
- Joystick and analog input interface (*requires ANI SIM on expansion I/O brick*) 114
- Host computer interface 118

Safety Features



WARNING



The 6K Product is used to control your system's electrical and mechanical components. Therefore, you should test your system for safety under all potential conditions. Failure to do so can result in damage to equipment and/or serious injury to personnel.

To help ensure a safe operating environment, you should take advantage of the safety features listed below. These features must not be construed as the only methods of ensuring safety. *See Also* refers you to where you can find more in-depth information about the feature (system connections and/or programming instructions).

Feature	Description	See Also
Enable Input	The ENABLE input, found on the 6K product's screw-terminal connector, is provided as an emergency stop input to the controller. If the ENABLE input is opened (disconnected from GND), output to the drives is inhibited. For stepper axes, step pulses to the drive are immediately cut off. For servo axes, the analog output voltage between CMD+ and CMD- is clamped to almost zero, and the shutdown outputs are activated on all axes. (The clamping circuit is also connected to the watchdog timer; if the controller's microprocessor fails, the analog output voltage will be clamped.)	6K Product's <i>Installation Guide</i>
Shutdown Outputs	The controller uses the shutdown outputs to disable the attached drive if it detects a problem. Two types of relay outputs are found on the DRIVE connectors— SHTNC for drives that require a closed contact to disable the drive, and SHTNO for drives that require an open contact to disable the drive. The shutdown relay outputs are essential for smooth power-up and power-down of the system. The shutdown relay is active (disabling the drive) when no power is applied to the controller. When the controller is powered up, the shutdown relay remains active until you issue the DRIVE1111 command. CAUTION: When shut down, the drive cuts all control to the motor and allows the load to freewheel to a stop.	6K Product's <i>Installation Guide</i>
Drive Fault Inputs	The drive fault (DFT) inputs, found on the product's DRIVE connectors, allows the drives to tell the controller if they encounter a fault condition. When a drive fault occurs, the controller stops motion (at the rate set with the LHAD and LHADA commands) and terminates program execution. No drive shutdown will result unless it is initiated with an ERRORP error program. NOTE: The drive fault input state will not be checked unless you enable the drive fault input (DRFEN1) and enable the drive (DRIVE1).	6K Product's <i>Installation Guide</i>
End-of-travel Limit Inputs	End-of-travel limits prevent the load from crashing through mechanical stops, an incident that can damage equipment and injure personnel. Use hardware or software limits, as your application requires. The default hardware limits are found on the LIMITS connector. Software limits are set with the LSPOS and LSNEG commands.	<i>End-of-Travel Limits</i> (page 57)
User Fault Input	Using the INFNCi-F command or the LIMFNCi-F command, you can assign any of the programmable inputs the <i>user fault</i> function. You can then wire the input to activate when an external event, considered a <i>fault</i> by the user, occurs.	<i>Input Functions</i> (page 83)
Maximum Allowable Position Error (servos only)	A <i>position error</i> (TPER) is defined as the difference between the commanded position (TPC) and the actual position as measured by the feedback device (TFB). The maximum allowable position error is set with the SMPER command. When the maximum allowable position error is exceeded (usually due to instability or loss of position feedback), the controller shuts down the drive and sets error status bit 12 (reported by the TER command). If SMPER is set to zero (SMPER0), position error will not be monitored.	<i>Servo Setup</i> (page 67)

When any of the safety features listed above are exercised (e.g., **DFT** input is activated, etc.), the controller considers it an error condition. With the exception of the shutdown output activation, you can enable the **ERROR** command to check for the error condition, and when it occurs to branch to an assigned **ERRORP** program. Refer to *Error Handling* (page 30) for further information.

Options Overview

Stand-Alone Interface Options

After defining and storing controller programs, the controller can operate in a stand-alone fashion. A program stored in the controller can interactively prompt the user for input as part of the program (input via I/O switches, thumbwheels, RP240, joystick). A joystick can be used for situations requiring manual manipulation of the load.

Option	Application Example
RP240 (see page 107)	Grinding: Program the RP240 function keys to select certain part types, and program one function key as a GO button. Select the part you want to grind, then put the part in the grinding machine and press the GO function key. The controller will then move the machine according to the predefined program assigned to the function key selected.
Joystick (see page 114)	X-Y scanning/calibration: Enter the joystick mode and use the 2-axis joystick to position an X-Y table under a microscope to arbitrarily scan different parts of the work piece (e.g., semi-conductor wafer). You can record certain locations to be used later in a motion process (e.g., for drilling, cutting, or photographing the work piece). The <i>Variable Arrays</i> section on page 94 provides an example using the joystick to teach positions.
ANI Analog Inputs (see page 117)	Injection Molding: Use for feedback from a pressure sensor to maintain constant, programmable force. (NOTE: ANI control requires that you install an analog input SIM on an expansion I/O brick.)
Programmable inputs scanned in PLC Scan Mode (see page 104)	The PLC Scan feature allows you to create a pre-compiled program that mimics PLC functionality by scanning through the I/O faster than in a normal program run. Because the functions of PLC programs are pre-compiled, delays associated with command processing are eliminated during profile execution, allowing more rapid sequencing of actions than would be possible with programs that are not compiled. Command processing is then free to monitor other activities.

Programmable Logic Controller

The controller's programmable I/O can be connected to most PLCs. After defining and storing controller programs, the PLC typically executes programs, loads data, and manipulates inputs to the controller. The PLC instructs the controller to perform the motion segment of a total machine process.

EXAMPLE (X-Y point-to-point): A PLC controls several tools to stack and bore several steel plates at once. The controller is programmed to move an X-Y table in a pre-programmed sequence. The controller moves the load when the inputs are properly configured, signals the PLC when the load is in position, and waits for the signal to continue to the next position.

Host Computer Interface

A computer can be used to control a motion or machine process. A PC can monitor processes and orchestrate motion by sending motion commands to the controller or by executing motion programs already stored in the controller. This control might come from a BASIC or C program. A BASIC program example is provided on page 101.

Programmable I/O Devices

Programmable I/O Functions

Programmable inputs and outputs are provided to allow the controller to detect and respond to the state of switches, thumbwheels, electronic sensors, and outputs of other equipment such as drives and PLCs. Listed below are the programmable functions that can be assigned to the programmable I/O.

Programmable I/O offering differs by product. The total number of onboard inputs and outputs (trigger inputs, limit inputs, digital outputs) depends on the product. The total number of expansion inputs and outputs (analog inputs, digital inputs and digital outputs) depends on your configuration of expansion I/O bricks. To determine I/O bit pattern for your product, see page 76.

NOTE

Refer to page 75 for instructions on establishing programmable input and output functions. Instructions for connecting to I/O devices are provided in your product's *Installation Guide*.

Input Functions

Input functions can be assigned to two basic groups of programmable inputs. LIMFNC assigns functions to the limit inputs found on the "LIMITS/HOME" connector. INFNC assigns functions to the trigger inputs (on the "TRIGGERS/OUTPUTS" connector) and to the digital inputs installed on expansion I/O bricks. The syntax, LIMFNC_i-c or INFNC_i-c, requires a letter designator ("c") that corresponds to an input function; options for the input functions are listed in the table below.

Virtual Inputs can be established to provide programmable input functionality for data or external events that are not ordinarily represented by inputs. See page 79 for details.

