

D		Distance	Product	Rev
Type	Motion		6K	5.0
Syntax	<!><@><a>D<r>,<r>,<r>,<r>,<r>,<r>,<r>,<r>			
Units	r = distance units (scalable by SCLD)			
Range	±999,999,999.99999			
Default	4000			
Response	D: *D+4000,+4000,+4000,+4000 ... 1D: *1D+4000			
See Also	[D], GO, MA, MC, PSET, SCLD, TSTAT			

The Distance (D) command defines either the number of units the motor will move or the absolute position it will seek after a GO command. In the incremental mode (MAØ), the distance value represents the total number of units you wish the motor to move. In the absolute mode (MA1) the distance value represents the absolute position the motor will end up at; the actual distance traveled will vary depending on the absolute position of the motor before the move is initiated.

In the incremental mode (MAØ), you can specify a negative distance by placing a dash or hyphen (-) in front of the distance value (e.g., D-10000). Otherwise, the direction is considered positive. You can change direction without changing the distance value by using the +, -, or ~ operators (e.g. D+, +, +, or D-, -, -, or D~, ~, ~); the tilde (~) is a means of toggling the direction.

The distance remains set until you change it with a subsequent distance command. Distances outside the valid range are flagged as an error, returning the message *INVALID DATA-FIELD x, where x is the field number.

UNITS OF MEASURE and SCALING: refer to page 16.

ON-THE-FLY CHANGES: You can change distance *on the fly* (while motion is in progress) in two ways. One way is to send an immediate distance command (!D) followed by an immediate go command (!GO). The other way is to enable the continuous command execution mode (COMEXC1) and execute a buffered distance command (D) followed by a buffered go command (GO).

Example:

```

DEL proga           ; Delete program called proga
DEF proga           ; Begin definition of program called proga
MA0000             ; Incremental index mode for all axes
MC0000             ; Preset index mode for all axes
A10,12,1,2         ; Set the acceleration to 10, 12, 1, and 2 units/sec/sec for
                   ; axes 1, 2, 3 and 4, respectively
AD1,1,1,2          ; Set the deceleration to 1, 1, 1, and 2 units/sec/sec for
                   ; axes 1, 2, 3 and 4, respectively
V1,1,1,2           ; Set the velocity to 1, 1, 1, and 2 units/sec for
                   ; axes 1, 2, 3 and 4, respectively
D100000,1000,10,100 ; Set the distance to 100000, 1000, 10, and 100 units
                   ; for axes 1, 2, 3 and 4, respectively
GO1100             ; Initiate motion on axes 1 and 2, 3 and 4 do not move
END                ; End definition of program called proga

```

[D] Distance Assignment

Type	Assignment or Comparison	Product	Rev
Syntax	See below	6K	5.0
Units	distance units (scalable by SCLD)		
Range	±999,999,999.99999		
Default	n/a		
Response	n/a		
See Also	D, GO, MA, MC, PSET, SCLD		

The distance assignment (D) command is used to compare the programmed distance value to another value or variable, or to assign the current programmed distance to a variable.

Syntax: VARn=aD where n is the variable number, and a is the axis number, or [D] can be used in an expression such as IF(1D<25000). When assigning the distance value to a variable, an axis specifier must always precede the D command (e.g., VAR1=1D) or it will default to axis 1. When making a comparison to the programmed distance, an axis specifier must also be used (e.g., IF(1D<20000)). The D value used in any comparison, or in any assignment statement is the programmed D value. If the actual position information is required, refer to the PC command for steppers, or the PE or ANI commands for servos.

UNITS OF MEASURE and SCALING: refer to page 16.

Example:

```
IF(2D<25000)      ; If the programmed distance on axis 2 is less than 25000 units,
                  ; then do the statements between the IF and NIF
VAR1=2D*2         ; Variable 1 = programmed distance of axis 2 times 2
D,(VAR1)          ; Set the distance on axis 2 to the value of variable 1
NIF               ; End the IF statement
```

[DAC] Value of DAC Output

Type	Assignment or Comparison	Product	Rev
Syntax	See below	6K	5.0
Units	Volts		
Range	-10.000 to +10.000		
Default	n/a		
Response	n/a		
See Also	DACLIM, SOFFS, TDAC		

Use the DAC command to compare the value of the DAC (commanded analog control signal output) to another value or variable, or to assign the value of the DAC to a variable.

Syntax: VARn=aDAC where “n” is the variable number, and “a” is the axis number, or DAC can be used in an expression such as IF(1DAC<6). An axis specifier must precede the DAC command, or it will default to axis 1 (e.g., VAR1=1DAC, IF(1DAC<2), etc.).

Example:

```
VAR6=2DAC         ; Set variable #6 equal to the DAC voltage output to axis #2
IF(2DAC>5.0)      ; If the DAC voltage to axis #2 is > 5V, do the IF statement.
TDAC              ; Transfer the current DAC values
NIF               ; End IF statement
```

DACLIM Digital-to-Analog Converter (DAC) Limit

Type	Servo; Controller Setup	Product	Rev
Syntax	<!><@><a>DACLIM<r>, <r>, <r>, <r>, <r>, <r>, <r>, <r>	6K	5.0
Units	r = volts		
Range	0.000 to 10.000		
Default	10.000		
Response	DACLIM: *DACLIM10.000,10.000,10.000,10.000 ... 1DACLIM: *1DACLIM10.00		
See Also	[DAC], SOFFS, TDAC		

This command sets the maximum absolute value the commanded analog control signal output can achieve. For example, setting the DAC limit to 8.000V (DACLIM8.000) will clamp the DAC output range from -8.000 to +8.000. Use the TDAC command to verify the voltage being commanded at the servo controller's analog output.

Example:

```
DACLIM7.000,9.000 ; Axis #1 DAC output is limited to -7.000 to +7.000 volts;  
                  ; Axis #2 DAC output is limited to -9.000 to +9.000 volts
```

[DAT] Data Assignment

Type	Data Storage	Product	Rev
Syntax	DATi	6K	5.0
Units	i = data program #		
Range	1-50		
Default	n/a		
Response	n/a		
See Also	DATA, [DATP], DATPTR, DATRST, DATTC		

The Data Assignment (DAT) command recalls data from the data program (DATP). The data is loaded into a command field, or into a variable (VAR). As the data is loaded, the internal data pointer to the DATP data increments and points to the next datum for the next DAT command.

Syntax: VARn=DATi where “n” is the variable number, and “i” is the data program number, or DAT can be used as a command argument such as A(DAT1), 5, 4, 10

If the data is to be loaded into a command field, the DAT command must be placed within parentheses (e.g., AD(DAT2), 3, 4, 5). If the data is loaded into a variable, parentheses are not required. (e.g., VAR1=DAT2).

Rule of Thumb for command value substitutions: If the command syntax shows that the command field requires a real number (denoted by <r>) or an integer value (denoted by <i>), you can use the DAT substitution (e.g., HOMV2, 1, (DAT1)).

The DAT command cannot be used in an expression, such as IF(DAT2 < 5) or VAR1=1 + DAT3.

