DMC-3425 COMMAND REFERENCE

Manual Rev. 2.0a

By Galil Motion Control, Inc.

Galil Motion Control, Inc.
270 Technology Way
Rocklin, California 95765
Phone: (916) 626-0101
Fax: (916) 626-0102
Email Address: support@galilmc.com
URL: www.galilmc.com

Rev 6/06

ARRAYS	
DA	deallocate
_DA	arrays left
DM	define
_DM	space left
LA	list
QD	download
QU	print/upload
RA	record
RC	begin
_RC	recording?
RD	data
_RD	address
[ ]	index

COMMUNICATE	
CF	unsolicited
CW	unsolicited bit
DR	data record
EO	echo
HS	handle switch
IA	IP address
_IA	Ethernet info
IH	open handle
_IH	handle info
IN	user input
LZ	leading zeros
MG	message
PF	position format
QR	query record
QZ	record info
SA	send command
_SA	response
TH	tell handles
VF	variable format
WH	which handle
_WH	numeric
#TCPERR; RE	

CONTOUR	
CD	data
CM	axes
_CM	buffer full
DT	delta time
WC	wait for buffer

CONTROL	
DV	dual loop
FA	accel feedfwd
FV	speed feedfwd
IL	integrator limit
KD	derivative gain
KI	integral gain
KP	proportional gain
MO	motor off
_MO	motor off?
NB	notch width
NF	notch frequency
NZ	notch zero
OF	offset
PL	low pass
SH	servo here
TE	tell error
TL	torque limit
TM	sample time
TT	tell torque

ECAM	
EA	master
EB	enable
EC	counter
EG	engage slave
EM	modulus
EP	master
EQ	disengage
ET	table
EW	widen segment

EEPROM	
^R^S	master reset
BN	burn
BP	burn program
BV	burn variables
RS	reset

ERRORS	
AB	abort
_AB	abort input
BL	reverse soft limit
_ED	program line
_ED1	thread
ER	maximum TE
FL	forward soft limit
_LF	forward limit
_LR	reverse limit
OE	off on error
SC	stop code
TC	tell code
#CMDERR; EN1	
#LIMSWI; RE1	
#POSERR; RE1	

FEEDBACK	
AF	analog feedback
AL	arm latch
_AL	latch occurred?
CE	configure
OC	output compare
_OC	first pulse?
RL	read latch
_RL	latch position
TD	tell dual
TP	tell position
TV	tell velocity

GEAR	
GA	axes
GM	gantry mode
_GP	phase
GR	ratio

HOME	
DE	define dual
DP	define position
FE	find home only
FI	find index only
HM	home
_HM	home input

INFO	
_BN	serial number
_BV	axes
^R^V	firmware rev

I/O	
@AN[x]	analog in
@IN[x]	digital in
@OUT[x]	digital out
AI	wait for input
AO	set analog output
CB	clear digital out
CN	configure
CO	extended I/O
II	input interrupt
MB	Modbus TCP
MW	Modbus wait
OB	output bit
OP	output port
SB	set digital out
TI	tell input byte
TS	tell switches
TZ	tell Ethernet I/O
#ININT; RI1	

MATH	
@ABS[n]	|n|
@ACOS[n]	arccos
@ASIN[n]	arcsin
@ATAN[n]	arctan
@COM[n]	bit not
@COS[n]	cosine
@FRAC[n]	fraction
@INT[n]	integer
@RND[n]	round
@SIN[n]	sine
@SQR[n]	x^0.5
@TAN[n]	tangent
+	add
-	subtract
*	multiply
/	divide
( )	parenthesis
&	and
|	or
$	hexadecimal
<	less than
>	greater than
=	assign / equal
<=	less or equal
>=	greater or equal
<>	not equal

MOTION	
AC	acceleration
BG	begin
_BG	in motion?
DC	deceleration
IP	increment position
IT	s curve
JG	jog
PA	position absolute
_PA	last target
PR	position relative
_PR	relative target
RP	desired position
SP	speed
ST	stop

MOTION WAIT	
AD	distance (RP)
AM	complete (RP)
AP	position (TP)
AR	distance (RP)
AS	at speed (SP)
MC	complete (TP)
MF	forward (TP)
MR	reverse (TP)
TW	MC timeout
#MCTIME; EN1	

PROGRAM	
BK	breakpoint
DL	download
_DL	labels left
ED	edit
ELSE	if else
EN	end
ENDIF	if endif
HX	halt thread
IF	conditional
JP	for/while loop
JS	jump subroutine
LL	list labels
LS	list
LV	list variables
NO (')	comment
RE	return error
REM	fast comment
RI	return interrupt
SL	single step
TB	tell status byte
TR	debug trace
UL	upload
_UL	variables left
XQ	execute
_XQ	current line #
ZS	zero stack
ZS	stack level
#AUTO; EN	
#AUTOERR; EN	
;	command delimiter
#	subroutine

TIME	
AT	wait reference
TIME	clock
WT	wait

SINE DRIVE	
BA	axes
_BA	2nd DAC axis
BB	hall offset
BC	calibration
_BC	hall state
BD	degrees
BI	hall inputs
BM	magnetic cycle
BO	DAC offset
BS	setup
BZ	find zero
_BZ	distance to zero

STEPPER	
KS	smoothing
MT	motor type

VECTOR	
AV	wait for arc length
_AVS	arc length
CR	circle
CS	clear sequence
_CS	segment
ES	elliptical scale
LE	linear end
_LE	total arc length
LI	linear point
LM	linear axes
_LM	buffer space
VA	acceleration
VD	deceleration
VE	vector end
VM	vector axes
_VM	velocity
VP	vector point
_VP	last point
VR	VS multiplier
VS	speed
VT	s curve

DISTRIBUTED	
CH	connect handle
HC	handle configure
HR	handle restore
HW	handle wait
LO	lockout handle
LR	launch record
NA	number of axes
OQ	I OC-7007 output
QW	slave update rate
ZA	user variable A
ZB	user variable B

Contents

THIS PAGE LEFT BLANK INTENTIONALLY

Overview

Controller Notation

This command reference is a supplement to the Galil User Manual. For proper controller operation, consult the Users Manual. This command reference describes commands for Galil's E-Series Motion Controllers: DMC-3415 and DMC-3425. Commands are listed in alphabetical order.

Please note that all commands may not be valid for every controller. To identify the controllers for which the command is applicable, please review the Usage Section of the command description.

Axes Notation:

The E-Series Line of motion controllers can operate with up to 8 axes in a distributed control system. These axes are designated in order as A,B,C,D,E,F,G and H. For example, to determine the encoder position for the first axis, the command would be either TPA (Tell Position of A Axis).

Traditional notation refers to the first 4 axes as X,Y,Z and W. For simplicity, this reference only uses the axes designators, A,B,C,D,E,F,G and H. Please note that the controller will continue to accept X,Y,Z and W for the first 4 axes.

Command Descriptions

Each executable instruction is listed in the following section in alphabetical order. Below is a description of the information, which is provided for each command.

The two-letter Opcode for each instruction is placed in the upper right corner. Some commands have a binary equivalent and the binary value is listed next to the ASCII command in parenthesis. For binary command mode, see discussion below. Below the opcode is a description of the command and required arguments.

Global Vs. Local Commands

The DMC-3425 can be operated in either the global or the local mode. The local mode is used when a PC communicates directly with a DMC-3425 controller either through Ethernet or the serial port. The global mode is used to denote commands sent to the distributed system, which is a master controller with up to 8 axes of slave controllers.

The DMC-3425 controller in local mode will only use commands addressed to 2 axes of control, A and B.

The DMC-3425 controller in global mode, with up to 8 axes of control, will used commands addresses to axes A – H.

Command Syntax

Galil commands sometimes require arguments. These arguments identify specific axes to be affected or provide the numerical value to be set.

Axes Arguments

For the Galil commands require identification of an axis or axes, the following axes designators are used: A,B,C,D,E,F,G and H. The following syntax rules apply:

1. No commas are needed and the axes do not have to be specified in any order.

2. Do not insert any spaces prior to any command.

3. The command argument must be separated from the command by a single space.

4. When an argument is not required and is not given, the command is executed for all axes.

Examples

VALID SYNTAX	INTERPRETATION
SH BCAD	Servo Here, A, B, C and D axes
SH ADEG	Servo Here, A, D, E and G axes
SH H	Servo Here, H axis only
SH	Servo Here, all axes

Numerical Arguments

Some commands require numerical arguments. In the argument description, these commands are followed by lower case n,n,n,n,n,n,n,n, where the letter, n, represents the value. Values may be specified for any axis separately or any combination of axes. Commas separate the argument for each axis. Examples of valid syntax are listed below.