Letter Designator	Function
A.....	General-purpose input (default function for triggers & inputs on I/O bricks)
B.....	BCD program select
C.....	Kill
<a>D.....	Stop (axis designator "a" is optional)
E.....	Pause/Continue
F.....	User fault
G.....	<RESERVED>
H.....	Trigger Interrupt for position capture or registration (trigger inputs only). Special trigger functions can be assigned with the TRGFN command (see page 162).
I.....	Cause alarm event (requires Ethernet interface in Communications Server)
aJ.....	Jog in the positive-counting direction (axis designator is required)
aK.....	Jog in the negative-counting direction (axis designator is required)
aL.....	Jog velocity select (axis designator is required)
M.....	Joystick release
N.....	Joystick axis select
O.....	Joystick velocity select
P.....	One-to-one program select
Q.....	Program security
aR.....	End-of-travel limit for positive-counting direction (axis designator is required) *
aS.....	End-of-travel limit for negative-counting direction (axis designator is required) *
aT.....	Home limit (axis designator is required) *

* The limit inputs are factory-set to their respective end-of-travel or home limit function (see page 76).

Output Functions

Command	Function
OUTFNC _i -A.....	General-purpose output (default function)
OUTFNC _i -<a>B.....	Moving/not moving (axis designator optional)
OUTFNC _i -C.....	Program in progress
OUTFNC _i -<a>D.....	Hardware or software end-of-travel limit encountered (axis designator optional)
OUTFNC _i -<a>E.....	Stall indicator (axis designator optional) — stepper axes only
OUTFNC _i -F.....	Fault indicator (indicates drive fault input or user fault input is active)
OUTFNC _i -G.....	Position error exceeds maximum limit set with SMPER — servo axes only
OUTFNC _i -H.....	Output on position

* The "i" in the command syntax represents the number of the programmable input (e.g., OUTFNC3-H assigns onboard output 3 as an "output on position" function).

Thumbwheels

You can connect the controller's programmable I/O to a bank of thumbwheel switches to allow operator selection of motion or machine control parameters.

The commands that allow for thumbwheel data entry are:

```
INSTW..... Establish thumbwheel data inputs
OUTTW..... Establish thumbwheel data outputs
TW..... Read thumbwheels or PLC inputs
INPLC..... Establish PLC data inputs
OUTPLC..... Establish PLC data outputs
```

Thumbwheel Setup

The controller's programming language allows direct input of BCD thumbwheel data via the programmable inputs. Use the steps below to set up and read the thumbwheel interface. Refer to the *6K Series Command Reference* for descriptions of the commands used below.

- Step 1* Wire your thumbwheels to the 6K according to the I/O connection instructions provided in your product's *Installation Guide*.
- Step 2* Set up the inputs and outputs for operation with thumbwheels. The data valid input will be an input that the operator holds active to let the controller read the thumbwheels. This input is not necessary; however, it is often used when interfacing with PLCs.

```
OUTPLC1,1-4,0,12 ; Config PLC output set 1:
                  ; onboard outputs 1-4 are strobe outputs,
                  ; no output enable bit,
                  ; 12 ms strobe time per digit read.
INPLC1,1-8,9 ; Configure PLC input set 1:
              ; onboard inputs 1-8 are data inputs,
              ; onboard input 9 is a sign input,
              ; no data valid input.
INLVL000000000 ; Onboard inputs 1-9 configured active low
```

- Step 3* The thumbwheels are read sequentially by outputs 1-4, which strobe two digits at a time. The sign bit is optional. Set the thumbwheels to **+12345678** and type in the following commands:

```
VAR1=TW5 ; Assign data from all 8 thumbwheel digits to VAR1
VAR1 ; Displays the variable (*VAR1=+0.12345678).
      ; If you do not receive this response, return to
      ; step 1 and retry.
```

PLCs

The controller's programmable I/O can be connected to most PLCs. After defining and storing controller programs, the PLC typically executes programs, loads data, and manipulates inputs to the controller. The PLC instructs the controller to perform the motion segment of a total machine process.

Refer to your product's *Installation Guide* for instructions on connecting to I/O devices. For higher current or voltages above 24VDC, use external signal conditioning such as OPTO-22 compatible I/O signal conditioning racks. Contact your local distributor or automation technology center for information on these products.

PLC Scan Mode

The PLC feature allows you to create a pre-compiled program that mimics PLC functionality by scanning through the I/O faster than in a normal program run. Because the functions of PLC programs are pre-compiled, delays associated with command processing are eliminated during profile execution, allowing more rapid sequencing of actions than would be possible with programs that are not compiled. Command processing is then free to monitor other activities.

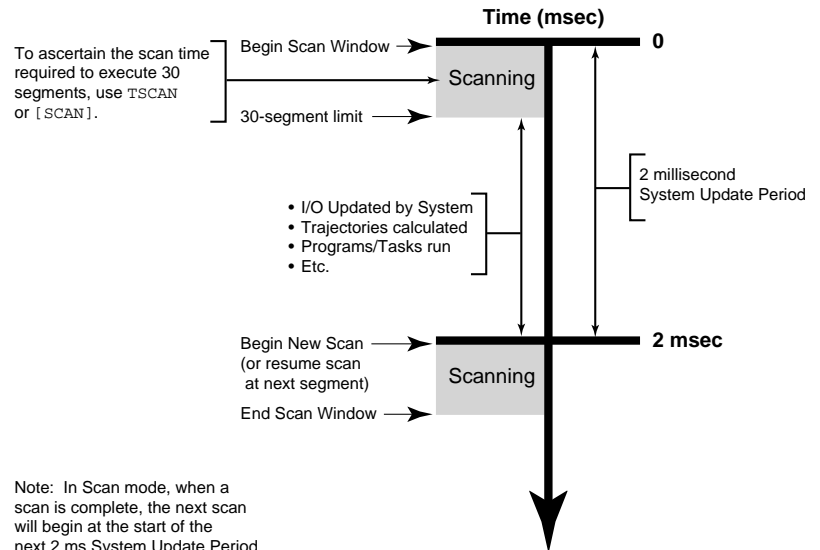
Scan Time

To check how much time (in 2 ms increments) the last scan took to complete, issue the `TSCAN` command or use the `SCAN` operand for variable assignments and `IF` conditions.

For example, if the last PLC program took 3 system updates (2 ms each) to scan, then `TSCAN` would report `*TSCAN6`, indicating that it took 6 ms to complete the scan. Each pass, if taking the same path through the conditional branches (`IF` statements), will always report the same `TSCAN` value.

The PLC program is scanned/ executed at the beginning of the 2-ms update period. The scan will stop after 30 segments have been executed and resume at the next 2-ms update. PLCP programs, when compiled, comprise a linked list of PLC segments. During a scan, each segment is counted until the total number of segments executed exceeds 30.

If the 30-segment limit occurs while executing a multi-segment statement, then that statement will finish no matter how many segments it executes. For example, if 29 segments have executed, then the next segment will cause the scan to pause until the next 2-ms update. If that next statement is `VARI1=1PE`, which executes 2 segments, then `VARI1=1PE` will complete its operations before pausing the scan.



To Implement a PLC Program ...