Example: Refer to the Reset Data Pointer (DATRST) command example.

DATA Data Statement

Type	Data Storage	Product	Rev
Syntax	<!>DATA=<r>, <r>, <r>, <r>	6K	5.0
Units	r = data value		
Range	±999,999,999.99999999		
Default	n/a		
Response	n/a		
See Also	[DAT], [DATP], DATPTR, DATRST, DATTC, MEMORY		

The Data Statement (DATA) command is used only in the data programs (DATP) to identify the data statements. The DATA command is followed by an equal sign (=), and a maximum of four data values. The maximum number of data statements is limited only by the amount of memory available.

Example: Refer to the Reset Data Pointer (DATRST) command example.

[DATP] Data Program

Type	Data Storage	Product	Rev
Syntax	DATPi	6K	5.0
Units	i = data program #		
Range	1-50		
Default	n/a		
Response	n/a		
See Also	[DAT], DATA, DATPTR, DATRST, DATSIZ, DATTC, MEMORY		

DATP is not a command, but is the name of the program that is the default for storing data. Fifty such data programs can be created, DATP1 - DATP50. The program is defined with the DEF command, just as any other program would be, but only the DATA and END commands are allowed within the program definition. DATPi will contain the array of data to be recalled by the DATi command. Upon completion of the definition, the internal data pointer is pointing to the first datum in the data program.

Example:

```
DEF DATP5          ; Define data program 5
DATA=1,2,3,4      ; Enter data
DATA=5.62,6.52,7.12,8.47 ; Enter data
END              ; End program definition
A(DAT5)          ; Load data from data program 5 and store in axis 1 acceleration.
                ; Axis 1 acceleration = 1
V(DAT5)          ; Load data from data program 5 and store in axis 1 velocity.
                ; Axis 1 velocity = 2
D(DAT5)          ; Load data from data program 5 and store in axis 1 distance.
                ; Axis 1 distance = 3
A,(DAT5)         ; Load data from data program 5 and store in axis 2 acceleration.
                ; Axis 2 acceleration = 4
A,,(DAT5)        ; Load data from data program 5 and store in axis 3 acceleration.
                ; Axis 3 acceleration = 5.62
```

DATPTR Set Data Pointer

Type	Data Storage	Product	Rev
Syntax	<!>DATPTRi,i,i	6K	5.0
Units	n/a		
Range	1st i = program # 1 to 50 2nd i = data element # 1 to 6500 3rd i = increment setting of 1 to 100		
Default	1,1,1		
Response	n/a		
See Also	[DAT], DATA, [DATP], DATSIZ, DATTC, [DPTR], TDPTR		

The Set Data Pointer (DATPTR) command moves the internal data pointer to a specific data element in the specified data program (DATPi). This command also establishes the number of data elements by which the pointer increments after writing each data element from a DATTC command, or after recalling a data element with the DAT command.

The data program selected with the first integer in the DATPTR command becomes the active data program. Subsequent DATTC, TDPTR, and DPTR commands will reference the active data program. You can use the TDPTR command to ascertain the current active data program, as well as the current location of the data pointer and the increment setting (see TDPTR command description for details).

The DPTR command can be used to compare the current pointer location (the number of the data element to which the data pointer is pointing) against another value or variable, or to assign the pointer location number to a variable.

As an example, suppose data program #1 (DATP1) is configured to hold 15 data elements (DATSIZ1, 15), the data pointer is configured to start at the first data element and increment 1 data element after every DATTC value is stored (DATPTR1, 1, 1), and the values of numeric variables #1 through #4 are already assigned (VAR1=2, VAR2=4, VAR3=8, VAR4=64). If you then enter the DATTC1, 2, 3, 4 command, the values of VAR1 through VAR4 will be assigned respectively to the first four data elements in the data program and the pointer will stop at data element #5. The response to the TPROG DATP1 command would be

as depicted below (the text is highlighted to illustrate the location of the data pointer after the DATTCH1,2,3,4 command is executed). The response to the TDPTR command would be *TDPTR1,5,1.

```
*DATA=2.0,4.0,8.0,64.0
*DATA=0.0,0.0,0.0,0.0
*DATA=0.0,0.0,0.0,0.0
*DATA=0.0,0.0,0.0
```

Once you have stored (taught) the variables to the data program, you can use the DATPTR command to point to the data elements and then use the DAT data assignment command to read the stored variables to your motion program.

During the process of writing data (DATTCH) or recalling data (DAT), if the pointer reaches the last data element in the program, it automatically wraps around to the first datum in the program and a warning message is displayed (*WARNING: POINTER HAS WRAPPED AROUND TO DATA POINT 1). This warning will not interrupt command execution.

Example: (See Also: DATSIZ command)

```
DEL DATP5          ; Delete data program #5 (DATP5)
DEF DATP5          ; Define data program #5 (DATP5)
DATA=1,2,3,4      ; Enter data
DATA=5.62,6.52,7.12,8.47 ; Enter data
END                ; End program definition
A(DAT5)           ; Load data from DATP5 and store in axis 1 acceleration.
                  ; Axis 1 acceleration = 1
V(DAT5)           ; Load data from DATP5 and store in axis 1 velocity.
                  ; Axis 1 velocity = 2
D(DAT5)           ; Load data from DATP5 and store in axis 1 distance.
                  ; Axis 1 distance = 3
DATPTR5,1,1       ; Set the data pointer to datum 1 in DATP5; increment the
                  ; pointer by one after each DAT command
A,(DAT5)          ; Load data from DATP5 and store in axis 2 acceleration.
                  ; Axis 2 acceleration = 1
A,,(DAT5)         ; Load data from DATP5 and store in axis 3 acceleration.
                  ; Axis 3 acceleration = 2
```

DATRST Reset Data Pointer

Type	Data Storage	Product	Rev
Syntax	<!>DATRST<i>,<i>	6K	5.0
Units	n/a		
Range	1st i = program # 1 to 50, 2nd i = data element # 1 to 6500		
Default	n/a		
Response	n/a		
See Also	[DAT], DATA, [DATP]		

The Reset Data Pointer (DATRST) command sets the internal data pointer to a specific data element in a data program (DATP<i>). As data is recalled from a data program with the DAT command, the pointer automatically increments to the next data element. If the pointer reaches the end of the program, it automatically wraps around to the first data element in the program. DATRST allows the pointer to be set to any location within the data program (DATP).