Examples

VALID SYNTAX	INTERPRETATION
AC n	Specify argument for a axis only
AC n,n	Specify argument for a and b only
AC n,,n	Specify argument for a and c only
AC n,n,n,n	Specify arguments for a,b,c,d axes
AC n,n,n,n	Specify arguments for a,b,c,d
AC ,n,,,n	Specify arguments for b and e axis only
AC ,,,n,n	Specify arguments for e and f

Where n is replaced by actual values.

Direct Command Arguments

An alternative method for specifying data is to set data for individual axes using an axis designator followed by an equals sign. The * symbol can be used in place of the axis designator. The * defines data for all axes to be the same. For

Examples

VALID SYNTAX	INTERPRETATION
PRA=1000	Sets A axis data at 1000
PR*=1000	Sets all axes to 1000

Interrogation

Most commands accept a question mark (?) as an argument. This argument causes the controller to return the information listed in the command description. Type the command followed by a ? for each axis requested. The syntax format is the same as the parameter arguments described above except '?' replaces the values.

Examples

VALID SYNTAX	INTERPRETATION
PR ?	The controller will return the PR value for the A axis
PR ,,,?	The controller will return the PR value for the D axis
PR ?,?,?,?	The controller will return the PR value for the A,B,C and D axes
PR ,,,,,,,?	The controller will return the PR value for the H axis

Operand Usage

Most commands have a corresponding operand that can be used for interrogation. The Operand Usage description provides proper syntax and the value returned by the operand. Operands must be used inside of valid DMC expressions. For example, to display the value of an operand, the user could use the command:

MG ‘operand’

All of the command operands begin with the underscore character (_). For example, the value of the current position on the A axis can be assigned to the variable ‘V’ with the command:

V=_TPA

Usage Description

The Usage description specifies the restrictions on proper command usage. The following provides an explanation of the command information provided:

"While Moving":

Describes whether the command is valid while the controller is performing a motion.

"In a program":

Describes whether the command may be used as part of a user-defined program.

"Command Line":

Describes whether the command may be used as a direct command.

“Controller Usage”:

Identifies the controller models that can accept the command.

Default Description

In the command description, the DEFAULT section provides the default values for controller setup parameters. These parameters can be changed and the new values can be saved in the controller's non-volatile memory by using the command, BN. If the setup parameters are not saved in non-volatile memory, the default values will automatically reset when the system is reset. A reset occurs when the power is turned off and on, when the reset button is pushed, or the command, RS, is given.

Resetting the Controller to Factory Default

When a master reset occurs, the controller will always reset all setup parameters to their default values and the non-volatile memory is cleared to the factory state. A master reset is executed by the command, <ctrl R> <ctrl S> <Return> OR by powering up or resetting the controller with the MRST jumper or dip switch on.

For example, the command KD is used to set the Derivative Constant for each axis. The default value for the derivative constant is 64. If this parameter is not set by using the command, KD, the controller will automatically set this value to 64 for each axis. If the Derivative Constant is changed but not saved in non-volatile memory, the default value of 64 will be used if the controller is reset or upon

power up of the controller. If this value is set and saved in non-volatile memory, it will be restored upon reset until a master reset is given to the controller.

The default format describes the format for numerical values that are returned when the command is interrogated. The format value represents the number of digits before and after the decimal point.

Servo Update Rates

The standard servo update period on all E-Series Motion Controllers is 1msec. To change the servo update, use the command, TM. The controller firmware will allow operation up to the following rates:

DMC-3415	250 usec
DMC-3425	375 usec

Rules for Operating with Non-Default Servo Update Rates

As mentioned, the default servo update rate is 1msec. When operating with different servo update rates, the following rules apply:

1. When operating multiple controllers in a distributed controls system, the master controller cannot operate with fast firmware.
2. The servo update rate of the master must be greater than the servo update rate of all slaves

# #

FUNCTION: Label (subroutine)

DESCRIPTION:

The # operator denotes the name of a program label (for example #Move). Labels can be up to seven characters long and are often used to implement subroutines or loops. Labels are divided into (a) user defined and (b) automatic subroutines. User defined labels can be printed with LL and the number of labels left available can be queried with MG _DL. The automatic subroutines include #CMDERR, #LIMSWI, #POSERR, #ININT, #AUTO, and #MCTIME.

ARGUMENTS: #nnnnnnn where

nnnnnnn is a label name up to seven characters

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	No
Controller Usage	**ALL**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

LL	List labels
_UL	Labels left
JP	Jump statement
JS	Jump subroutine

EXAMPLES:

```
#Loop; JP#Loop,          ;'wait until x becomes 10
x=10


#Move                    ;'define a subroutine to move the x axis
  PRX=1000
  BGX
  AMX
EN
```

$

FUNCTION: Hexadecimal

DESCRIPTION:

The $ operator denotes that the following string is in hexadecimal notation

ARGUMENTS: $nnnnnnnn.mmmm

n is up to eight hexadecimal digits (denoting 32 bits of integer)

m is up to four hexadecimal digits (denoting 16 bits of fraction)

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

*	Multiply (shift left)
/	Divide (shift right)
MG {$8.4}	Print in hexadecimal

EXAMPLES:

```
x = $7fffffff.0000                  ;'store 2147483647 in x
y = x & $0000ffff.0000              ;'store lower 16 bits of x in y
z = x & $ffff0000.0000 /
$10000                              ;'store upper 16 bits of x in z
```

& |

FUNCTION: Bitwise Logical Operators AND and OR

DESCRIPTION:

The operators & and | are typically used with IF, JP, and JS to perform conditional jumps; however, they can also be used to perform bitwise logical operations.

ARGUMENTS: n & m or n | m where

n and m are signed numbers in the range -2147483648 to 2147483647.

For IF, JP, and JS, n and m are typically the results of logical expressions such as (x > 2)

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

@COM[n]	Bitwise complement
IF	If statement
JP	Jump statement
JS	Jump subroutine

EXAMPLES:

```
IF (x > 2) & (y = 4)        ;x must be greater than 2 and y equal to 4 for the message to print
  MG “true”
ENDIF

:MG 1 | 2                   ;'Bitwise operation:  01 OR 10 is 11 = 3
 3.0000
  :
```

( )

FUNCTION: Parentheses (order of operations)

DESCRIPTION:

The parentheses denote the order of math and logical operations. Note that the controller DOES NOT OBEY STANDARD OPERATOR PRECEDENCE. For example, multiplication is NOT evaluated before addition. Instead, the controller follows left-to-right precedence. Therefore, it is recommended to use parenthesis as much as possible.

ARGUMENTS: (n) where

n is a math (+ - * /) or logical (& |) expression

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

+ - * /	Math Operators
& |	Logical Operators

EXAMPLES:

```
:MG 1 + 2 * 3
 9.0000
:MG 1 + (2 * 3)
 7.0000
  :
```

;

FUNCTION: Semicolon (Command Delimiter)

DESCRIPTION:

The semicolon operator allows multiple Galil commands to exist on a single line. It is used for the following three reasons:

(1) To put comments on the same line as the command (BGX ;'begin motion)

(2) To compress DMC programs to fit within the program line limit (Note: use a compression utility to do this. Do not program this way because it is hard to read.)

(3) To give higher priority to a thread. All commands on a line are executed before the thread scheduler switches to the next thread.

ARGUMENTS: n; n; n; … where

n is a Galil command

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

NO or '	comment

EXAMPLES:

```
BGX ;'comment

PRX=1000;BGX;AM          ;'Save program line space
X

#High                    ;'#High priority thread executes twice as fast as #Low when run in
  a = a + 1; b = b + 1   ;'parallel
JP#High

#Low
  c = c + 1
  d = d + 1
JP#Low
```

[ ]

FUNCTION: Square Brackets (Array Index Operator)

DESCRIPTION:

The square brackets are used to denote the array index for an array, or to denote an array name.

ARGUMENTS: mmmmmmmm[n] where

mmmmmmmm is the array name

n is the array index and is an integer between 0 and 7999

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

DM	Dimension Array
QU	Print/Upload Array

EXAMPLES:

```
DM A[100]      ;'define a 100 element array
A[0] = 3       ;'set first element to 3
MG A[0]        ;'print element 0
QU A[]         ;'print entire array
```

+ - * /

FUNCTION: Math Operators

DESCRIPTION:

The addition, subtraction, multiplication, and division operators are binary operators (they take two arguments and return one value) used to perform mathematical operations on variables, constants, and operands.