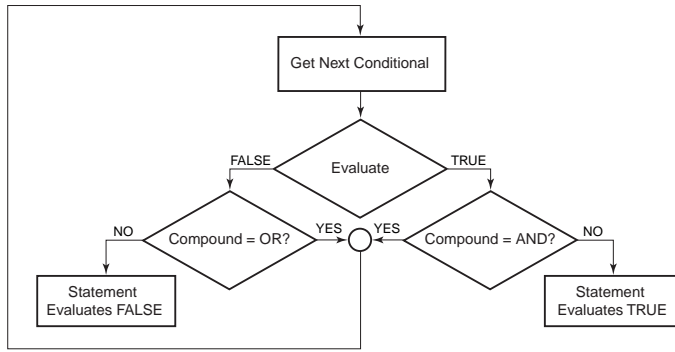
1. Define the PLC program (`DEF PLCPi` statement, followed by commands from the list below, followed by `END`). Up to 99 PLC programs can be defined, identified as `PLCP1`, `PLCP2`, `PLCP3`, and so on. Only these commands are allowed in a PLC program:
 - `IF`, `ELSE`, and `NIF` (conditional branching) —The PLC program uses the non-scaled integer (“raw”) operand values (e.g., `PE` value is not scaled by `SCLD` or `ERES`; `PANI` value is ADC counts, not volts). The only operands that are not allowed are: `SIN`, `COS`, `TAN`, `ATAN`, `VCVT`, `SQRT`, `VAR`, `TW`, `READ`, `DREAD`, `DREADF`, `DAT`, `DPTR`, and `PI`.
 - `L` and `LN` (loops)
 - `HALT` — as of OS revision 5.1
 - If the PLC program is executed with `SCANP`, `HALT` will terminate the current PLC program and kill the `SCANP` mode
 - If the PLC program is executed with `PRUN`, `HALT` will stop the PLC program (and the program that executed the `PRUN` statement) running in that task.
 - `BREAK` — as of OS revision 5.1
 - If the PLC program is executed with `SCANP`, `BREAK` will end only the current scan loop. At the next 2-millisecond update, the scan will restart at the first line of the PLC program.
 - If the PLC program is executed with `PRUN`, `BREAK` will stop the PLC program and execution will continue in the program from which the `PRUN` was executed (resuming at the command immediately following the `PRUN` command).
 - `TIMST` and `TIMSTP` (start and stop the timer) — available as of OS revision 5.1
 - `OUT` (turn on a digital output)

- ANO (set an analog output voltage – requires an extended I/O brick with on an analog output SIM) — available as of OS revision 5.1
 - EXE (execute a program in a specific task — e.g., 2%EXE MOVE)
 - PEXE (execute a compiled program in a specific task — e.g., 3%PEXE PLCP4)
 - VARI (integer variables). A VARI assignment expression is limited to one math function — either addition (+) or subtraction (-). The operands can be any of the monitored integer assignment operators (e.g., PE, PC, etc.).
NOTE: The PLC program uses the non-scaled (“raw”) operand values.
 - VARB (binary variables). Bitwise operations are limited to Boolean And (&), Boolean Inclusive Or (|), and Boolean Exclusive Or (^). The operands can be any of the monitored binary assignment operators (e.g., IN, LIM, AS, VARB, etc.).
2. Compile the PLC program (PCOMP PLCPi). A compiled program runs much faster than a standard program. After the PLCP program is compiled, it is placed in the 6K controller’s “compiled” memory partition (see MEMORY command). To verify which PLC programs are compiled, type in the TDIR command; compiled PLC programs are identified as “COMPILED AS A PLC PROGRAM”.
 3. Execute the PLC program (SCANP PLCPi). When the PLC program is launched with the SCANP command, it is executed in the “PLC Scan Mode”. The advantage of the PLC Scan Mode is that the PLC program is executed within a dedicated 0.5 ms time slot during every 2 ms system update period. This gives the PLCP program faster throughput for monitoring and manipulating I/O.

An alternative execution method is to use the PRUN command (PRUN PLCPi). This method is similar to the SCANP PLCPi method, but will only run through the PLCP program once.

Technical Notes About PLC Programs

- **Controller Status** bit 3 is set when a PLCP program is being executed in Scan Mode. To check the controller status, use SC, TSC, or TSCF.
- **Launching programs external to the scan:** Using the EXE command or the PEXE command, a scan program can launch another program in a specified task. EXE launches a standard, non-compiled program; PEXE launches a compiled program.
- **Stopping the scan:** The scan program can be stopped in either of two ways: using the !K command, or clearing the scan program by issuing a SCANP CLR command.
- **Timing the PLC program outputs:** It is not possible to control where the PLC program will pause if the scan takes more than the allowed time. This means that there can be a time lag of several update periods before the outputs, analog outputs, and/or variables affected by the PLC program are updated. The order in which the scan takes place should be considered when creating PLC programs to minimize the effects of such a lag. One way to avoid the lag is to create a binary variable as a temporary holding place for the desired output states. The last commands before the END statement of the PLC program can set the outputs according to the final status of the variable, such that all output states are written at the same time, just as the scan completes. This method is demonstrated in the example program below.
- **Memory Requirements:** Most commands allowed in a PLC program consume one segment of compiled memory after the program is compiled with PCOMP; the exceptions are VARI and VARB (each consume 2 segments) and IF statements. Each IF conditional evaluation compounded with either an AND or an OR operator consumes an additional segment (e.g., IF (IN.1=b1 AND 1AS.1=b0) consumes three segments of compiled memory). The number of compounds is limited only by the memory available.
- **Order of Evaluation for Conditional Expressions:**
Because only one level of parenthesis is allowed, the order of evaluation of IF conditionals is from left to right. Refer to the flowchart below for the evaluation logic.



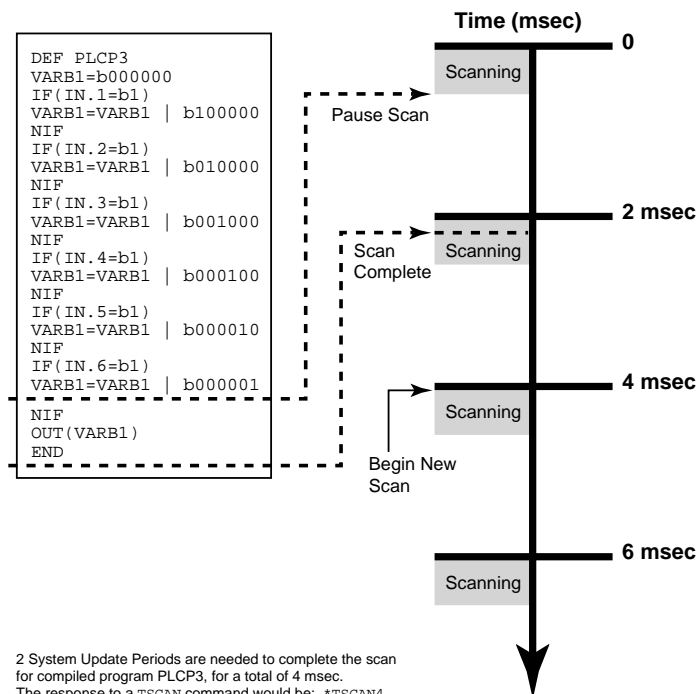
Programming Example

```

DEF PLCP3
; Binary states of outputs 1-6 represented by VARB1 bits 1-6.
; Outputs 1-6 set at the end of program.
VARB1=b000000      ; Initialize binary variable 1
IF(IN.1=b1)        ; If onboard input 1 is ON, turn output 1 ON
  VARB1=VARB1 | b100000 ; Set binary bit for output 1 only to ON
NIF
IF(IN.2=b1)        ; If onboard input 2 is ON, turn output 2 ON
  VARB1=VARB1 | b010000 ; Set binary bit for output 2 only to ON
NIF
IF(IN.3=b1)        ; If onboard input 3 is ON, turn output 3 ON
  VARB1=VARB1 | b001000 ; Set binary bit for output 3 only to ON
NIF
IF(IN.4=b1)        ; If onboard input 4 is ON, turn output 4 ON
  VARB1=VARB1 | b000100 ; Set binary bit for output 4 only to ON
NIF
IF(IN.5=b1)        ; If onboard input 5 is ON, turn output 5 ON
  VARB1=VARB1 | b000010 ; Set binary bit for output 5 only to ON
NIF
IF(IN.6=b1)        ; If onboard input 6 is ON, turn output 6 ON
  VARB1=VARB1 | b000001 ; Set binary bit for output 6 only to ON
NIF
OUT(VARB1)         ; Turn on appropriate onboard outputs
END

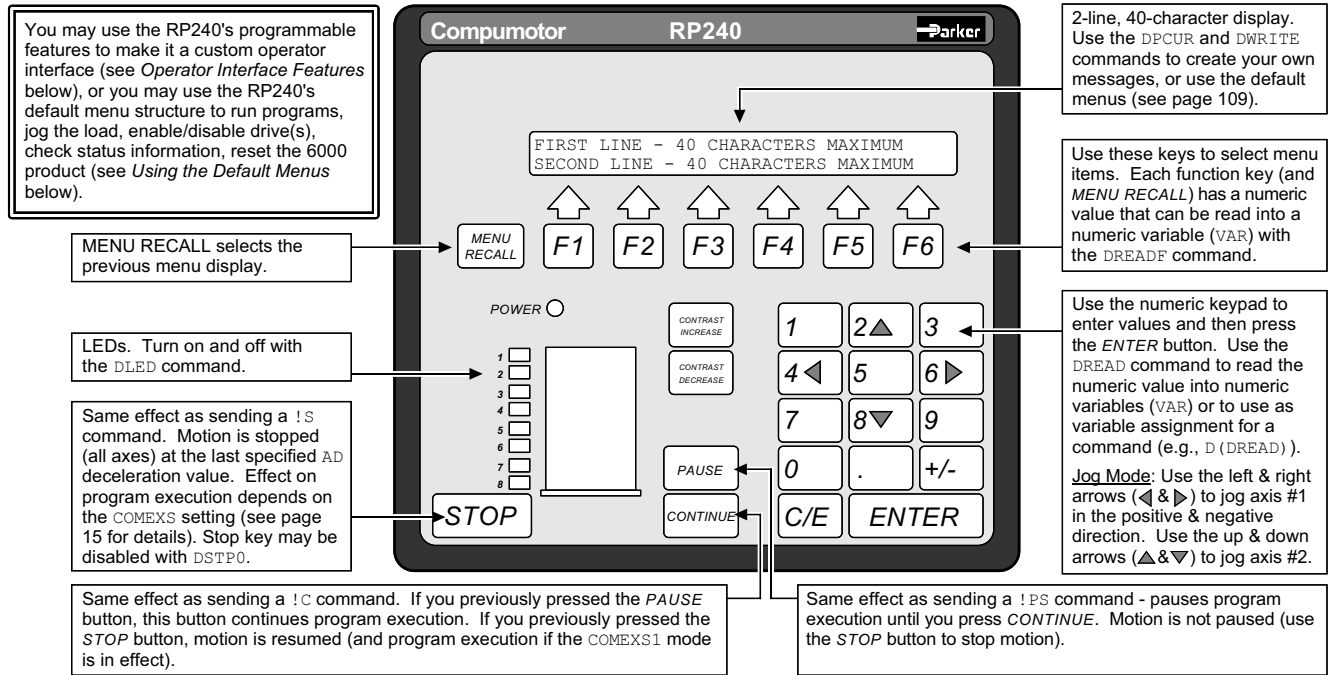
PCOMP PLCP3        ; Compile program PLCP3
SCANP PLCP3        ; Run compiled program PLCP3 in Scan mode. See diagram.