Example:

```
DEF DATP5          ; Define data program 5
DATA=1,2,3,4      ; Enter data
DATA=5.62,6.52,7.12,8.47 ; Enter data
END                ; End program definition
A(DAT5)           ; Load data from data program 5 and store in axis 1 acceleration.
                  ; Axis 1 acceleration = 1
V(DAT5)           ; Load data from data program 5 and store in axis 1 velocity.
                  ; Axis 1 velocity = 2
D(DAT5)           ; Load data from data program 5 and store in axis 1 distance.
                  ; Axis 1 distance = 3
DATRST5,1         ; Set the data pointer to datum 1 in data program 5
A,(DAT5)          ; Load data from data program 5 and store in axis 2 acceleration.
                  ; Axis 2 acceleration = 1
A,,(DAT5)         ; Load data from data program 5 and store in axis 3 acceleration.
                  ; Axis 3 acceleration = 2
```

DATSIZ Data Program Size

Type	Data Storage	Product	Rev
Syntax	<!>DATSIzi<,i>	6K	5.0
Units	n/a		
Range	1st i = program # 0 - 50 (0 = disable) 2nd i = data element # 1 - 6500		
Default	0,1		
Response	n/a		
See Also	[DAT], DATPTR, [DATP], DATTCH		

The Data Program Size (DATSIz) command creates a new data program (DATP) and establishes the number of data elements the data program contains.

The DATSIz command syntax is DATSIzi<,i>. The first integer (i) represents the number of the data program (1 - 50). You can create up to 50 separate data programs. The data program is automatically given a specific program name (DATPi). If the program number 0 is selected, then the DATTCH command is disabled. Before creating a new data program, be sure to delete the existing data program that has the same name. For example, if you wish to create data program #5 with the DATSIz5,1,144 command and DATP5 already exists, first delete DATP5 with the DEL DATP5 command and then issue the DATSIz5,1,144 command.

The second integer represents the total number of data elements (up to 6,500) you want in the data program. Upon issuing the DATSIz command, the data program is created with all the data elements initialized with a value of zero. (The DATSIz command is equivalent to creating a DATP program and filling it with DATA=0.0,0.0,0.0,0.0 commands up to the size indicated in the second integer.)

Each data statement, which contains four data elements, uses 43 bytes of memory. This amount of memory is subtracted from the memory allocated for user programs (see MEMORY command). Use the TDIR command to determine the amount of remaining memory for user program storage.

The data program has a tabular structure, where the data elements are stored 4 to a line. Each line of data elements is called a *data statement*. Each element is numbered in sequential order from left to right (1 - 4) and top to bottom (1 - 4, 5 - 8, 9 - 12, etc.). You can use the TPROG DATPi command ("i" represents the number of the data program) to display all the data elements of the data program. For example, if you issue the DATSIz1,13 command, data program #1 (called DATP1) is created with 13 data elements initialized to zero. The response to the TPROG DATP1 command is depicted below. Each line (*data statement*) begins with DATA=, and each data element is separated with a comma.

```
*DATA=0.0,0.0,0.0,0.0
*DATA=0.0,0.0,0.0,0.0
*DATA=0.0,0.0,0.0,0.0
*DATA=0.0
```

The DATSIz, DATTCH, and DAT commands will typically be used as a teach mode in this manner:

1. Issue the DATSIz command to create (or recall) the data program.
2. Store variable data (e.g., position, acceleration, velocity, etc.) to numeric variables (VAR).
3. Use DATTCH commands to store the data from the numeric variables into the data program. You can use the data pointer (DATPTR) command to select any data element in the data program, and to determine the number by which the pointer increments after each value from the DATTCH command is stored.
NOTE: If the DATTCH command is issued without having issued the DATSIz command, an error will result.
4. Use the DAT commands to read the stored data from the data program into the variable parameters of your motion program. You can use the DATPTR command to select any data element in the data program, and to determine the number by which the pointer increments after each DAT command.

Any use of the DATTCH and DAT commands will reference the current active data program (DATP) specified by the first integer of the last DATSIz or DATPTR command. If you want to use the DATSIz command to recall a data program, **and not create one**, specify only the first integer and not the second integer. For example, DATSIz7 recalls data program #7 (DATP7) as the active data program.

Example (for 4 axes):

```

DEL DATP5           ; Delete existing data program #5 (DATP5)
DATSIZ5,200        ; Create data program #5 (DATP5) with 200 data elements
DEF TEACH          ; Begin definition of program called TEACH
COMEXC0           ; Disable continuous command execution mode
MA1111            ; Enable the absolute positioning mode for all axes
HOM1111           ; Home all axes (absolute position counter set to zero after homing)
DATPTR5,1,1       ; Set data pointer to data element #1 in DATP5, and increment the
                  ; pointer by one element after every DATTCH value or DAT command
REPEAT            ; Set up a loop for teaching the positions
JOY1111           ; Enable joystick mode on all axes so that you can start moving the
                  ; axes into position with the joystick. Command processing stops
                  ; here until you activate trigger input TRG-A1 (IN.2) to disable the
                  ; joystick mode and execute the rest of the commands in the
                  ; repeat/until loop (assign the motor positions to the variables and
                  ; then store the positions from the variables to the data program).
VAR1=1PM          ; Store the current position of axis #1 in variable #1
VAR2=2PM          ; Store the current position of axis #2 in variable #2
VAR3=3PM          ; Store the current position of axis #3 in variable #3
VAR4=4PM          ; Store the current position of axis #4 in variable #4
DATTCH1,2,3,4     ; Store variables #1 - #4 into consecutive data elements
WAIT(IN.2=b0)     ; Wait for the "joystick release" input (TRG-A1) to be de-activated
UNTIL(DPTR=1)     ; Repeat loop until the data pointer wraps around to data element #1
HOM1111           ; Home all axes (absolute position counter set to zero after homing)
DATPTR5,1,1       ; Set data pointer to data element #1, read one data element at a time
REPEAT            ; Set up a repeat/until loop to read all data elements
D(DAT5),(DAT5),(DAT5),(DAT5) ; Read position data from the data program to the
                  ; distance command
GO1111            ; Make the move to the positions that were taught
T.2               ; Wait 0.2 seconds
UNTIL(DPTR=1)     ; Repeat loop until the data pointer wraps around to data element #1
END               ; End definition of program called TEACH

```

DATTCH Data Teach

Type	Data Storage	Product	Rev
Syntax	<I>DATTCH<,i,i,i>	6K	5.0
Units	i = number of a numeric variable		
Range	i = 1 - maximum number of numeric variables		
Default	n/a		
Response	n/a		
See Also	[DAT], [DATP], DATPTR, DATSIZ, DATTCH, VAR		

The Data Teach (DATTCH) command stores the values from the specified numeric variables (VAR) into the **currently active data program** (i.e., the data program specified with the last DATSIZ or DATPTR command). The value that is in the specified variable at the time the DATTCH command is executed is the value that is stored in the data program.

If the DATTCH command is issued without having first issued the DATSIZ command, an error will result. If a zero is entered in the first integer of the DATSIZ command (e.g., DATSIZØ), the DATTCH command is disabled.

As indicated by the number of integers in the syntax, the maximum number of variables that can be stored in the data program per DATTCH command is 4. The variables are stored in the data program, starting at the current location of the data pointer. The data pointer's position can be moved to any data element in any data program by use of the DATPTR command. After each successive DATTCH value is stored, the data pointer will increment by the number specified in the third integer of the DATPTR command. Any data element in the data program can be edited by setting the data pointer to that element and then issuing the DATTCH command.