ARGUMENTS: (n + m) or (n – m) or (n * m) or (n / m) where

n and m are signed numbers in the range -2147483648 to 2147483647

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

( )	Parenthesis

EXAMPLES:

```
x = ((1 + (2 * 3)) / 7) - 2          ;’assign -1 to x
```

<, >, =, <=, >=, <>

FUNCTION: Comparison Operators

DESCRIPTION:

The comparison operators are as follows:

<	less than
>	greater than
=	equals
<=	less than or equal
>=	greater than or equal
<>	not equals

These are used in conjunction with IF, JP, JS, ( ), &, and | to perform conditional jumps. The result of a comparison expression can also be printed with MG or assigned to a variable.

ARGUMENTS: (n < m) or (n > m) or (n = m) or (n <= m) or (n >= m) or (n <> m) where

n and m are signed numbers in the range -2147483648 to 2147483647

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

( )	Parentheses
IF	If statement
JP	Jump
JS	Jump subroutine

EXAMPLES:

```
IF (x > 2) & (y = 4)      ;x must be greater than 2 and y equal to 4 for the message to print
  MG “true”
ENDIF
```

=

FUNCTION: Equals (Assignment Operator)

DESCRIPTION:

The assignment operator is used for three reasons:

(1) to define and initialize a variable (x = 0) before it is used

(2) to assign a new value to a variable (x = 5)

(3) to print a variable or array element (x= which is equivalent to MG x). MG is the preferred method of printing.

ARGUMENTS: mmmmmmmm = n where

mmmmmmmm is a variable name and n is a signed number in the range -2147483648 to 2147483647

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	-
Command Line	Yes		
Controller Usage	**ALL**		

RELATED COMMANDS:

MG	Print Message

EXAMPLES:

```
:x=5                ;'define and initialize x to 5
:x=                 ;'print x two different ways
 5.0000
:MG x
 5.0000
:
```

AB

FUNCTION: Abort

DESCRIPTION:

AB (Abort) stops a motion instantly without a controlled deceleration. If there is a program operating, AB also aborts the program unless a 1 argument is specified. The command, AB, will shut off the motors for any axis in which the off-on-error function is enabled (see command "OE").

AB aborts motion on all axes in motion and cannot stop individual axes.

ARGUMENTS: AB n where

n = 0 The controller aborts motion and program

n = 1 The controller aborts motion only

No argument will cause the controller to abort the motion and program

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	---
Default Format	---

OPERAND USAGE:

_AB gives state of Abort Input, 1 inactive and 0 active.

RELATED COMMANDS:

"SH"	Specifies deceleration rate.
"OE"	Specifies Off-On-Error

EXAMPLES:

AB	Stops motion
OE 1,1,1,1	Enable off-on-error
AB	Shuts off motor command and stops motion
#A	Label - Start of program
JG 20000	Specify jog speed on A-axis
BGA	Begin jog on A-axis
WT 5000	Wait 5000 msec
AB1	Stop motion without aborting program
WT 5000	Wait 5000 milliseconds
SH	Servo Here
JP #A	Jump to Label A
EN	End of the routine

Hint: *Remember to use the parameter 1 following AB if you only want the motion to be aborted. Otherwise, your application program will also be aborted.*

@ABS[n]

FUNCTION: Absolute value

DESCRIPTION:

Takes the absolute value of the given number. Returns the value if positive, and returns -1 times the value if negative.

ARGUMENTS: @ABS[n] where

n is a signed number in the range -2147483647 to 2147483647

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

@SQR	Square Root

EXAMPLES:

```
:MG @ABS[-2147483647]
 2147483647.0000
:
```

AC

FUNCTION: Acceleration

DESCRIPTION:

The Acceleration (AC) command sets the linear acceleration rate of the motors for independent moves, such as PR, PA and JG moves. The acceleration rate may be changed during motion. The DC command is used to specify the deceleration rate.

ARGUMENTS: AC n,n,n,n,n,n,n,n or ACA=n where

n is an unsigned numbers in the range 1024 to 67107840. The parameters input will be rounded down to the nearest factor of 1024. The units of the parameters are counts per second squared.

n = ? Returns the acceleration value for the specified axes.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	256000
Default Format	8.0

OPERAND USAGE:

_ACa contains the value of acceleration for the specified axis.

RELATED COMMANDS:

"DC"	Specifies deceleration rate.
"FA"	Feed forward Acceleration
"IT"	Smoothing constant - S-curve

EXAMPLES:

AC 150000,200000,300000,400000	Set A-axis acceleration to 150000, B-axis to 200000 counts/sec^2, the C-axis to 300000 counts/sec^2, and the D-axis to 400000 count/sec^2.
AC ?,?,?,?	Request the Acceleration
0149504,0199680,0299008,0399360	Return Acceleration (resolution, 1024)
V=_ACB	Assigns the B acceleration to the variable V

Hint: *Specify realistic acceleration rates based on your physical system such as motor torque rating, loads, and amplifier current rating. Specifying an excessive acceleration will cause large following error during acceleration and the motor will not follow the commanded profile. The acceleration feed forward command FA will help minimize the error.*

@ACOS[n]

FUNCTION: Inverse cosine

DESCRIPTION:

Returns in degrees the arc cosine of the given number.

ARGUMENTS: @ACOS[n] where

n is a signed number in the range -1 to 1.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

@ASIN	Arc sine
@SIN	sine
@ATAN	Arc tangent
@COS	Cosine
@TAN	Tangent

EXAMPLES:

```
:MG @ACOS[-1]
 180.0000
:MG @ACOS[0]
 90.0000
:MG @ACOS[1]
 0.0001
:
```

AD

FUNCTION: After Distance

DESCRIPTION:

The After Distance (AD) command is a trippoint used to control the timing of events. This command will hold up the execution of the following command until ***one*** of the following conditions have been met:

1. The commanded motor position crosses the specified relative distance from the start of the move.
2. The motion profiling on the axis is complete.
3. The commanded motion is in the direction that moves away from the specified position.

The units of the command are quadrature counts. Only one axis may be specified at a time. The motion profiler must be on or the trippoint will automatically be satisfied.

Note: AD command will be affected when the motion smoothing time constant, IT, is not 1. See IT command for further information.

ARGUMENTS: AD n,n,n,n,n,n,n,n or ADA=n where

n is an unsigned integers in the range 0 to 2147483647 decimal.

Note: The AD command cannot have more than 1 argument.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

"AV"	After distance for vector moves
"AP"	After position trip point
"AR"	After relative distance trip point
"MF"	Motion Forward trip point
"MR"	Motion Reverse trip point

EXAMPLES:

#A;DP0,0,0,0	Begin Program
PR 10000, 20000, 30000, 40000	Specify positions
BG	Begin motion
AD 5000	After A reaches 5000
MG "Halfway to A";TPA	Send message
AD ,10000	After B reaches 10000
MG "Halfway to B";TPB	Send message
AD ,,15000	After C reaches 15000
MG "Halfway to C";TPC	Send message
AD ,,,20000	After D reaches 20000
MG "Halfway to D";TPD	Send message
EN	End Program

Hint: *The AD command is accurate to the number of counts that occur in 2 msec. Multiply your speed by 2 msec to obtain the maximum position error in counts. Remember AD measures incremental distance from start of move on one axis.*

AF

FUNCTION: Analog Feedback

DESCRIPTION:

The Analog Feedback (AF) command is used to set an axis with analog feedback instead of digital feedback (quadrature/pulse dir). The analog feedback is decoded by a 12-bit A/D converter, therefore an input voltage of 10 volts is decoded as a position of 2047 counts and a voltage of -10 volts corresponds to a position of -2048 counts. An option is available for 16-bits where an input voltage of 10 volts is decoded as a position of 32,768 counts and a voltage of -10 volts corresponds to a position of -32,767 counts.

When using the analog feedback mode, analog input 1 is used for the first axis on the controller and analog input 2 is used for the second axis.

ARGUMENTS: AF n,n,n,n,n,n,n,n or AFA=n where

n = 1 Enables analog feedback

n = 0 Disables analog feedback and switches to digital feedback

n = ? Returns the state of analog feedback for the specified axes. 0 disabled, 1 enabled

USAGE:

While Moving	No
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	0,0,0,0
Default Format	-

OPERAND USAGE:

_AFa contains a “1” if analog feedback is enabled and “0” if not enabled for the specified axis.

RELATED COMMANDS:

"MT"	Motor Type
"CE"	Configure Encoder

EXAMPLES:

AF 1,0,0,1	Analog feedback on A and D axis
V1 = _AFA	Assign feedback type to variable
AF ?,?,?	Interrogate feedback type

AI

FUNCTION: After Input

DESCRIPTION:

The AI command is a trippoint used in motion programs to wait until after a specified input has changed state. This command can be configured such that the controller will wait until the input goes high or the input goes low.