```



RP240 Remote Operator Panel

6K Series products are directly compatible with the Compumotor RP240 Remote Operator Panel. This section describes how to use your 6K product with the RP240. Instructions for connecting the RP240 are provided in the *6K Hardware Installation Guide*. Refer to the *Model RP240 User Guide* (p/n 88-012156-01) for information on RP240 hardware specifications, mounting guidelines, environmental considerations, and troubleshooting.



Configuration

NOTE

As shipped from the factory, your 6K product is configured to operate an RP240 from the **RS-232/485** connector (referred to as the "COM 2" port). This should be appropriate for the majority of applications requiring RP240 interface.

For more information on controlling your product's multiple serial ports, see page 37.

Every 6K Series product has two serial ports. The **RS-232** connector is referenced as the "COM1" serial port, and the **RS-232/485** connector is referenced as the "COM2" serial port.

To configure the 6K product's serial ports for use with the RP240 and/or 6K language commands, use the `DRPCHK` command. Be sure to select the affected serial port (**COM 1** or **COM 2**) with the `PORT` command before you execute the `DRPCHK` command. Once you issue the `DRPCHK` command, it is automatically saved in non-volatile memory. The configuration options are:

- `DRPCHK0`Use the serial port for 6K commands only (default for **COM 1**)
- `DRPCHK1`Check for RP240 on power up or reset. If detected, initialize RP240. If no RP240, use serial port for 6K commands.
- `DRPCHK2`Check for RP240 every 5-6 seconds. If detected, initialize RP240. Do not use port for 6K commands.
- `DRPCHK3`Check for RP240 on power up or reset. If detected, initialize RP240. If no RP240, use serial port for `DWRITE` command only. The `DWRITE` command can be used to transmit text strings to remote RS-232C devices. (default setting for **COM 2**)

Example **COM 2** is to be used for RS-485; **COM 1** is to be used for RP240, but the RP240 will be plugged in on an as-needed basis. The set-up commands for such an application should be executed in the following order:

```
PORT1           ; Select COM1 serial port for setup
DRPCHK2         ; Configure COM1 for RP240, periodic check
PORT2           ; Select COM2 serial port for setup
DRPCHKØ        ; Configure COM2 for 6K commands only
```

Operator Interface Features

The RP240 can be used as your product's *operator interface*, not a program entry terminal. As an operator interface, the RP240 offers the following features:

- Displays text and variables
- 8 LEDs can be used as programmable status lights
- Operator data entry of variables: read data from RP240 into variables and command value substitutions (see *Command Value Substitutions* on page 7).

Typically the user creates a program in the 6K controller to control the RP240 display and RP240 LEDs. The program can read data and make variable assignments via the RP240's keypad and function keys.

The 6K Series software commands for the RP240 are listed below. Detailed descriptions are provided in the *6K Series Command Reference*. The example below demonstrates the majority of these 6K Series commands for the RP240.

```
DCLEAR.....Clear The RP240 Display
DJOG.....Enter RP240 Jog Mode
[DKEY].....Numeric value of RP240 Key
DLED.....Turn RP240 LEDs On/Off
DPASS.....Change RP240 Password
DPCUR.....Position The Cursor On The RP240 Display
[DREAD] .....Read RP240 Data
[DREADF] .....Read RP240 Function Key
DREADI.....RP240 Data Read Immediate Mode
DRPCHK.....Check for RP240
DSTP.....Enable/Disable the RP240 Stop Key
DVAR.....Display Variable On RP240
DWRITE.....Display Text On The RP240 Display
```

*Programming
Example*

```
DEF panell ; Define program panell
REPEAT ; Start of repeat loop
DCLEAR0 ; Clear display
DWRITE"SELECT A FUNCTION KEY" ; Display text "SELECT A FUNCTION KEY"
DPCUR2,2 ; Move cursor to line 2 column 2
DWRITE"DIST" ; Display text "DIST"
DPCUR2,9 ; Move cursor to line 2 column 9
DWRITE"GO" ; Display text "GO"
DPCUR2,35 ; Move cursor to line 2 column 35
DWRITE"EXIT" ; Display text "EXIT"
VAR1 = DREADF ; Input a function key
IF (VAR1=1) ; If function key 1 hit
GOSUB panel2 ; GOSUB program panel2
ELSE ; Else
IF (VAR1=2) ; If function key 2 hit
DLED1 ; Turn on LED 1
GO1 ; Start motion on axis 1
DLEDO ; Turn off LED 1
NIF ; End of IF (VAR1=2)
NIF ; End of IF (VAR1=1)
UNTIL (VAR1=6) ; Repeat until VAR1=6 (function key 6)
DCLEAR0 ; Clear display
DWRITE"LAST FUNCTION KEY = F" ; Display text "LAST FUNCTION KEY = F"
DVAR1,1,0,0 ; Display variable 1
END ; End of panell

DEF panel2 ; Define prog panel2
DCLEAR0 ; Clear display
DWRITE"ENTER DISTANCE" ; Display text "ENTER DISTANCE"
D(DREAD) ; Enter distance number from RP240
END ; End of panel2
```

Using the Default Menus

On power-up, the 6K product will automatically default to a mode in which it controls the RP240 with the menu-driven functions listed below.