As an example, suppose data program #1 (DATP1) is configured to hold 15 data elements (DATSIZ1,15), the data pointer is configured to start at the first data element and increment 1 data element after every DATTCH value is stored (DATPTR1,1,1), and the values of numeric variables #1 through #4 are already assigned (VAR1=2, VAR2=4, VAR3=8, VAR4=64). If you then enter the DATTCH1,2,3,4 command, the values of VAR1 through VAR4 will be assigned respectively to the first four data elements in the data

program and the pointer will stop at data element #5. The response to the TPROG DATP1 command would be as follows (the text is highlighted to illustrate the location of the data pointer after the DATTC1, 2, 3, 4 command is executed).

```
*DATA=2.0,4.0,8.0,64.0
*DATA=0.0,0.0,0.0,0.0
*DATA=0.0,0.0,0.0,0.0
*DATA=0.0,0.0,0.0
```

Example: Refer to the DATSIZ command.

DCLEAR Clear Display

Type	Display (RP240) Interface	Product	Rev
Syntax	<!>DCLEARi	6K	5.0
Units	n/a		
Range	i = 0 (clear all lines), 1 (clear line 1), or 2 (clear line 2)		
Default	n/a		
Response	n/a		
See Also	DLLED, DPASS, DPCUR, DSTP, DVAR, DVARB, DVARI, DWRITE		

The Clear Display (DCLEAR) command clears lines (as specified with *i*) of the RP240 display. After clearing a line, the cursor will be reset to the beginning of that line (or to the beginning of line 1 if all lines are cleared).

DEF Begin Program/Subroutine/Path Definition

Type	Program or Subroutine Definition	Product	Rev
Syntax	<!>DEF<t>	6K	5.0
Units	t = alpha text string (name of a program)		
Range	text string of 6 characters or less		
Default	n/a		
Response	n/a		
See Also	\$, DEL, END, ERASE, GOBUF, GOSUB, GOTO, MEMORY, PCOMP, PLCP, PRUN, RUN, [SS], TDIR, TMEM, TPROG, TSS, TSTAT		

The Define a Program/Subroutine (DEF) command is the beginning of a program, path contour, or subroutine definition. The syntax for the command is DEF followed by 6 or fewer alpha-numeric characters. The first character may not be a number. Refer to the MEMORY command description for information on program size restriction and total number of programs possible per product.

All programs are stored in a binary fashion within the 6K Series products. A program transferred back out (TPROG) after it has been defined (DEF), may not look identical to the program defined. However, the program is functionally identical.

NOTE

When defining a program and the memory limitation is exceeded, an error message will be generated, and bit 11 of the system status register will be set (SS or TSS). The program will be stored up to the point where the memory limitation was exceeded.

There is no actual difference in the definition of, or execution of a program versus the definition, or execution of a subroutine. Both a program and a subroutine are defined as the set of commands between a DEF<t> and an END command. If an invalid program/subroutine name is entered, an error message will be generated. An invalid program/subroutine name is any name that is also a current command (An example of an invalid name would be DEF_{homx}, because it is impossible for the operating system to distinguish the _{homx} subroutine call from the _{HOMx111} go home command.). A subroutine/program definition cannot be assigned the name "CLR" and cannot contain any of the following characters:

!, -, #, \$, %, ^, &, *, (,), +, -, _, =, {, }, \, |, ", :, ;, ', <, >, /, ., ?, /.

The RUN command can be used to start executing a program/subroutine. The program name by itself can also be used to start executing a program/subroutine. A compiled profile (contour or compiled motion profile) must be compiled with the PCOMP command before it can be executed with the PRUN command; and a compiled PLC (PLCP) program must be compiled with PCOMP before it can be executed with the SCANP or PRUN command.

The GOTO and GOSUB commands can be used within a program/subroutine to go to another program/subroutine.

NOTE: Program, compiled profile, or subroutine names must be deleted (DEL) before they can be redefined.

Example:

```
DEL pick          ; Delete program named pick
DEF pick         ; Begin definition of program named pick
G01100          ; Initiate motion on axes 1 and 2, not on axes 3 and 4
END             ; End program definition
RUN pick        ; Execute program pick
```

DEL

Delete a Program/Subroutine/Path

Type	Program or Subroutine Definition	Product	Rev
Syntax	<!>DEL<t>	6K	5.0
Units	t = alpha text string (name of a program)		
Range	text string of 6 characters or less		
Default	n/a		
Response	n/a		
See Also	\$, DEF, END, ERASE, GOSUB, GOTO, RUN		

The Delete a Program/Subroutine (DEL) command removes a program, path contour, or subroutine definition. The syntax for the command is DEL followed by 6 or fewer alpha-numeric characters. To delete all programs refer to the ERASE command. The DEL command can be placed inside a program (e.g., to delete a DATP program).

To edit an existing program, you must first delete it. The DEL command will not delete a label (\$).

Example:

```
DEF pick          ; Begin definition of program named pick
G01100          ; Initiate motion on axes 1 and 2
END             ; End program definition
RUN pick        ; Execute program pick
DEL pick        ; Deletes program named pick
```

DJOG

Enable RP240 Jog Mode

Type	Display (RP240) Interface	Product	Rev
Syntax	<!>DJOG	6K	5.0
Units	b = 0 or 1		
Range	0 = disable, 1 = enable		
Default	0		
Response	DJOG: *DJOG0		
See Also	JOG, JOGA, JOGAA, JOGAD, JOGADA, JOGVH, JOGVL		

The DJOG command allows you to branch into the RP240 front panel jog mode from within your user-defined program, adjust the position of the axes, and then return to program execution.

The DJOG1 command enables the RP240 jog mode on all axes. Once the RP240 jog mode is enabled, you can use the RP240 arrow keys to jog individual axes. Unlike the JOG command, command processing is suspended after the DJOG1 command is issued. Jogging acceleration and deceleration are performed with the parameters set with the Jog Acceleration (JOGA) and Jog Deceleration (JOGAD) commands. Jogging velocities are set with the Jog Velocity High (JOGVH) and the Jog Velocity Low (JOGVL) commands. Once in the RP240 Jog Mode, you can switch between low and high jog velocities for any axis, and you can also modify the two jog velocities using the RP240's EDIT key.

To disable the RP240 jog mode, press the **MENU RECALL** key or issue the immediate !DJOGØ command. Upon exiting the RP240 jog mode, the RP240's display is cleared.

To have the jog mode continually enabled during program execution, you must use jog inputs and the JOG command.

[DKEY] Value of RP240 Key

Type	Display (RP240) Interface; Assignment or Comparison	Product	Rev
Syntax	See below	6K	5.0
Units	n/a		
Range	n/a		
Default	n/a		
Response	n/a		
See Also	DCLEAR, DPCUR, [DREAD], DREADF, DREADI, DVAR, DWRITE		

The DKEY operator allows you to read the current state of the RP240 key-pad and use it in comparison commands (e.g., IF, WHILE, etc.) or variable assignments. **NOTE:** If two or more keys are pressed simultaneously, DKEY will report -1.