ARGUMENTS: AI +/-n where

n is an integer between 1 and 7 and represents the input number. If n is positive, the controller will wait for the input to go high. If n is negative, it waits for n to go low.

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	-
Command Line	Yes		
Controller Usage	**ALL CONTROLLERS, LOCAL AXES ONLY**		

RELATED COMMANDS:

@IN[n]	Function to read input 1 through 8
"II"	Input interrupt
#ININT	Label for input interrupt

EXAMPLES:

#A	Begin Program
AI 8	Wait until input 8 is high
SP 10000	Speed is 10000 counts/sec
AC 20000	Acceleration is 20000 counts/sec2
PR 400	Specify position
BG A	Begin motion
EN	End Program

Hint: *The AI command actually halts execution until specified input is at desired logic level. Use the conditional Jump command (JP) or input interrupt (II) if you do not want the program sequence to halt.*

AL

FUNCTION: Arm Latch

DESCRIPTION:

The AL command enables the latching function (high speed main or auxiliary position capture) of the controller. When the position latch is armed, the main encoder position will be captured upon a low going signal. Each axis has a position latch and can be activated through the general inputs. For a single axis master or slave, Input 1 is the latch for the axis. For a 2-axis master or slave, Input 1 is the latch for the first axis and Input 2 is the latch for the second axis.

The command RL returns the captured position for the specified axes. When interrogated the AL command will return a 1 if the latch for that axis is armed or a zero after the latch has occurred. The CN command can be used to change the polarity of the latch function.

ARGUMENTS: AL nnnnnnnn or AL n,n,n,n,n,n,n,n where

n can be A,B,C,D,E,F,G or H. The value of n is used to specify main encoder for the specified axis to be latched

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	0
Default Format	1.0

OPERAND USAGE:

_ALa contains the state of the specified latch. 0 = not armed, 1 = armed.

RELATED COMMANDS:

"RL"	Report Latch

EXAMPLES:

#START	Start program
ALB	Arm B-axis latch
JG,50000	Set up jog at 50000 counts/sec
BGB	Begin the move
#LOOP	Loop until latch has occurred
JP #LOOP,_ALB=1	
RLY	Transmit the latched position
EN	End of program

AM

FUNCTION: After Move

DESCRIPTION:

The AM command is a trippoint used to control the timing of events. This command will hold up execution of the following commands until the current move on the specified axis or axes is completed. Any combination of axes or a motion sequence may be specified with the AM command. For example, AM AB waits for motion on both the A and B axis to be complete. AM with no parameter specifies that motion on all axes is complete.

ARGUMENTS: AM nnnnnnnnnn where

n is A,B,C,D,E,F,G,H,S or any combination to specify the axis or sequence

No argument specifies to wait for after motion on all axes and / or sequences

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	0
Default Format	1.0

RELATED COMMANDS:

"BG"	_BGa contains a 0 if motion complete
“AR”	After relative distance
"MC"	Motion complete

EXAMPLES:

#MOVE	Program MOVE
PR 5000,5000,5000,5000	Position relative moves
BG A	Start the A-axis
AM A	After the move is complete on A,
BG B	Start the B-axis
AM B	After the move is complete on B,
BG C	Start the C-axis
AM C	After the move is complete on C
BG D	Start the D-axis
AM D	After the move is complete on D
EN	End of Program

Hint: *AM is a very important command for controlling the timing between multiple move sequences. For example, if the A-axis is in the middle of a position relative move (PR) you cannot make a position absolute move (PAA, BGA) until the first move is complete. Use AMA to halt the program sequences until the first motion is complete. AM tests for profile completion. The actual motor may still be moving. To halt program sequence until the actual motion has completed, use the MC command. Another method for testing motion complete is to check for the internal variable _BG, being equal to zero.*

@AN[n]

FUNCTION: Read analog input

DESCRIPTION:

Returns the value of the given analog input in volts

ARGUMENTS: @AN[n] where

n is an unsigned integer in the range 1 to 8

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

@IN	Read digital input
@OUT	Read digital output
SB	Set digital output bit
CB	Clear digital output bit
OF	Set analog output offset

EXAMPLES:

```
:MG @AN[1] ;'print analog input 1
 1.7883
  :x = @AN[1] ;'assign analog input 1 to a variable
```

AO

FUNCTION: Analog Out

DESCRIPTION:

The AO command sets the analog output voltage of Modbus Devices connected via Ethernet.

ARGUMENTS: AO m, n where

m is the I/O number calculated using the following equations:

m = (SlaveAddress*10000) + (HandleNum*1000) + ((Module-1)*4) + (Bitnum-1)

Slave Address is used when the ModBus device has slave devices connected to it and specified as Addresses 0 to 255. Please note that the use of slave devices for modbus are very rare and this number will usually be 0.

HandleNum is the handle specifier from A to H.

Module is the position of the module in the rack from 1 to 16.

BitNum is the I/O point in the module from 1 to 4.

n = the voltage which ranges from 9.99 to –9.99

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	---
Default Format	---

RELATED COMMANDS:

"SB"	Set Bit
“CB"	Clear Bit

AP

FUNCTION: After Absolute Position

DESCRIPTION:

The After Position (AP) command is a trippoint used to control the timing of events. This command will hold up the execution of the following command until one of the following conditions have been met:

1. The actual motor position crosses the specified absolute position. When using a stepper motor, this condition is satisfied when the stepper position (as determined by the output buffer) has crossed the specified position. For further information see Chapter 6 of the User Manual *"Stepper Motor Operation"*.
2. The motion profiling on the axis is complete.
3. The commanded motion is in the direction which moves away from the specified position.

The units of the command are quadrature counts. Only one axis may be specified at a time. The motion profiler must be on or the trippoint will automatically be satisfied

ARGUMENTS: AP n,n,n,n,n,n,n,n or APA=n where

n is a signed integer in the range -2147483648 to 2147483647 decimal

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	---
Default Format	---

RELATED COMMANDS:

"AD"	Trippoint for relative distances
"AR"	After relative distance
"MF"	Trippoint for forward motion

EXAMPLES:

#TEST	Program B
DP0	Define zero
JG 1000	Jog mode (speed of 1000 counts/sec)
BG A	Begin move
AP 2000	After passing the position 2000
V1=_TPA	Assign V1 A position
MG "Position is", V1=	Print Message
ST	Stop
EN	End of Program

Hint: *The accuracy of the AP command is the number of counts that occur in 2 msec. Multiply the speed by 2 msec to obtain the maximum error. AP tests for absolute position. Use the AD command to measure incremental distances.*

AR

FUNCTION: After Relative Distance

DESCRIPTION:

The After Relative (AR) command is a trippoint used to control the timing of events. This command will hold up the execution of the following command until one of the following conditions have been met:

1. The actual motor position crosses the specified relative distance from either the start of the move or the last AR or AD command. When using a stepper motor, this condition is satisfied when the stepper position (as determined by the output buffer) has crossed the specified Relative Position. For further information see Chapter 6 of the User Manual *"Stepper Motor Operation"*.
2. The motion profiling on the axis is complete.
3. The commanded motion is in the direction which moves away from the specified position.

The units of the command are quadrature counts. Only one axis may be specified at a time. The motion profiler must be on or the trippoint will automatically be satisfied.

Note: AR will be affected when the motion smoothing time constant, IT, is not 1. See IT command for further information.

ARGUMENTS: AR n,n,n,n,n,n,n,n or ARA=n where

n is an unsigned integer in the range 0 to 2147483647 decimal.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

"AD"	Trippoint for after distance
"AV"	Trippoint for after vector position for coordinated moves
"AP"	Trippoint for after absolute position

EXAMPLES:

#A;DP 0,0,0,0	Begin Program
JG 50000,,,7000	Specify speeds
BG AD	Begin motion
#B	Label
AR 25000	After passing 25000 counts of relative distance on A-axis
MG "Passed _A";TPA	Send message on A-axis
JP #B	Jump to Label #B
EN	End Program

Hint: *AR is used to specify incremental distance from last AR or AD command. Use AR if multiple position trippoints are needed in a single motion sequence.*

AS

FUNCTION: At Speed

DESCRIPTION:

The AS command is a trippoint that occurs when the generated motion profile has reached the specified speed. This command will hold up execution of the following command until the commanded speed has been reached. The AS command will operate after either accelerating or decelerating. If the speed is not reached, the trippoint will be triggered after the speed begins diverging from the AS value.