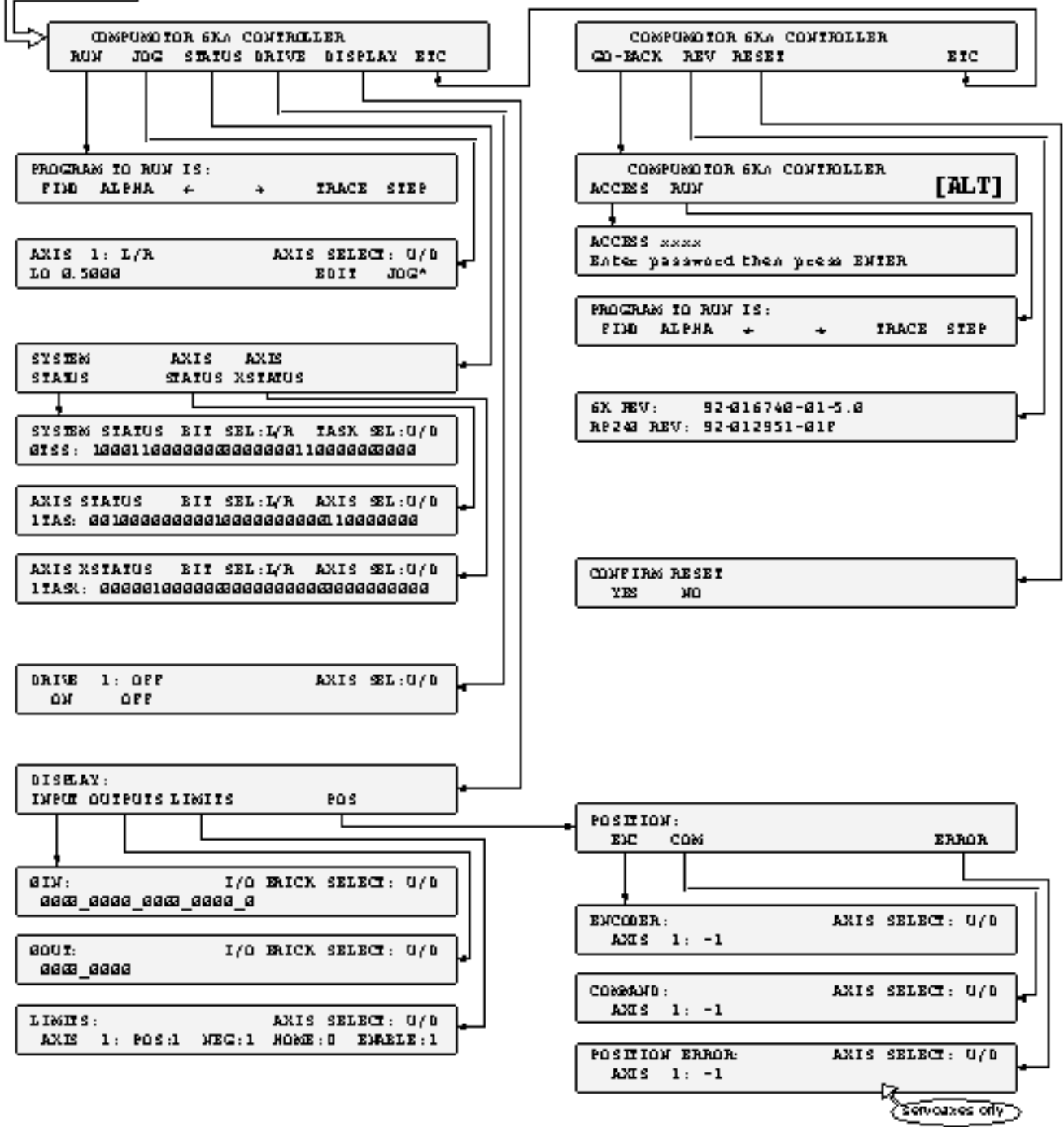
The flow chart below illustrates the RP240's menu structure in the default operating mode (when no 6K product user program is controlling the RP240). Press the **Menu Recall** key to back up to the previous screen. The menu functions are described in detail below.

- Run a stored program (run, stop, pause and continue functions)
- Jog the load
- Display the status of:
 - System (TSS), for each task
 - Axis (TAS)
 - Extended Axis (TASX)
 - I/O (TIN and TOUT)
 - Limits (TLIM) and ENABLE input (TINO bit 6)
 - Position: Commanded (TPC), Encoder (TPE)
 - Firmware revision levels for the 6K product (TREV) and the RP240
- Enable or disable the internal drive (DRIVE)
- Access RP240 menu functions with a security password (set with DPASS)
- Reset the 6K product (equivalent to the Reset command)

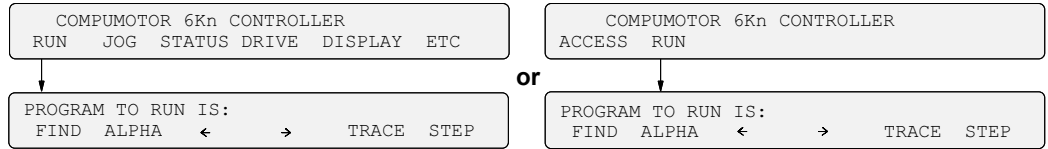
NOTE: To disable these menus, the start-up program (the program assigned with the STARTP command) must contain the DCLEARØ command.

Default
Power-up
Menu

- If you change the password with the DPASS command, the menu marked with [ALT] becomes the power-up menu. The default password is '8000' for the 6K controller.
- Arrows indicate the menu path when you press the corresponding function key below the menu item.
- To back up to the previous menu item, press the MENU RECALL button.



Running a Stored Program



After accessing the RUN menu, press **F1** to “find” the names of the programs stored in the 6K product’s memory; pressing **F1** repeatedly displays subsequent programs in the order in which they were stored in BBRAM. To execute the program, press the **ENTER** key.

To type in a program name at the location of the cursor, first select alpha or numeric characters with the **F2** function key (characters will be alpha if an asterisk appears to the right of ALPHA, or numeric if no asterisk appears). If alpha, press the up (2) or down (8) keys to move through the alphabet, if numeric, press the desired number key. Press **F3** to move the cursor to the left, or **F4** to move the cursor to the right.

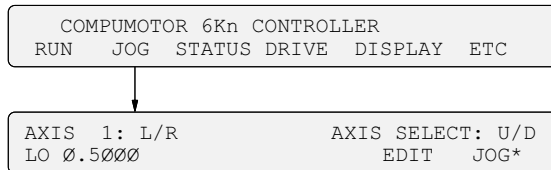
Only user programs defined with DEF and END can be executed from this menu. Compiled profiles (contouring and GOBUF profiles) cannot be executed from this menu; they must be executed from the terminal emulator with the PRUN command, or you can place the PRUN (name of path) command in a user program and then execute that program from this menu.

When a program is RUN and TRACE is selected (TRACE*), the RP240 display will trace all program commands as they are executed. This is different from the TRACE command in that the trace output goes to the RP240 display, not to a terminal via the serial port.

When a program is RUN and STEP is selected, step mode has been entered. This is similar to the STEP command, but when selected from the RUN menu the step mode allows single stepping by pressing the ENTER key. Both RP240 trace mode and step mode are exited when program execution is terminated.