Syntax: VARn=DKEY where “n” is the variable number,
or DKEY can be used in an expression such as IF (DKEY=-1)

The value reported by the DKEY command is defined by the following table:

<u>Value of DKEY</u>	<u>Key currently active</u>
-1	None or multiple keys
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	.
11	+/-
12	C/E
13	ENTER
14	Menu Recall
15	STOP
16	PAUSE
17	CONTINUE
21	F1
22	F2
23	F3
24	F4
25	F5
26	F6

DLED Turn RP240 Display LEDs On/Off

Type	Display (RP240) Interface	Product	Rev
Syntax	<!>DLED	6K	5.0
Units	n/a		
Range	b = 0 (off) or 1 (on)		
Default	n/a		
Response	DLED: *DLED1101_0001		
See Also	DCLEAR, DPASS, DPCUR, DSTP, DVAR, DVARB, DVARI, DWRITE		

The DLED command controls the state of the 8 programmable LEDs on the RP240. *It is legal to substitute a binary variable (VARB) for the DLED command.*

Example:

```
DLED11XXXX01 ; Turn on LEDs 1, 2, and 8; turn off LED 7; leave LEDs 3,4,5,
                ; and 6 unchanged
VARB1=b10101010 ; Set bits 1, 3, 5 & 7 low, and bits 2, 4, 6, & 8 high
DLED(VARB1) ; Turn on LEDs 1, 3, 5 & 7; turn off LEDs 2, 4, 6, & 8
```

DPASS Change RP240 Password

Type	Display (RP240) Interface	Product	Rev
Syntax	<!>DPASS<i>	6K	5.0
Units	i = integer of up to 9 characters		
Range	1 - 9999		
Default	For the 6K: DPASS6850		
Response	DPASS: *DPASS6850		
See Also	DCLEAR, DLED, DPCUR, DSTP, DVAR, DVARB, DVARI, DWRITE		

The DPASS command changes the RP240 password. If the default password is not changed by the user, then there will be no password protection.

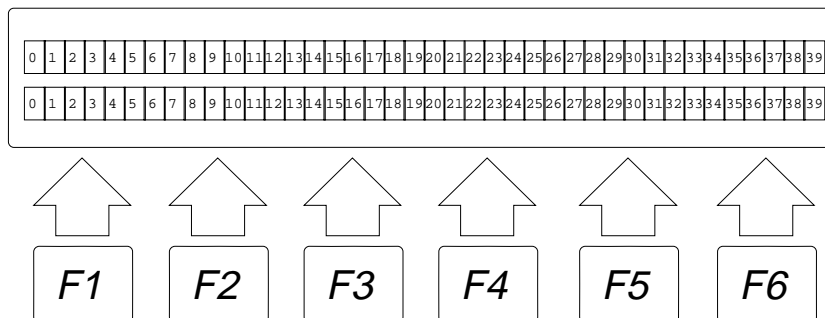
Example:

```
DPASS2001 ; New password = 2001
```

DPCUR Position Cursor

Type	Display (RP240) Interface	Product	Rev
Syntax	<!>DPCURi,i	6K	5.0
Units	1st i = line number, 2nd i = column		
Range	line number = 1 or 2, column = 0 - 39		
Default	n/a		
Response	n/a		
See Also	DCLEAR, DLED, DPASS, DREADI, DSTP, DVAR, DVARI, DVARB, DWRITE		

The Position Cursor (DPCUR) command changes the location of the cursor on the RP240 display. The RP240 lines are numbered from top to bottom, 1 to 2. The columns are numbered left to right, 0 to 39.



Example:

```
DPCUR2,15 ; Position cursor on line 2, column 15
```

[DPTR] Data Pointer Location

Type	Data Storage; Assignment or Comparison	Product	Rev
Syntax	see below	6K	5.0
Units	n/a		
Range	n/a		
Default	n/a		
Response	n/a		
See Also	[DAT], DATA, [DATP], DATPTR, DATSIZ, TDPTR		

The DPTR command can be used to compare the current pointer location (the number of the data element to which the data pointer is pointing) against another value or numeric variable, or to assign the pointer location number to a variable. The current data pointer location is referenced to the current active data program specified in the first integer of the last DATSIZ or DATPTR command.

Syntax: VARn=DPTR where “n” is the variable number,
or DPTR can be used in an expression such as IF (DPTR=1)

Example:

```
DATSIZ4,200 ; Create data program called DATP4 with 200 data elements
DATPTR4,20,2 ; Set the data pointer to data element #20 in DATP4 and set the
              ; increment to 2 (DATP4 becomes the current active data program)
VAR1=DPTR    ; Assign the number of the pointer location in DATP4 to numeric
              ; variable #1
VAR1         ; Response is *VAR1=20. Indicates that the data pointer is
              ; pointing to data element #20.
```

[DREAD] Read RP240 Data

Type	Display (RP240) Interface	Product	Rev
Syntax	See below	6K	5.0
Units	n/a		
Range	n/a		
Default	n/a		
Response	n/a		
See Also	[DREADF], DREADI, DVAR, DWRITE, [SS], TSS, VAR		

The Read RP240 Data (DREAD) command allows you to store numeric data entered in from the RP240's keypad into a variable. As the user presses RP240 numeric keys, the data will be displayed on the RP240 starting at the location equal to the current cursor location + 1 (for a sign bit):

```
VAR1=DREAD    Wait for RP240 numeric entry (terminated with the ENTER key), then set
              VAR1 equal to that value.
```

Additionally the DREAD command can be used as a variable assignment within another command that is expecting numeric data (Rule of Thumb: If the command syntax shows that the command field requires a real number (denoted by <r>) or and integer value (denoted by <i>), you can use the DREAD substitution.):

```
A(DREAD),5.0  Wait for RP240 numeric entry (terminated with the ENTER key), then set axis
              #1 acceleration to that value and set axis #2 acceleration to 5.0.
```

The DREAD command cannot be used in an expression such as VAR5=4+DREAD or IF (DREAD=1).

[DREADF] Read RP240 Function Key

Type	Display (RP240) Interface	Product	Rev
Syntax	See below	6K	5.0
Units	n/a		
Range	n/a		
Default	n/a		
Response	n/a		
See Also	[DREAD], DREADI, DVAR, DWRITE, [SS], TSS, VAR, VARI		

The Read RP240 Function Key (DREADF) command allows you to store numeric data entered in from a RP240 function key into a variable. Function key 1 (**F1**) = 1, **F2** = 2, etc., and **MENU RECALL (F0)** = 0.

Rule of Thumb for command value substitutions: If the command syntax shows that the command field requires a real number (denoted by <r>) or and integer value (denoted by <i>), you can use the DREADF substitution (e.g., V2, (DREADF)).