ARGUMENTS: AS nnnnnnnnn where

n is A,B,C,D,E,F,G,H,S or any combination to specify the axis or sequence

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	-
Default Format	-

EXAMPLES:

#SPEED	Program SPEED
PR 100000	Specify position
SP 10000	Specify speed
BG A	Begin A
ASA	After speed is reached
MG "At Speed"	Print Message
EN	End of Program

WARNING:

The AS command applies to a trapezoidal velocity profile only with linear acceleration.. AS used with Smoothing profiling will be inaccurate.

@ASIN[n]

FUNCTION: Inverse sine

DESCRIPTION:

Returns in degrees the arc sine of the given number.

ARGUMENTS: @ASIN[n] where

n is a signed number in the range -1 to 1.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

@ACOS	Arc cosine
@SIN	Sine
@ATAN	Arc tangent
@COS	Cosine
@TAN	Tangent

EXAMPLES:

```
:MG @ASIN[-1]
 -90.0000
:MG @ASIN[0]
 0.0000
:MG @ASIN[1]
 90.0000
:
```

AT

FUNCTION: At Time

DESCRIPTION:

The AT command is a trippoint which is used to hold up execution of the next command until after the specified time has elapsed. The time is measured with respect to a defined reference time. AT 0 establishes the initial reference. AT n specifies n msec from the reference. AT -n specifies n msec from the reference and establishes a new reference after the elapsed time period.

ARGUMENTS: AT n where

n is a signed integer in the range 0 to 2 Billion

n = 0 defines a reference time at current time

n > 0 specifies a wait time of n msec from the reference time

n < 0 specifies a wait time of n msec from the reference time and re-sets the reference time when the trippoint is satisfied.

(AT -n is equivalent to AT n; AT <old reference +n>

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	0
Default Format	-

EXAMPLES:

The following commands are sent sequentially

AT 0	Establishes reference time 0 as current time
AT 50	Waits 50 msec from reference 0
AT 100	Waits 100 msec from reference 0
AT -150	Waits 150 msec from reference 0 and sets new reference at 150
AT 80	Waits 80 msec from new reference (total elapsed time is 230 msec)

@ATAN[n]

FUNCTION: Inverse tangent

DESCRIPTION:

Returns in degrees the arc tangent of the given number.

ARGUMENTS: @ATAN[n]

n is a signed number in the range -2147483647 to 2147483647

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

@ASIN	Arc sine
@SIN	sine
@ACOS	Arc cosine
@COS	Cosine
@TAN	Tangent

EXAMPLES:

```
:MG @ATAN[-10]
 -84.2894
:MG @ATAN[0]
 0.0000
:MG @ATAN[10]
 84.2894
:
```

#AUTO

FUNCTION: Subroutine to run automatically upon power up

DESCRIPTION:

#AUTO denotes code to run automatically when power is applied to the controller, or after the controller is reset. When no host software is used with the controller, #AUTO and the BP command are required to run an application program on the controller.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	No
Controller Usage	**ALL**

RELATED COMMANDS:

BP	Burn program

EXAMPLES:

```
#AUTO          ;'move the x axis upon power up
  PRX=1000     ;'move 1000 counts
  BGX          ;'begin motion
  AMX          ;'wait until motion is complete
EN
```

***NOTE**: Use EN to end the routine*

#AUTOERR

FUNCTION: Automatic subroutine for notification of EEPROM checksum errors

DESCRIPTION:

#AUTOERR will run code upon power up if data in the EEPROM has been corrupted. The EEPROM is considered corrupt if the checksum calculated on the bytes in the EEPROM do not match the checksum written to the EEPROM. The type of checksum error can be queried with _RS

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	No
Controller Usage	**ALL**

RELATED COMMANDS:

_RS	Checksum error code

EXAMPLES:

```
#AUTO
  WT 2000
  MG "AUTO"
  JP#AUTO
EN

#AUTOERR
  WT500
  MG "AUTOERR ", _RS
EN
```

***NOTE**: An application program must be executing for the automatic subroutine to function, which runs in thread 0.*

***NOTE**: Use EN to end the routine*

AV

FUNCTION: After Vector Distance

DESCRIPTION:

The AV command is a trippoint which is used to hold up execution of the next command during coordinated moves such as VP,CR or LI. This trippoint occurs when the path distance of a sequence reaches the specified value. The distance is measured from the start of a coordinated move sequence or from the last AV command. The units of the command are quadrature counts.

ARGUMENTS: AV s where

s is an unsigned integer in the range 0 to 2147483647 decimal. 's' represents the vector distance to be executed in the S coordinate system.

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	-
Command Line	Yes		
Controller Usage	**ALL CONTROLLERS**		

OPERAND USAGE:

_AVS contains the vector distance from the start of the sequence in the S coordinate system.

EXAMPLES:

#MOVE;DP 0,0	Label
LMAB	Linear move for A,B
LI 1000,2000	Specify distance
LI 2000,3000	Specify distance
LE	
BGS	Begin motion in the T coordinate system
AV500	After path distance = 500,
MG "Path>500";TPAB	Print Message
EN	End Program

Hint: *Vector Distance is calculated as the square root of the sum of the squared distance for each axis in the linear or vector mode.*

BA

FUNCTION: Brushless Axis

DESCRIPTION:

The BA command sets the controller up for sinusoidal commutation of the axis. This command enables the second DAC for use as the second phase of commutation.

Note: If a DMC-3425 is used for sinusoidal commutation, both axes are taken for a single axis of control. This controller will still take up two axes within the distributed control network. The DMC-3415, on the other hand, will control brushless motors with a single axis.

ARGUMENTS: BAx where

x specifies sinusoidal commutation for the axis.

No argument clears all axes configured for sinusoidal commutation.

USAGE:

While Moving	No	Default Value	0
In a Program	Yes	Default Format	0
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	Yes		
Controller Usage	**ALL CONTROLLERS**		

OPERAND USAGE:

_BA indicates whether the axis has been configured for sinusoidal commutation. If the motor is configured as brush-type or stepper motor, _BAx contains 0.

RELATED COMMANDS:

"BB"	Brushless Phase Begins
"BC"	Brushless Commutation
"BD"	Brushless Degrees
"BI"	Brushless Inputs
"BM"	Brushless Modulo
"BO"	Brushless Offset
"BS"	Brushless Setup
"BZ"	Brushless Zero

BB

FUNCTION: Brushless Phase Begins

DESCRIPTION:

The BB function describes the position offset between the Hall transition point and θ = 0, for a sinusoidally commutated motor. This command must be saved in non-volatile memory to be effective upon reset.

ARGUMENTS: BB n,n,n,n,n,n,n,n or BBA=n where

n represents the phase offset of the brushless motor, expressed in multiples of 30°.

USAGE:

While Moving	No	Default Value	0
In a Program	Yes	Default Format	0
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**ALL CONTROLLERS**		

EXAMPLES:

BB30	The offsets sinusoidal motor is 30°

RELATED COMMANDS:

“BA”	Brushless Axis
"BC"	Brushless Commutation
"BD"	Brushless Degrees
"BI"	Brushless Inputs
"BM"	Brushless Modulo
"BO"	Brushless Offset
"BS"	Brushless Setup
"BZ"	Brushless Zero

Note: BB is only effective as part of the BC command or upon reset.

BC

FUNCTION: Brushless Calibration

DESCRIPTION:

The function BC monitors the status of the Hall sensors of a sinusoidally commutated motor, and upon transition, replaces the estimated value of a commutated phase by an exact value.

ARGUMENTS: BCx where

x is the specified sinusoidal axis to be calibrated

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	0
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	Yes		
Controller Usage	**ALL CONTROLLERS**		

OPERAND USAGE:

_BC contains the state of the Hall sensor inputs. This value should be between 1 and 6.

RELATED COMMANDS:

"BA"	Brushless Axis
"BB"	Brushless Phase Begins
"BD"	Brushless Degrees
"BI"	Brushless Inputs
"BM"	Brushless Modulo
"BO"	Brushless Offset
"BS"	Brushless Setup
"BZ"	Brushless Zero

BD

FUNCTION: Brushless Degrees

DESCRIPTION:

This command sets the commutation phase of a sinusoidally commutated motor. When using hall effect sensors, a more accurate value for this parameter can be set by using the command, BC. This command should not be used except when the user is creating a specialized phase initialization procedure.

ARGUMENTS: BDn,n,n,n,n,n,n,n or BDA=n where

n sets the commutation phase in degrees within 0-360° for the specified sinusoidal axis.

USAGE:

While Moving	No	Default Value	0
In a Program	Yes	Default Format	0
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**ALL CONTROLLERS**		

OPERAND USAGE:

_BD contains the commutation phase of the brushless motor.