⇒ **HINT:** If you wish to display each command as it is executed, select **STEP** and **TRACE** and press the **ENTER** key to step through the program.

Jogging



You can jog individual axes by pressing the arrow keys on the RP240’s numeric keypad. Pressing an arrow key on the numeric keypad will start motion and releasing the arrow key will stop motion. The up and down arrows keys are for selecting the axis to jog. The left and right arrow keys are for jogging the selected axis in the negative and positive direction, respectively.

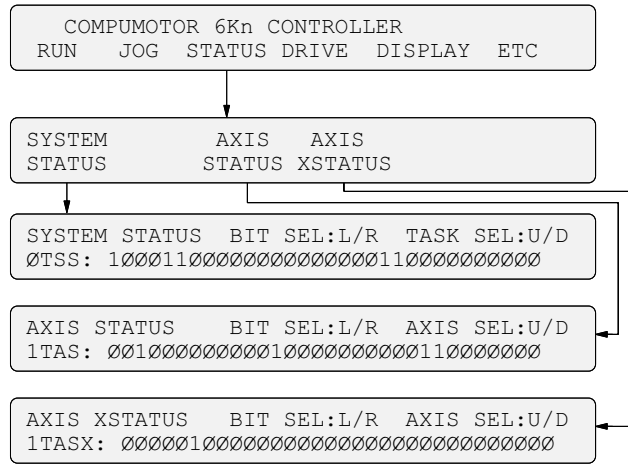
The HI and LO values in the jog menu represent the velocity in units of revs/sec (or volts/sec for ANI feedback). If scaling is enabled, the value is multiplied by the programmed SCLV value.

To edit the jog velocity* values:

1. Press the **F5** function key under EDIT (edit mode indicated with an asterisk).
2. Press the **F1** function key to select the HI and LO values (cursor appears under the first digit of the value selected).
3. Using the numeric keyboard, enter the value desired.
4. Repeat steps 2 and 3 for all values to be changed.
5. Press **ENTER** when finished editing.
6. To jog with the new velocity values, first press the **F6** function key (under JOG) to enable the arrow keys again.

Jog accel and decel values are specified by the JOGA and JOGAD commands, respectively.

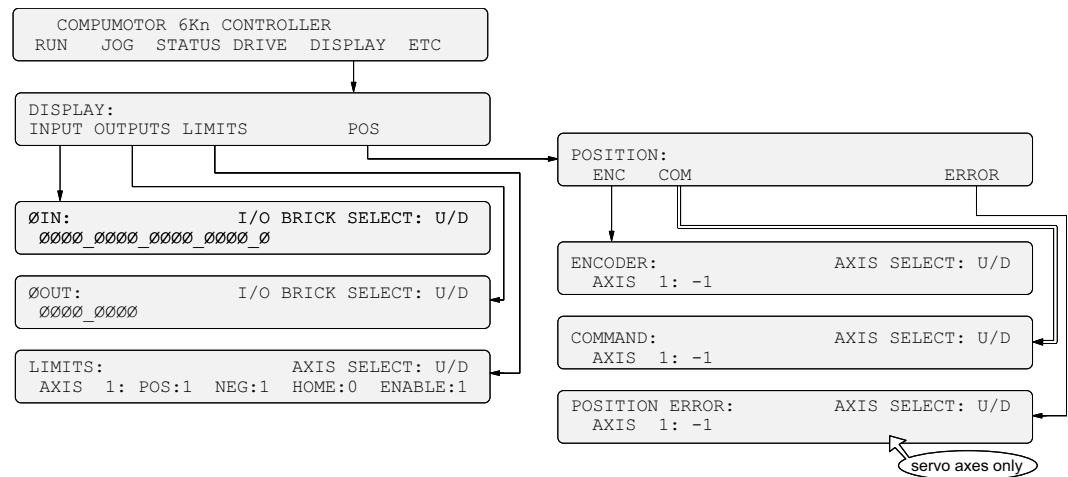
Status Reports:
System & Axis



After accessing the desired status menu, you can ascertain the function and status of each system (TSS, for each task) or axis (TAS and TASX) status bit by pressing the arrow keys on the numeric keypad.

To view a more descriptive explanation of each status bit (includes a text description), press the left or right arrow keys on the numeric keypad.

Status Reports:
I/O, Limits, Position



INPUTS Menu: This menu displays the TIN bit patterns for programmable inputs (onboard triggers and digital inputs on expansion I/O bricks). Remember that I/O bit patterns vary by product (see page 76 to find the bit patterns for your product). The initial menu show the trigger inputs status; use the up and down arrows to select the status of inputs on expansion I/O bricks.

OUTPUTS Menu: This menu displays the TOUT bit patterns for programmable outputs (onboard outputs and digital outputs on expansion I/O bricks). Remember that I/O bit patterns vary by product (see page 76 to find the bit patterns for your product). The initial menu show the onboard outputs; use the up and down arrows to select the status of outputs on expansion I/O bricks.

LIMITS Menu:

- The POS, NEG and HOME status items represent the hardware states of the limit inputs on the “LIMITS/HOME” connector, regardless of their LIMFNC input function assignments; they **do not** represent INFNC limit functions assigned to onboard trigger inputs or digital inputs on expansion I/O bricks.
- “POS” refers to the hardware end-of-travel limit imposed when counting in the positive

direction. “NEG” refers to the limit imposed when counting in the negative direction.

- A “1” indicates that the input is grounded, “0” indicates not grounded.

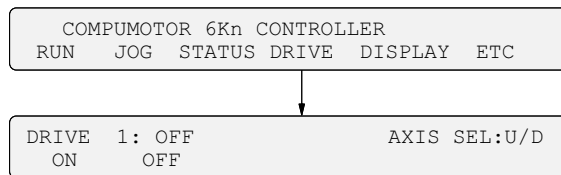
The end-of-travel limits (POS and NEG) must be grounded to allow motion (this is reversed if the active level is reversed with the LHLVL command).

The Enable input (ENABLE input terminal) must also be grounded before motion is allowed. When not grounded, the output (analog voltage or step pulse) to the drives is cut off and the shutdown outputs are activated.

POS Menu:

- The position values (encoder, commanded, and position error) shown are continually updated.
- The position error (“ERROR”) report is applicable only to servo axes.
- Position values are subject to the SCLD scaling factor (if scaling is enabled—SCALE1), PSET offset value, encoder polarity (ENCPOL), and commanded direction polarity (CMDDIR).

Enabling and Disabling the Drive(s)

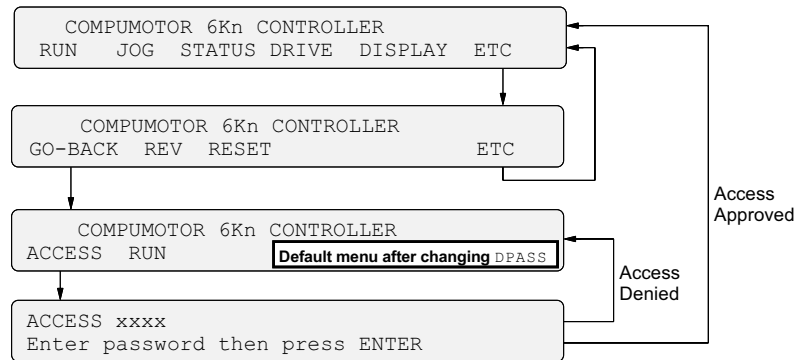


In the DRIVE menu, the current status of the drive(s) is displayed. To enable or disable the drive, press **F1** or **F2**, respectively. This menu offers the same functionality as the DRIVE command.

WARNING

Shutting down a rotary drive system allows the load to freewheel if there is no brake installed.