Example:

```
VAR1=DREADF      ; Wait for RP240 function key entry, then set VAR1 equal to that
                  ; value
IF(VAR1=5)       ; If function key 5 was hit then ...
GOx1            ; Start motion on axis #2
NIF             ; End if statement
```

DREADI RP240 Data Read Immediate Mode

Type	Display (RP240) Interface	Product	Rev
Syntax	<!>DREADI	6K	5.0
Units	n/a		
Range	1 (enable) or 0 (disable)		
Default	0		
Response	DREADI: *DREADI0		
See Also	DPCUR, [DREAD], [DREADF], VAR, VARI		

The DREADI1 command allows continual numeric or function key data entry from the RP240 (when used in conjunction with the DREAD and/or DREADF commands). In this immediate mode, program execution is not paused (waiting for data entry) when a DREAD or DREADF command is encountered.

NOTES

- While in the Data Read Immediate Mode (DREADI1), data is read into VAR and VARI variables only (e.g., A(DREAD) or V(DREAD) substitutions are not valid).
 - This feature is not designed to be used in conjunction with the RP240's standard menus (see *Programmer's Guide* for menu structure); the **RUN** and **JOG** menus will disable the DREADI mode.
 - After the RP240's ENTER key is pressed (to enter numeric data), the value is displayed on the RP240 display at the 1,30 location (showing 10 significant digits).
 - Do not assign the same variable to read numeric data **and** function key data—pick only one.
-

Simple Numeric Data Entry (example):

```
VAR1=25000      ; Initialize variable #1
DCLEAR0        ; Clear entire RP240 display
DWRITE"ENTER VALUE > " ; Send message to RP240 display starting at location 1,0
DREADI1        ; Enable RP240 data read immediate mode
VAR1=DREAD     ; Set variable #1 (VAR1) to receive data entered on the RP240.
                ; Current VAR1 data will be displayed at cursor location 1,30
                ; (fixed). New data will be displayed at current cursor location
                ; as defined by the previous DCLEAR, DWRITE and DPCUR commands—
                ; this is the home cursor location for subsequent data entries.
L77            ; Start loop of 77 repetitions
D(VAR1)        ; Set distance equal to the current (last entered) RP240 data
GO1            ; Initiate move on axis one
LN            ; End loop
DREADI0        ; Exit RP240 data read immediate mode
```

```

; As the loop is running, the user may enter in a new distance value
; (which must be terminated with the ENTER key) via the RP240 numeric keypad.
; The numeric keystrokes cause the digits to be displayed on the RP240
; starting at the home cursor location (see VAR1=DREAD description in the
; example above). When the ENTER key is pressed, the variable is updated;
; the most significant 10 digits (total, including sign & decimal point
; if appropriate) of this variable are displayed at cursor location 1,30;
; and then the data entry field (starting at home) is cleared.
; The 6K controller is ready to accept new data.

```

Numeric Data & Function Key Entry (example):

```

VAR1=25000      ; Initialize variable #1
VAR2=1          ; Initialize variable #2
DCLEAR0        ; Clear the RP240 display
DPCUR2,0       ; Place RP240 cursor on line 2, column 0 (bottom left corner of
                ; display)
DWRITE" SLOW FAST" ; Send message to RP240 display starting at location 2,0
DPCUR1,0       ; Place RP240 cursor on line 1, column 0 (top left corner)
DWRITE"ENTER VALUE > " ; Send message to RP240 display starting at location 1,0
DREADI1       ; Enable RP240 data read immediate mode
VAR1=DREAD    ; Set variable #1 (VAR1) to receive numeric data entered on the
                ; RP240's keypad
VAR2=DREADF   ; Set VAR2 to receive RP240 function key input
L             ; Begin loop
IF(VAR2=1)    ; If function key 1 was last pressed, do the IF statement
                ; (slow velocity)
V3.6         ; Set velocity to 3.6 units per second
NIF          ; End IF statement
IF(VAR2=2)    ; If function key 2 was last pressed, do the IF statement
                ; (fast velocity)
V6.4         ; Set velocity to 6.4 units per second
NIF          ; End IF statement
D(VAR1)      ; Set distance equal to the current (last entered) RP240 numeric
                ; data
GO1          ; Initiate the move on axis one
LN           ; End loop
; As the loop is running, the user may enter in a new distance value and/or
; choose between two different preset velocities. The display does not change
; when a function key is pressed.

```

Multiple Numeric Data Entry (example):

```

VAR2=0         ; Initialize variable #2 (VAR2)
VAR3=99        ; Initialize variable #3 (VAR3)
VAR4=10        ; Initialize variable #4 (VAR4)
VAR5=25000    ; Initialize variable #5 (VAR5)
DCLEAR0        ; Clear the entire RP240 display
DPCUR2,0       ; Place RP240 cursor on line 2, column 0 (bottom left corner)
DWRITE" ACCEL VEL DIST" ; Send message to RP240 display starting at location 2,0
DREADI1       ; Enable RP240 data read immediate mode
VAR2=DREADF   ; VAR2 will capture function key entries (0 - 6)
L             ; Begin loop
IF(VAR2<>0)   ; If a new function key is pressed, do the following code:
DCLEAR1       ; Clear line one of the RP240 display (top line)
IF(VAR2=1)    ; If function key 1 is pressed, do the IF statement
                ; (input acceleration)
DWRITE"ENTER ACCEL VALUE> " ; Send message to RP240 display starting at location 1,0
VAR3=DREAD    ; Set VAR3 equal to the numeric data entered on the RP240's keypad
NIF          ; End IF statement
IF(VAR2=2)    ; If function key 2 is pressed, do the IF statement (input velocity)
DWRITE"ENTER VEL VALUE> " ; Send message to RP240 display starting at location 1,0
VAR4=DREAD    ; Set VAR4 equal to the numeric data entered on the RP240's keypad
NIF          ; End IF statement
IF(VAR2=3)    ; If function key 3 is pressed, do the IF statement (input distance)
DWRITE"ENTER DIST VALUE> " ; Send message to RP240 display starting at location 1,0
VAR5=DREAD    ; Set VAR5 equal to the numeric data entered on the RP240's keypad
NIF          ; End IF statement
VAR2=0        ; Prohibit repeated execution of this code
VAR2=DREADF   ; Re-enable VAR2 to capture new function key entry
NIF          ; End IF statement
A(VAR3)      ; Set acceleration equal to the numeric value of VAR3

```

```

V(VAR4)          ; Set velocity equal to the numeric value of VAR4
D(VAR5)          ; Set distance equal to the numeric value of VAR5
GO1              ; Initiate the move on axis one
LN              ; End loop
; As the loop is running, the user may select among the three variables he wants
; to enter data into. These three variables correspond with acceleration,
; velocity, and distance. Each time the function key variable changes from 0
; (to 1, 2 or 3), then a new message is displayed and the VARi=DREAD command
; will put the current value of that variable in location 1,30 (upper right hand
; corner of the display). For example, the user can choose VEL (F2) and then
; repeatedly change VAR4 by entering a value on the RP240 numeric keypad and
; pressing the ENTER key. Each time through the loop, the VAR4 data is loaded
; into the V command.