RELATED COMMANDS:

"BA"	Brushless Axis
"BB"	Brushless Phase Begins
"BC"	Brushless Commutation
"BI"	Brushless Inputs
"BM"	Brushless Modulo
"BO"	Brushless Offset
"BS"	Brushless Setup
"BZ"	Brushless Zero

BG

FUNCTION: Begin

DESCRIPTION:

The BG command starts a motion on the specified axis or sequence.

ARGUMENTS: BG nnnnnnnnnn where

n is A,B,C,D,E,F,G,H,S or N, or any combination to specify the axis or sequence

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	0
Default Format	-

OPERAND USAGE:

_BGa contains a '0' if motion complete on the specified axis or coordinate system, otherwise contains a '1'.

RELATED COMMANDS:

"AM"	After motion complete
"ST"	Stop motion

EXAMPLES:

PR 2000,3000,,5000	Set up for a relative move
BG ABD	Start the A,B and D motors moving
HM	Set up for the homing
BGA	Start only the A-axis moving
JG 1000,4000	Set up for jog
BGB	Start only the B-axis moving
BSTATE=_BGB	Assign a 1 to BSTATE if the B-axis is performing a move
VP 1000,2000	Specify vector position
VS 20000	Specify vector velocity
BGS	Begin coordinated sequen0ce
VMAB	Vector Mode
VP 4000,-1000	Specify vector position
VE	Vector End
PR ,,8000,5000	Specify C and D position
BGSCD	Begin sequence and C,D motion
MG _BGS	Displays a 1 if motion occurring on coordinated system "S"

Hint: *A BG command cannot be executed for any axis in which motion has not completed. Use the AM trippoint to wait for motion complete between moves. Determining when motion is complete can also be accomplished by testing for the value of the operand _BG.*

BI

FUNCTION: Brushless Inputs

DESCRIPTION:

The BIx indicates the starting number for the input lines to which the Hall sensors have been wired for sinusoidally commutated motors. These inputs must be the general use inputs (bits 1-7). The Hall sensors of the motor must be connected with consecutive numbers of input lines.

The brushless setup command, BS, can be used to determine the proper wiring of the hall sensors.

Note: Only the first three inputs of the DMC-3425 can be used for hall inputs, as the others are used for the Y axis limits, home and index inputs.

ARGUMENTS: BIn,n,n,n,n,n,n,n or BIA=n where

n indicates the starting input line for the specified sinusoidal axis.

0 clears the hall sensor configuration for the axis.

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	0
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**ALL CONTROLLERS**		

EXAMPLE:

BI1	The Hall sensors of the brushless motor are connected to inputs 1, 2 and 3.

RELATED COMMANDS:

"BA"	Brushless Axis
"BB"	Brushless Phase Begins
"BC"	Brushless Commutation
"BD"	Brushless Degrees
"BM"	Brushless Modulo
"BO"	Brushless Offset
"BS"	Brushless Setup
"BZ"	Brushless Zero

BK

FUNCTION: Breakpoint

DESCRIPTION:

For debugging. Causes the controller to pause execution of the given thread at the given program line number (which is not executed). All other threads continue running. Only one breakpoint may be armed at any time. After a breakpoint is encountered, a new breakpoint can be armed (to continue execution to the new breakpoint) or BK will resume program execution. The SL command can be used to single step from the breakpoint. The breakpoint can be armed before or during thread execution.

ARGUMENTS: BK n,m where

n is an integer in the range 0 to 999 which is the line number to stop at. n must be a valid line number in the chosen thread.

m is an integer in the range 0 to 7. The thread.

USAGE:

While Moving	Yes
In a Program	No
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value of m	0

OPERAND USAGE:

_BK will tell whether a breakpoint has been armed, whether it has been encountered, and the program line number of the breakpoint:

= -LineNumber: breakpoint armed

= LineNumber: breakpoint encountered

= -2147483648: breakpoint not armed

RELATED COMMANDS:

"SL"	Single Step
"TR"	Trace

EXAMPLES:

BK 3	Pause at line 3 (the 4th line) in thread 0
BK 5	Continue to line 5
SL	Execute the next line
SL 3	Execute the next 3 lines
BK	Resume normal execution

BL

FUNCTION: Reverse Software Limit

DESCRIPTION:

The BL command sets the reverse software limit. If this limit is exceeded during motion, motion on that axis will decelerate to a stop. Reverse motion beyond this limit is not permitted.

When the reverse software limit is activated, the automatic subroutine #LIMSWI will be executed if it is included in the program and a program is executing. See User's Manual, Automatic Subroutine.

ARGUMENTS: BL n,n,n,n,n,n,n,n or BLA=n where

n is a signed integer in the range -2147483648 to 2147483647. The reverse limit is activated at the position n-1. The units are in quadrature counts.

n = -214783648 Turns off the reverse limit.

n = ? Returns the reverse software limit for the specified axes.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	-214783648
Default Format	Position format

OPERAND USAGE:

_BLa contains the value of the reverse software limit for the specified axis.

RELATED COMMANDS:

"FL"	Forward Limit
"PF"	Position Formatting

EXAMPLES:

#TEST	Test Program
AC 1000000	Acceleration Rate
DC 1000000	Deceleration Rate
BL -15000	Set Reverse Limit
JG -5000	Jog Reverse
BGA	Begin Motion
AMA	After Motion (limit occurred)
TPA	Tell Position
EN	End Program

Hint: *Galil Controllers also provide hardware limits.*

BM

FUNCTION: Brushless Modulo

DESCRIPTION:

The BM command defines the length of the magnetic cycle in encoder counts.

ARGUMENTS: BMn,n,n,n,n,n,n,n or BMA=n where

n is a decimal value between 1 and 1000000 with a resolution of 1/10 that represents the magnetic cycles of the brushless motor. This value can also be specified as a fraction with a resolution of 1/16.

USAGE:

While Moving	No	Default Value	0
In a Program	Yes	Default Format	0
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**ALL CONTROLLERS**		

OPERAND USAGE:

_BMx indicates the cycle length in counts for the brushless motor.

RELATED COMMANDS:

"BA"	Brushless Axis
"BB"	Brushless Phase Begins
"BC"	Brushless Commutation
"BD"	Brushless Degrees
"BI"	Brushless Inputs
"BO"	Brushless Offset
"BS"	Brushless Setup
"BZ"	Brushless Zero

Note: *Changing the BM parameter causes an instant change in the commutation phase.*

BN

FUNCTION: Burn

DESCRIPTION:

The BN command saves controller parameters shown below in Flash EEPROM memory. This command typically takes 1 second to execute and must not be interrupted. The controller returns a: when the Burn is complete.

PARAMETERS SAVED DURING BURN:

AC	CO	GA	KP	SB
AF	CW	GM	KS	SP
BA	DC	GR	LZ	TL
BB	DV	HC	MO	TM
BI	EI	HR	MT	TR
BL	EO	IA	OE	VA
BM	ER	IL	OF	VD
BO	FA	IT	OP	VF
CE	FL	KD	PF	VS
CN	FV	KI	PL	VT

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	-
Default Format	-

OPERAND USAGE:

_BN contains the serial number of the controller.

RELATED COMMANDS:

"BP"	Burn Program
"BV"	Burn Variables

EXAMPLES:

KD 100	Set damping term for A axis
KP 10	Set proportional gain term for A axis
KI 1	Set integral gain term for A axis
AC 200000	Set acceleration
DC 150000	Set deceleration rate
SP 10000	Set speed
MT -1	Set motor type for A axis to be type '-1', reversed polarity servo motor
MO	Turn motor off
BN	Burn parameters; may take up to 15 seconds

BO

FUNCTION: Brushless Offset

DESCRIPTION:

The BOx sets a fixed offset on the DAC's of a sinusoidally commutated motor. This may be used to offset any bias in the amplifier, or can be used for phase initialization.

ARGUMENTS: BOn,n,n,n,n,n,n,n or BOA=n where

n specifies the voltage as a real value between -10 and 10.

n = ? Returns the brushless offset for the 'x' axis.

USAGE:

While Moving	No	Default Value	0
In a Program	Yes	Default Format	0
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**ALL CONTROLLERS**		

OPERAND USAGE:

_BOx contains the offset voltage on the DAC for the brushless motor.

EXAMPLES:

BO –2,1	Generates –2 volts on the first DAC and +1 volts on the second DAC of a sinusoidally commutated motor.