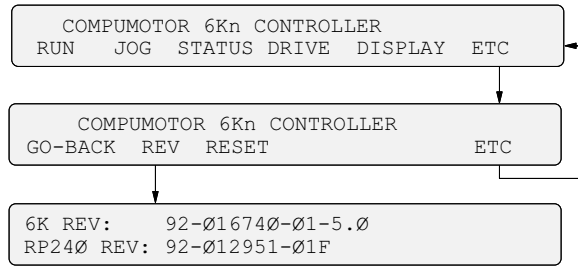
Access Security



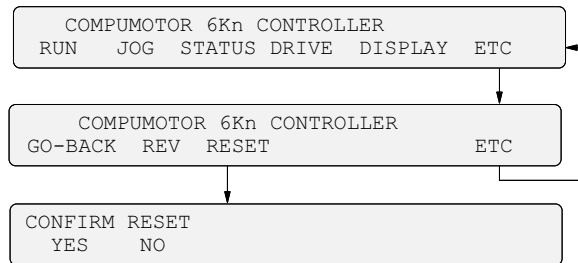
If the RP240 password is modified with the DPASS command to be other than the default (see *Changing the Password* below) the ACCESS menu then becomes the new default menu after power-up or executing a RESET. After that, the new password must then be entered to access the original default menu (see “Access Approved” path in illustration). If the operator does not know the new password, all he or she can do is run programs stored in the 6K product (RUN).

Changing the Password: The default password for 6K products is “6000”. A new password (numeric value of up to 4 characters) can be established with the DPASS command. For example, the DPASS2001 command sets the password to 2001.

Revision Levels



Resetting the 6K Product



After accessing the RESET menu, press the **F1** key to execute a reset (or press **F2** to cancel and exit the menu). The reset is identical to issuing a RESET command or cycling power to the 6K product. If a start-up program has been assigned with the STARTP command, that program will be executed.

CAUTION

Executing a reset will restore most command values (exclusions: see page 33) to their factory default setting.

Joystick Control, Analog Inputs



Refer to your
Installation Guide for
connection
procedures.

The 6K allows you to add analog input (ANI) SIMs to the expansion I/O bricks (sold separately). Each ANI SIM provides 8 analog inputs. The default voltage range for these input is -10VDC to +10VDC, but other ranges are selectable with the ANIRNG command. ANI inputs can be used in a variety of ways:

- Control an axis with a joystick (see *Joystick Control*)
- Position feedback for servo axes. The position of the ANI inputs can be read using the PANI or TPANI commands.
- Use the voltage value to control other events. The voltage value on the ANI inputs can be read using the ANI or TANI commands.

Joystick Control

The 6K controller supports joystick operation with digital inputs and analog inputs. The digital inputs include the onboard limit inputs and trigger inputs, as well as digital input SIMs on an external I/O brick. The 12-bit analog inputs are available only if you install an analog input SIM on an external I/O brick (default voltage range is -10V to +10V, selectable with ANIRNG).

To Set Up Joystick Operation

(refer also to the example code below)

1. Select the required digital inputs and analog inputs required for joystick operation. Connect the joystick as instructed in your controller's *Installation Guide*.
2. Assign the appropriate input functions to the digital inputs used for joystick's operation:
 - Release Input: `INFNCi-M` for triggers & I/O brick inputs, or `LIMFNCi-M` for limit inputs.
 - Axis Select Input: `INFNCi-N` for triggers & I/O brick inputs, or `LIMFNCi-N` for limit inputs. **NOTE:** If you're not using this input, assign the analog inputs to the axes with the `JOYAXH` command.
 - Velocity Select Input: `INFNCi-O` for triggers & external inputs, or `LIMFNCi-O` for limit inputs.
3. (optional) Use the `ANIRNG` command to select the voltage range for the analog inputs you will use. The default range is -10VDC to +10VDC (other options are 0 to +5V, -5 to +5V, and 0 to +10V).
4. Assign analog inputs to control specific axes, using:
 - `JOYAXH`: Standard analog input-to-axis assignment.
 - `JOYAXL` (optional). Analog input-to-axis assignment when the Axis Select Input is low.
5. Define the joystick motion parameters:
 - Maximum Velocity when Velocity Select input switch is open/high (`JOYVH` command). If the Velocity Select input is not used, joystick motion always uses the `JOYVH` velocity.
 - Maximum Velocity when Velocity Select input switch is closed/low (`JOYVL` command).
 - Accel (`JOYA` command).
 - Accel for s-curve profiling (`JOYAA` command).
 - Decel (`JOYAD` command).
 - Decel for s-curve profiling (`JOYADA` command).
6. Define the usable voltage zone for your joystick:
(make sure you have first assigned the analog inputs – see step 3 above)
 - End Deadband (`JOYEDB`): Defines the voltage offset (from the -10V & +10V endpoints) at which maximum velocity occurs. Default is 0.1V, maxing voltage at -9.9V and +9.9V.
 - Center Voltage (`JOYCTR` or `JOYZ`): Defines the voltage when the joystick is at rest to be the zero-velocity center. Default `JOYCTR` setting is 0V.
 - Center Deadband (`JOYCDB`): Defines the zero-velocity range on either side of the Center Voltage. Default is 0.1V, setting the zero-velocity range at -0.1V to +0.1V.
7. To jog the axes:
 - a. In your program, enable Joystick Operation with the `JOY` command (Joystick Release input must be closed in order to enable joystick mode). When the `JOY` command enables joystick mode for the affect axes, program execution stops on those axes (assuming the Continuous Command Execution Mode is disabled with the `COMEXCØ` command).
 - b. Move the load with the joystick.
 - c. When you are finished, open the Joystick Release input to disable joystick mode. This allows program execution to resume with the next statement after the initial `JOY` command that started the joystick mode.

Joystick Programming Example
(refer also to the illustration below)

Application Requirements: This example represents a typical two-axis joystick application in which a high-velocity range is required to move to a region, then a low-velocity range is required for a fine search. After the search is completed it is necessary to record the load positions, then move to the next region. A digital input can be used to indicate that the position should be read. The Joystick Release input is used to exit the joystick mode and continue with the motion program.

Hardware Configuration:

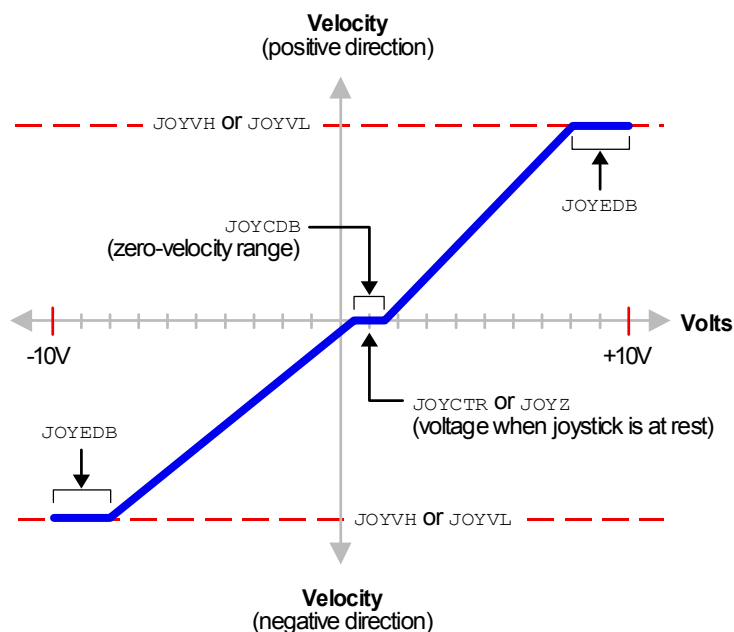
- An analog input SIM is installed in the 3rd slot of I/O brick 1. The eight analog inputs (1-8) are addressed as input numbers 17-24 on the I/O brick. Analog input 17 will control axis 1, and analog input 18 will control axis 2.
- A digital input SIM is installed in the 1st slot of I/O brick 1. The eight digital inputs (1-8) are addressed as input numbers 1-8 on the I/O brick. Digital input 6 will be used for the Joystick Release function, and input 7 will be used for the Joystick Velocity Select input. Input 8 will be used to indicate that the position should be read.