```

DRES Drive Resolution

Type	Drive Configuration	Product	Rev
Syntax	<!><@><a>DRES<i>,<i>,<i>,<i>,<i>,<i>,<i>,<i>	6K	5.0
Units	i = steps/rev		
Range	200-1024000		(applicable only to
Default	4000		stepper axes)
Response	DRES: *DRES4000,4000,4000,4000 ... 1DRES: *1DRES4000		
See Also	DRFEN, DRFLVL, DRIVE, ERES, PULSE, SCALE, TSTAT		

The Drive Resolution (DRES) command is used to match the controller resolution to that of the motor/drive to which it is attached. This command is necessary in order to accurately calculate motor drive accelerations and velocities whether scaling is disabled (SCALE0), or enabled (SCALE1).

Example:

```

DRES200,10000,25000,25000 ; Set drive res. for axis 1 to 200 steps/rev, axis 2
                          ; to 10000 steps/rev, and axes 3 & 4 to 25000 steps/rev

```

DRFEN Enable/Disable Checking the Drive Fault Input

Type	Drive Configuration	Product	Rev
Syntax	<!><@><a>DRFEN	6K	5.0
Units	b = enable bit		
Range	b = 1 (check the state of the drive fault input), 0 (don't check the state of the drive fault input), or x (don't change)		
Default	0 (disabled)		
Response	DRFEN: *DRFEN0000_0000 1DRFEN: *1DRFEN0		
See Also	DRFLVL, DRIVE, [AS], [ASX], [ER], ERROR, TAS, TASX, TER		

Use the DRFEN command to enable or disable checking the state of the drive fault input for each axis. The default condition is that the drive fault input is not checked (DRFEN0); therefore, a drive fault would not be detectable. Even with DRFEN enabled (DRFEN1), the controller will not respond to a drive fault condition until the respective axis is enabled with the DRIVE1 command.

DRFEN1 is required before you can use these functions (remember that the default power-up state is DRFEN0):

- AS, TAS, and TASF (axis status) bit #14 reports if a drive fault occurred.
- ERROR bit #4 enables checking for the occurrence of a drive fault, and when it does, to branch to the ERRORP program.
- ER, TER, and TERF (error status) bit #4 reports if a drive fault occurred (if ERROR bit #4 is enabled).
- An output assigned the “Fault Indicator” function (OUTFNCi-F) will turn on when a drive fault occurs or a user fault input (INFNCi-F or LIMFNCi-F) is activated.

Regardless of the state of the DRFEN command, ASX, TASX, and TASXF (extended axis status) bit #4 will accurately report the hardware state of the drive fault input.

The DRFEN command setting is not saved in the controller’s battery backed RAM.

DRFLVL Drive Fault Level

Type	Drive Configuration	Product	Rev
Syntax	<!><@><a>DRFLVL	6K	5.0
Units	n/a		
Range	b = 0 (active low), 1 (active high), or X (don't change)		
Default	1		
Response	DRFLVL *DRFLVL1111_1111 1DRFLVL *1DRFLVL1		
See Also	[AS], [ASX], DRIVE, DRES, DRFEN, [ER], TAS, TASX, TER		

The Drive Fault Level (DRFLVL) command is used to individually set the fault input level for each axis. To enable the drive fault inputs for each axis, use the DRFEN command (default power-up state is disabled). Use the following table for setting the drive fault level for Compumotor drives.

Compumotor Product	Drive Fault Level
GEMINI, APEX, Dynaserv, LN, OEM Series, S, TQ, ZETA	Active High (DRFLVL1)
SV, BLH, L, LE, PDS, PK130	Active Low (DRFLVLØ)

The drive fault input schematic is shown in your product's *Installation Guide*.

Drive Fault Input Status:

Use bit #14 in the TAS, TASF, or AS commands to check the status of the drive fault input (if the drive is enabled and the drive fault input is enabled). Bit #4 of the TASX, TASXF, and ASX commands reports the status *even if the drive and the drive fault input are disabled*.

Drive Fault Level (DRFLVL)	Status of device driving the Fault input	AS bit #14 and ASX bit #4
DRFLVL1 (active high)	OFF or not connected (not sinking current) ON (sinking current)	1 (drive fault has occurred) Ø
DRFLVLØ (active low)	OFF or not connected (not sinking current) ON (sinking current)	Ø 1 (drive fault has occurred)

When a drive fault occurs, motion will be stopped on all axes (stopped at the LHAD & LHADA deceleration values) and program execution will be terminated.

Example:

```
DRFLVL0101 ; Set drive fault level to be active low on axes 1 & 3,
; active high on axes 2 & 4
```

DRIVE Drive Enable

Type	Drive Configuration	Product	Rev
Syntax	<!><@><a>DRIVE	6K	5.0
Units	n/a		
Range	b = 0 (shutdown), 1 (enable), or X (don't change)		
Default	0 (shutdown)		
Response	DRIVE *DRIVE1111_1111 1DRIVE *1DRIVE1		
See Also	[AS], [ASS], AXSDEF, DRFEN, DRFLVL, DRES, KDRIVE, TAS, TASX, TER		

The Drive Enable command energizes (DRIVE1) or de-energizes (DRIVEØ) a Compumotor motor/drive combination. The internal shutdown output circuit is illustrated in the product's *Installation Guide*.

NOTE: If the Disable Drive on Kill (KDRIVE) mode is enabled, the drive will be disabled in the event of a kill command or kill input.

- Steppers:** DRIVE1 energizes the motor drive (Shutdown+ sinks current and Shutdown- sources current).
DRIVEØ de-energizes the motor drive (Shutdown+ sources current and Shutdown- sinks current).
- Servos:** DRIVE1 energizes the motor drive (the SHTNO relay output is connected to COM, and the SHTNC relay output is disconnected from COM). DRIVEØ de-energizes the motor drive (the SHTNO relay output is disconnected from COM, and the SHTNC relay output is connected to COM). DRIVE1 also sets the commanded position (TPC) equal to the actual position (TPE).

NOTE: The DRIVEØ command will not de-energize a motor drive during motion.

Example:

DRIVE1110 ; Energize drives 1 through 3, de-energize drive 4

DRPCHK RP240 Check

Type	Communication Interface; Display (RP240) Interface	Product	Rev
Syntax	<!>DRPCHK<i>	6K	5.0
Units	n/a		
Range	0-3		
Default	0 for port COM1, 3 for port COM2 (PORT command setting determines which COM port's DRPCHK setting is checked)		
Response	DRPCHK *DRPCHK3		
See Also	LOCK, PORT, XONXOFF		

The Remote COM Port Check (DRPCHK) command is used to indicate whether a port is to be used with an RP240 or with 6K language commands. The DRPCHK command affects the COM port selected with the last PORT command. The DRPCHK command value is automatically saved in battery-backed RAM.

NOTE: COM1 is the connector labeled “RS-232” and COM2 is the connector labeled “RS-232/485.”