RELATED COMMANDS:

"BA"	Brushless Axis
"BB"	Brushless Phase Begins
"BC"	Brushless Commutation
"BD"	Brushless Degrees
"BI"	Brushless Inputs
"BM"	Brushless Modulo
"BS"	Brushless Setup
"BZ"	Brushless Zero

HINT: To assure that the output voltage equals the BO parameters, set the PID and DF parameters to zero.

BP

FUNCTION: Burn Program

DESCRIPTION::

The BP command saves the application program in non-volatile EEPROM memory. This command typically takes up to 10 seconds to execute and must not be interrupted. The controller returns a: when the Burn is complete.

ARGUMENTS: None

USAGE:

While Moving	No
In a Program	No
Not in a Program	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	---

RELATED COMMANDS:

"BN"	Burn Parameters
"BV"	Burn Variable

Note: This command may cause the Galil software to issue the following warning "A time-out occurred while waiting for a response from the controller". This warning is normal and is designed to warn the user when the controller does not respond to a command within the timeout period. This occurs because this command takes more time than the default timeout of 1 sec. The timeout can be changed in the Galil software but this warning does not affect the operation of the controller or software.

BS

FUNCTION: Brushless Setup

DESCRIPTION:

The command BS tests the wiring of a sinusoidally commutated brushless motor. If no Hall sensors are connected, the function tests the wiring of the DAC's. If Hall sensors are connected, the function also tests the wiring of the Hall sensors. The first parameter indicates the voltage level to be applied to each phase, and the second parameter indicates the duration in milliseconds that the voltage will be applied.

This command returns status information regarding the setup of brushless motors. The following information will be returned by the controller:

1. Correct wiring of the brushless motor phases.
2. An approximate value of the motor's magnetic cycle.
3. The value of the BB command (If hall sensors are used).
4. The results of the hall sensor-wiring test (If hall sensors are used).

This command will turn the motor off when done and may be given when the motor is off.

Once the brushless motor is properly setup and the motor configuration has been saved in non-volatile memory, the BS command does not have to be re-issued. Using the burn command, BN, saves the configuration.

Note: In order to properly conduct the brushless setup, the motor must be allowed to move a minimum of one magnetic cycle in both directions.

ARGUMENTS: BSx= V, n where

x is the specified sinusoidal axis, V is a real number between 0 and 10, and n is a positive integer between 100 or 1000.

USAGE:

While Moving	No	Default Value of V	0
In a Program	Yes	Default Value of n	200
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	No		
Controller Usage	**ALL CONTROLLERS**		

EXAMPLES:

BS 2,900	Apply set up test to Z axis with 2 volts for 900 millisecond on each step.

RELATED COMMANDS:

"BA"	Brushless Axis
"BB"	Brushless Phase Begins
"BC"	Brushless Commutation
"BD"	Brushless Degrees
"BI"	Brushless Inputs
"BM"	Brushless Modulo
"BO"	Brushless Offset
"BZ"	Brushless Zero

Note: When using Galil Window's software, the timeout must be set to a minimum of 10seconds (timeout = 10000) when executing the BS command. This allows the software to retrieve all messages returned from the controller.

BV

FUNCTION: Burn Variables

DESCRIPTION:

The BV command saves the controller variables in non-volatile EEPROM memory. This command typically takes up to 2 seconds to execute and must not be interrupted. The controller returns a : when the Burn is complete.

ARGUMENTS: None

USAGE:

While Moving	Yes
In a Program	Yes
Not in a Program	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	---

RELATED COMMANDS:

“BN”	Burn Parameters
"BP"	Burn Program

Note: This command may cause the Galil software to issue the following warning "A time-out occurred while waiting for a response from the controller". This warning is normal and is designed to warn the user when the controller does not respond to a command within the timeout period. This occurs because this command takes more time than the default timeout of 1 sec. The timeout can be changed in the Galil software but this warning does not affect the operation of the controller or software.

BZ

FUNCTION: Brushless Zero

DESCRIPTION:

The BZ command is when the controller is configured for sinusoidal commutation. This command drives the motor to zero magnetic phase and then sets the commutation phase to zero.

This command may be given when the motor is off.

ARGUMENTS: BZn,n,n,n,n,n,n,n or BZA=n where

n is a real number between -9.998 and 9.998.

The parameter x sets the voltage to be applied to the amplifier during the initialization. In order to be accurate, the BZ command must be large enough to move the motor. When the argument is positive, the process ends up in MO state. A negative value causes the process to end up in the SH state.

Note: The BZ command causes instantaneous movement of the motor. It is recommended to start with small voltages and increase as needed.

Note: Always use the Off-On-Error function (OE Command) to avoid motor runaway whenever testing sinusoidal commutation.

USAGE:

While Moving	No	Default Value	0
In a Program	Yes	Default Format	0
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	No		
Controller Usage	**ALL CONTROLLERS**		

OPERAND USAGE:

_BZx contains the distance in encoder counts from the motor's current position and the position of commutation zero. This can useful to command a motor to move to the commutation zero position for phase initialization.

EXAMPLES:

BZ -3	Drive the motor to zero phase position with 3 volts signal, and end up in SH state.

RELATED COMMANDS:

"BA"	Brushless Axis
"BB"	Brushless Phase Begins
"BC"	Brushless Commutation
"BD"	Brushless Degrees
"BI"	Brushless Inputs
"BM"	Brushless Modulo
"BO"	Brushless Offset
"BS"	Brushless Setup

CB

FUNCTION: Clear Bit

DESCRIPTION:

The CB command sets an output bit low. CB can be used to clear outputs of extended I/O (when the I/O has been configured to operate as outputs). CB can also be used by a master controller to clear the outputs on slave controllers and IOC-7007 controllers when used in a distributed control system.

ARGUMENTS: CB n where

n is an integer corresponding to a specific output on the controller to be cleared (set to 0). The first output on the controller is denoted as output 1.

The outputs of the slave devices are calculated using the following formula:

n = (HandleNum*100) + (Bitnum) where

HandleNum is the number associated with the handle specified for the particular slave. 1 for handle A, 2 for handle B, etc.

BitNum is the specific output of the slave device. This includes extended I/O that has been configured to operate as outputs.

The outputs on an IOC-7007 I/O controller may also be set through the master controller. The outputs for the IOC-7007's IOM modules are calculated using the following formula:

n = ((HandleNum*1000) + (Bitnum) where

HandleNum is the number associated with the handle specified for the particular IOC controller. 1 for handle A, 2 for handle B, etc.

Bitnum is the specific output bit on the IOC controller to be set or cleared.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	--
Default Format	--

RELATED COMMANDS:

"SB"	Set Bit
"OB"	Output Bit
"OP"	Define output port (byte-wise).

EXAMPLES:

CB 7	Clear output bit 7
CB 202	Clear output bit 2 for the slave controller on handle 2 (B).

CD

FUNCTION: Contour Data

DESCRIPTION:

The CD command specifies the incremental position on A, B, C and D axes. The units of the command are in encoder counts. This command is used only in the Contour Mode (CM). The incremental position will be executed over the time period specified by the command DT (ranging from 2 to 256 servo updates)

ARGUMENTS: CD n,n,n,n,n,n,n,n or CDA=n where

n is an integer in the range of +/-32762.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS, LOCAL AXES ONLY**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

"CM"	Contour Mode
"WC"	Wait for Contour
"DT"	Time Increment
"CS"	_CS is the Segment Counter

EXAMPLES:

CM ABCD	Specify Contour Mode
DT 4	Specify time increment for contour
CD 200,350,-150,500	Specify incremental positions on A,B,C and D axes A-axis moves 200 counts B-axis moves 350 counts C-axis moves -150 counts D-axis moves 500 counts
WC	Wait for complete
CD 100,200,300,400	New position data
WC	Wait for complete
DT0	Stop Contour
CD 0,0,0,0	Exit Mode

CE

FUNCTION: Configure Encoder

DESCRIPTION:

The CE command configures the encoder to the quadrature type or the pulse and direction type. It also allows inverting the polarity of the encoders that reverses the direction of the feedback. Note: when using a servomotor, the motor will run away. The configuration applies independently to the main axes encoders and the auxiliary encoders.

ARGUMENTS: CE n,n,n,n,n,n,n,n or CEA=n where

n is an integer in the range of 0 to 3. The values of M are

n =	MAIN ENCODER TYPE
0	Normal quadrature
1	Normal pulse and direction
2	Reversed quadrature
3	Reversed pulse and direction

n = ? Returns the value of the encoder configuration for the specified axes.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	O
Default Format	2.0

OPERAND USAGE:

_CEa contains the value of encoder type for the axis specified by ‘a’.