Setup Code (the drawing below shows the usable voltage configuration):

```

1INFC7-M           ; Assign Joystick Release f(n) to brick 1, input 7
1INFC8-O           ; Assign Joystick Velocity Select f(n) to brick 1, input 8
JOYAXH1-17,1-18   ; Assign analog input 17 to control axis 1,
                  ; Assign analog input 18 to control axis 2
JOYVH1,1           ; Max. velocity on axes 1 & 2 is 10 units/sec when the
                  ; Velocity Select input (1IN.7) is open (high)
JOYVL10,10        ; Max. velocity on axes 1 & 2 is 1 unit/sec when the
                  ; Velocity Select input (1IN.7) is closed (low)
JOYA100,100       ; Set joystick accel to 100 units/sec/sec on both axes
JOYAD100,100      ; Set joystick decel to 100 units/sec/sec on both axes
;**** COMMANDS TO SET UP USABLE VOLTAGE: ****
1JOYCTR.17=+1.0   ; Set center voltage for analog input 17 (controls axis 1)
1JOYCTR.18=+1.0   ; and 18 (controls axis 2) to+1.0V. The +1.0V value was
                  ; ascertained by checking the voltage of the both
                  ; inputs (with the 1TANI.17 and 1TAIN.18 commands)
                  ; when the joystick was at rest.
1JOYCDB.17=0.5    ; Set center deadband to compensate for the fact that
1JOYCDB.18=0.5    ; when the joystick is at rest, the voltage received on
                  ; both analog inputs can fluctuate +/- 0.5V on either
                  ; side of the +1.0V center.
1JOYEDB.17=2.0    ; Set end deadband to compensate for the fact that the
1JOYEDB.18=2.0    ; joystick can produce only -8.0V to +8.0V.
;*****
JOY11              ; Enable joystick mode for axes 1 & 2

```



Analog Input Interface

*Refer to your product's
Installation Guide for
ANI connection
information.*

Using analog (ANI) inputs on expansion I/O bricks, your 6K Series controller can use analog voltage as position feedback (servo axes only) and simply as a means to control the program based on external conditions.

Each ANI SIM on an expansion I/O brick provides eight analog inputs. The 12-bit analog inputs have a default voltage range of -10VDC to +10VDC. Other ranges are selectable with the ANIRNG command (0 to +5VDC, 0 to +10VDC, and -5 to +5VDC). The voltage value of the ANI inputs can be transferred to the terminal with the TANI command, or used in an assignment or comparison operation with the ANI operator (e.g., `IF (1ANI < 2.4)`).

When used for position feedback, the position counter resolution is 205 counts/volt (under the default ANIRNG setting). The position value of the ANI inputs can be transferred to the terminal with the TPANI command, or used in an assignment or comparison operation with the PANI operator (e.g., `WAIT (1PANI > 421)`).

Three common applications of the ANI inputs are:

- Position command to the control loop (e.g., as a master axis in Following mode)
- Position feedback to the control loop (see page 67 to select ANI input feedback for servo axes)
- A force or torque feedback signal

Programming Example

The portion of 6K code below (for two axes of control) demonstrates how to read the analog inputs into the controller and set the commanded analog output of each axis to that value. If you have a torque drive, this provides open-loop torque control.

```
SGP0,0          ; Turn off servo proportional feedback gain
SGI0,0          ; Turn off servo integral feedback gain
SGV0,0          ; Turn off servo velocity feedback gain
SGAF0,0         ; Turn off servo acceleration feedforward gain
SGVF0,0         ; Turn off servo velocity feedforward gain
SOFFS0,0        ; Set offset to zero (analog output will be 0 volts)
L               ; Enter an infinite loop
  VAR1=2ANI.17   ; Read value of 1st analog input on SIM slot 3
                  ; (I/O point 17) on I/O brick 2 on into VAR variable 1
  VAR1=2ANI.18   ; Read value of 2nd analog input on SIM slot 3
                  ; (I/O point 18) on I/O brick 2 on into VAR variable 2
  SOFFS(VAR1), (VAR2) ; Assign voltages from 2ANI.17 and 2ANI.18 to the
                  ; analog output for axes 1 & 2, respectively
  T.01          ; Set time delay to 10 milliseconds
LN              ; End loop
```

ANI as a Feedback Device

The ANI analog inputs, when selected as a feedback source with the SFB command, is assumed to provide position information. With this feedback it is possible to solve applications that require positioning to a voltage, rather than positioning to a known position. Some example applications are as follows:

- Using a potentiometer as feedback (mechanical motion is mimicked by the 6K controller)
- Maintaining a force while position changes due to fluid evacuating a chamber
- Opening or closing a valve as another process changes

Host Computer Interface

Another choice for product control is to use a host computer and execute a motion program using the serial interface (RS-232 or RS-485). A host computer can be used to run a motion program interactively from a BASIC or C program (high-level program controls the 6K product and acts as a user interface). A BASIC program example (for the 6K2 product) is provided below.

```
10 '      6K Series Serial Communication BASIC Routine
12 '
14 '
16 ' *****
18 '
20 ' This program will set the communications parameters for the
22 ' serial port on a PC to communicate with a 6K series
24 ' stand-alone product.
26 '
28 ' *****
30 '
100 '*** open com port 1 at 9600 baud, no parity, 8 data bits, 1 stop bit
110 '*** activate Request to Send (RS), suppress Clear to Send (CS), suppress
120 '*** DATA set ready (DS), and suppress Carrier Detect (CD) ***
130 OPEN "COM1:9600,N,8,1,RS,CS,DS,CD" FOR RANDOM AS 1
140 '
150 '*** initialize variables ***
160 MOVE$ = ""      ' *** commands to be sent to the product ***
170 RESPONSE$ = ""  ' *** response from the product ***
180 POSITION$ = ""   ' *** feedback position reported ***
190 SETUP$ = ""    ' *** setup commands ***
200 '
210 '*** format the screen and wait for the user to start the program ***
220 CLS : LOCATE 12, 20
230 PRINT "Press any key to start the program"
240 '
250 '*** wait for the user to press a key ***
260 PRESS$ = INKEY$
270 IF PRESS$ = "" THEN 260
280 CLS
290 '
300 '*** set a pre-defined move to make ***
310 SETUP$ = "ECHO1:ERRLVLO:LH0,0:"
320 MOVE$ = "A100,100:V2,2:D50000,50000:GO11:TFB:"
330 '
340 '
400 '*** send the commands to the product ***
410 PRINT #1, SETUP$
420 PRINT #1, MOVE$
430 '
500 ' *** read the response from the TFB command ***
510 ' *** the controller will send a leading "+" or "-" in response to the TFB command to
520 ' *** indicate which direction travel has occurred. ***
530 WHILE (RESPONSE$ <> "+" AND RESPONSE$ <> "-") ' *** this loop waits for the "+"
540     RESPONSE$ = INPUT$(1, #1) ' *** or "-" characters to be returned
550 WEND ' *** before reading the position ***
560 '
570 WHILE (RESPONSE$ <> CHR$(13)) ' *** this loop reads one character at a time
580     POSITION$ = POSITION$ + RESPONSE$ ' *** from the serial buffer until a carriage
590     RESPONSE$ = INPUT$(1, #1) ' *** return is encountered ***
600 WEND
610 '
620 '*** print the response to the screen ***
630 LOCATE 12, 20: PRINT "Position is " + POSITION$
640 '
650 'END
```