- DRPCHKØ..... The serial port will be used for 6K language commands. This is the default setting for COM1, and if using RS-485 half duplex on COM2. Power-up messages appear on all ports set to DRPCHKØ. If you ordered the FieldBus version of the 6K product, COM2 is factory-set to DRPCHKØ.
- DRPCHK1..... A check for the presence of an RP240 will be performed at power-up/reset. If an RP240 is present, the 6K product will initialize the RP240. If an RP240 is not present, the port may be used for 6K language commands. Note that RP240 commands will be sent at power-up and reset.
- DRPCHK2..... A status check for the presence of an RP240 will be periodically performed (every 5-6 seconds). If an RP240 is plugged in, the 6K product will initialize the RP240. Press F6 on the RP240 periodically until the 6K product recognizes the RP240. (The RP240 indicates that it has been recognized by beeping when F6 is pressed.)
- DRPCHK3..... A status check for the presence of an RP240 will be performed at power-up/reset. If an RP240 is present, the 6K product will initialize the RP240. If an RP240 is not present, no commands except DWRITE will have any effect for that port and the COM port will ignore received characters. This is the default setting for COM2, unless you are using RS-485 multi-drop communication (in which case the default changes to DRPCHKØ).

Each port has its own DRPCHK value, but only one may be set to DRPCHK2 or DRPCHK3 at any time.

RS-485 Communication: If you are using RS-485 communication in a multi-drop (requires you to change an internal DIP switch to select half duplex), the default setting for COM2 is DRPCHKØ. If the internal DIP switch setting is left at full duplex, the default setting for COM2 is DRPCHK3.

FieldBus Option: If you ordered the FieldBus version of the 6K product, COM2 is factory-set to DRPCHKØ.

Default values are used until DRPCHK is set for the first time. DRPCHK values are automatically saved in non-volatile memory. They do not change until you set new values. It may be advisable to include the DRPCHK command in your start-up program to ensure that it powers up in the correct setting for your current application.

DSTP Enable/Disable RP240 Stop Key

Type	Display (RP240) Interface	Product	Rev
Syntax	<!>DSTP	6K	5.0
Units	b = enable bit		
Range	b = 1 (enable) or 0 (disable)		
Default	1 (enable)		
Response	DSTP *DSTP1		
See Also	DLED, [DREAD], [DREADF]		

Use the DSTP command to enable or disable the stop key on the RP240 panel.

DVAR Display Variable on RP240

Type	Display (RP240) Interface	Product	Rev
Syntax	<!>DVARi,<i>,<i>,<i>	6K	5.0
Units	See below		
Range	n/a		
Default	See below		
Response	n/a		
See Also	[DREAD], [DREADF], DVARB, DVARI, DWRITE, VAR		

The Display Variable on RP240 (DVAR) command is used to display a numeric variable on the RP240's LCD at the current cursor location:

- 1st i = Variable number [Range 1-225]
- 2nd i = Number of whole digits displayed (left of decimal point) [Range 0-9]
- 3rd i = Number of fractional digits displayed (right of decimal point) [Range 0-8]
- 4th i = Sign bit: 0 = no sign displayed, 1 = display + or -

Example:

```
VAR2=542.14        ; Assign the value 542.14 to variable #2
DVAR2,6,3,1        ; Display variable #2 as +000542.140
DVAR2,3,1,0        ; Display variable #2 as 542.1
DVAR2,3,,1        ; Display variable #2 as +542
```

DVARB Display Binary Variable on RP240

Type	Display (RP240) Interface	Product	Rev
Syntax	<!>DVARBi,<i>	6K	5.0
Units	See below		
Range	n/a		
Default	See below		
Response	n/a		
See Also	DVAR, DVARI, DWRITE, VARB		

The Display Binary Variable on RP240 (DVARB) command is used to display a binary variable on the RP240's LCD at the current cursor location:

- 1st i = Variable number [Range 1-125]
- 2nd i = Number of bits displayed [Range 1-32]

Example

```
VARB2=b11001X11 ; Assign the value 11001X11 to binary variable #2
DVARB2,6         ; Display binary variable #2 as 1100_1X
DVARB2,3         ; Display binary variable #2 as 110
DVARB2,1         ; Display binary variable #2 as 1
```

DVARI Display Integer Variable on RP240

Type	Display (RP240) Interface	Product	Rev
Syntax	<!>DVARIi,<i>,<i>	6K	5.0
Units	See below		
Range	n/a		
Default	See below		
Response	n/a		
See Also	DVAR, DVARB, DWRITE, VARI		

The Display Integer Variable on RP240 (DVARI) command is used to display an integer variable on the RP240's LCD at the current cursor location:

- 1st i = Variable number [Range 1-225]
- 2nd i = Number of whole digits displayed [Range 0-9]
- 3rd i = Sign bit: 0=no sign displayed, 1=display + or - sign

Example

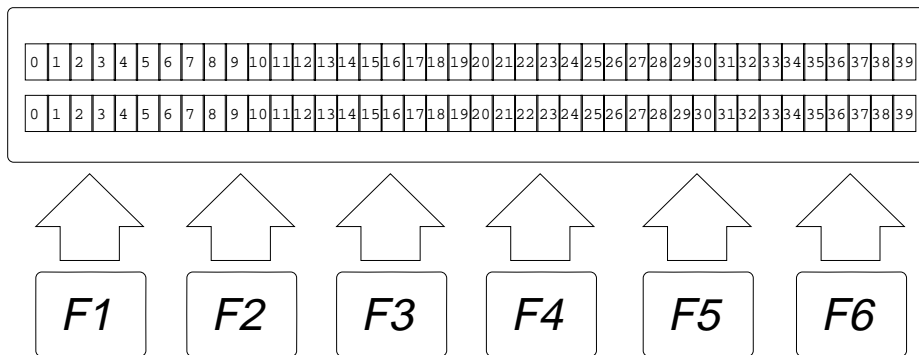
```
VARI2=542        ; Assign the value 542 to integer variable #2
DVARI2,6,1       ; Display integer variable #2 as +000542
DVARI2,3,1       ; Display integer variable #2 as +=542
```

DWRITE Write Text on RP240

Type	Display (RP240) Interface	Product	Rev
Syntax	<!>DWRITE "message"	6K	5.0
Units	n/a		
Range	Message can be ≤ 80 characters (may not use characters ", \, * or :)		
Default	See below		
Response	n/a		
See Also	DCLEAR, DLED, DPASS, DPCUR, DVAR, DVARB, DVARI, PORT		

The Write Text on RP240 (DWRITE) command displays a message on the RP240's LCD starting at the current cursor location. A message is a character string of up to 80 characters in length. The characters within the string may be any characters except quote ("), backslash (\), asterisk (*), and colon (:). Strings that have lower-case letters will be converted to upper case prior to display (see example).

The following graphic shows the location of the RP240's two-line, 40-character display. It also shows the characters in relation to the function keys.



HINT: If you do not have an RP240 and wish to send (only) characters out the serial port to another serial device, you may use this command. Place a backslash (\) before non-alphanumeric characters.

Example: DWRITE "HOMING SUCCESSFUL\13" = send message plus <CR>

Example:

```
DCLEAR0           ; Clear RP240 display
DPCUR1,12        ; Move cursor to line 1, column 12
DWRITE"Enter Number of Parts"; RP240 will display: ENTER NUMBER OF PARTS
VAR1=DREAD       ; RP240 waiting for data entry
```