RELATED COMMANDS:

"MT"	Specify motor type

EXAMPLES:

CE 0, 3, 6, 2	Configure encoders
CE ?,?,?,?	Interrogate configuration
V = _CEA	Assign configuration to a variable

Note: *When using pulse and direction encoders, the pulse signal is connected to CHA and the direction signal is connected to CHB.*

CF

FUNCTION: Configure

DESCRIPTION:

Sets the default port for unsolicited messages. By default, the DMC-3425 will send unsolicited responses to Ethernet Handle A. The CF command allows the user to send unsolicited responses to the Serial Port, or Handles A – H.

ARGUMENTS: CF n where

n is A thru H for Ethernet handles 1 thru 8, S for Main serial port.

USAGE:

While Moving	Yes	Default Value	A
In a Program	Yes	Default Format	--
Command Line	No		
Controller Usage	**ALL CONTROLLERS**		

OPERAND USAGE:

_CF contains the decimal value of the default port specified with the CF command. Handle A through H are represented by decimal 65 – 72 respectively. Decimal 83 represent the serial port S.

EXAMPLES:

CFB	Send unsolicited messages out Handle B
CFS	Send unsolicited messages out Serial Port

CH

FUNCTION: Configure Handle

DESCRIPTION:

The CH command is used to associate master and slave controllers in a distributed control system. The master controller must associate 1 Ethernet handle for sending commands to each slave, and 1 Ethernet handle for receiving status information from each slave. Note that these handles must first be opened before assigning them with this command, see the command IH.

This command is not necessary when using the automatic setup procedure HC.

ARGUMENTS: CHx=h1,h2 where

x is A,B,C,D,E,F,G or H. This is the first axis of the slave controller. X,Y,Z and W may be interchanged with A,B,C,D.

h1 is the handle to be used to send commands to the slave controller.

h2 is the handle to be used for receiving status from the slave controller (QW).

In a UDP system, h1 and h2 may be the same handle. This is not recommended in TCP/IP systems.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	--
Default Format	--

RELATED COMMANDS:

"IH"	Set Internet Handles
"NA"	Set Number of Axes for Distributed Control System
“QW”	Set Slave Data Record Update Rate

EXAMPLES:

CHC=A,B Using one DMC-3425 as a master and one DMC-3425 as a slave. This command assigns the slave, identified by the C axis designator, with Handle A for commands and Handle B for status returned from the slave.

CM

FUNCTION: Contour Mode

DESCRIPTION:

The Contour Mode is initiated by the instruction CM. This mode allows the generation of an arbitrary motion trajectory with any of the axes. The CD command specified the position increment, and the DT command specifies the time interval.

The command, CM?, can be used to check the status of the Contour Buffer. A value of 1 returned from the command CM? indicates that the Contour Buffer is full. A value of 0 indicates that the Contour Buffer is empty.

ARGUMENTS: CM nnnnnnnn where

n is A,B,C,D,E,F,G or any combination to specify the axis (axes) for contour mode

n = ? Returns a 1 if the contour buffer is full and 0 if the contour buffer is empty.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS, LOCAL AXES ONLY**

DEFAULTS:

Default Value	0
Default Format	2.0

OPERAND USAGE:

_CM contains a '0' if the contour buffer is empty, otherwise contains a '1'.

RELATED COMMANDS:

"CD"	Contour Data
"WC"	Wait for Contour
"DT"	Time Increment

EXAMPLES:

V=_CM;V=	Return contour buffer status
CM?	Return contour buffer status
CM AB	Specify A,B axes for Contour Mode

#CMDERR

FUNCTION: Command error automatic subroutine

DESCRIPTION:

Without #CMDERR defined, if an error (see TC command) occurs in an application program running on the Galil controller, the program (all threads) will stop. #CMDERR allows the programmer to handle the error by running code instead of stopping the program.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	No
Controller Usage	**ALL**

RELATED COMMANDS:

TC	Tell Error Code
_ED	Last program line with an error
EN	End routine

EXAMPLES:

```
#BEGIN                      ;'Begin main program
  IN "ENTER SPEED",Speed    ;'Prompt for speed
  JG Speed
  BGX                       ;'Begin motion
EN                          ;'End main program

#CMDERR                     ;'Command error utility
  JP#DONE,_ED<>2            ;'Check if error on line 2
  JP#DONE,_TC<>6            ;'Check if out of range
  MG "SPEED TOO HIGH"       ;'Send message
  MG "TRY AGAIN"            ;'Send message
  ZS1                       ;'Adjust stack
  JP #BEGIN                 ;'Return to main program
  #DONE                     ;'End program if other error
  ZS0                       ;'Zero stack
EN1                         ;'End routine
```

NOTE: An application program must be executing for the automatic subroutine to function, which runs in thread 0.

NOTE: Use EN to end the routine

CN

FUNCTION: Configure

DESCRIPTION:

The CN command configures the polarity of the limit switches, home switches, latch inputs and the selective abort function.

ARGUMENTS: CN m,n,o where

m,n,o are integers with values 1 or -1.

p is an integer, 0 or 1.

m =	1	Limit switches active high
	-1	Limit switches active low
n =	1	Home switch configured to drive motor in forward direction when input is high. See HM and FE commands.
	-1	Home switch configured to drive motor in reverse direction when input is high. See HM and FE commands
o =	1	Latch input is active high
	-1	Latch input is active low

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	-1,-1,-1,0
Default Format	2.0

OPERAND USAGE:

_CN0	Contains the limit switch configuration
_CN1	Contains the home switch configuration
_CN2	Contains the latch input configuration

RELATED COMMANDS:

"AL"	Arm latch
“HM”	Home axis

EXAMPLES:

CN 1,1	Sets limit and home switches to active high
CN,, -1	Sets input latch active low

CO

FUNCTION: Configure Extended I/O

DESCRIPTION:

The CO command configures which points are inputs and which are outputs on the DB-14064 extended I/O. The extended I/O is denoted as bits 17-80 and banks 2-9 on each controller.

ARGUMENTS: CO n where

n is a decimal value which represents a binary number. Each bit of the binary number represents one block of extended I/O. When set to 1, the corresponding block is configured as an output.

The least significant bit represents bank 2 and the most significant bit represents bank 9. The decimal value can be calculated by the following formula.

$n = n_2 + 2*n_3 + 4*n_4 + 8*n_5 +16* n_6 +32* n_7 +64* n_8 +128* n_9$

Where n_x represents the bank. If the n_x value is a one, then the bank of 8 I/O points is to be configured as an output. If the n_x value is a zero, then the bank of 8 I/O points will be configured as an input. For example, if banks 3 and 4 are to be configured as outputs, CO 6 is issued.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS WITH DB-14064**

DEFAULTS:

Default Value	0
Default Format	--

OPERAND USAGE:

_CO returns the extended I/O configuration value.

RELATED COMMANDS:

"CB"	Clear Output Bit
"SB"	Set Output Bit
"OP"	Set Output Port
"TI"	Tell Inputs

EXAMPLES:

CO 255	Configure all points as outputs
CO 0	Configure all points as inputs
CO 1	Configures bank 1 to outputs on extended I/O

NOTE: The CO command must be sent to slave controllers using the SA command.

Hint: *See user manual appendix for more information on the extended I/O boards.*

@COM[n]

FUNCTION: Bitwise complement

DESCRIPTION:

Performs the bitwise complement (NOT) operation to the given number

ARGUMENTS: @COM[n] where

n is a signed integer in the range -2147483647 to 2147483647.

The integer is interpreted as a 32-bit field.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

& |	Logical operators AND and OR

EXAMPLES:

```
:MG {$8.0} @COM[0]
$FFFFFFFF
:MG {$8.0} @COM[$FFFFFFFF]
$00000000
:
```

@COS[n]

FUNCTION: Cosine

DESCRIPTION:

Returns the cosine of the given angle in degrees

ARGUMENTS: @COS[n] where

n is a signed number in degrees in the range of -32768 to 32767, with a fractional resolution of 16-bit..

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

@ASIN	Arc sine
@SIN	sine
@ATAN	Arc tangent
@ACOS	Arc cosine
@TAN	Tangent

EXAMPLES:

```
:MG @COS[0]
 1.0000
:MG @COS[90]
 0.0000
:MG @COS[180]
 -1.0000
:MG @COS[270]
 0.0000
:MG @COS[360]
 1.0000
:
```

CR

FUNCTION: Circle

DESCRIPTION:

The CR command specifies a 2-dimensional arc segment of radius, r, starting at angle, θ, and traversing over angle Δθ. A positive Δθ denotes counterclockwise traverse, negative Δθ denotes clockwise. The VE command must be used to denote the end of the motion sequence after all CR and VP segments are specified. The BG (Begin Sequence) command is used to start the motion sequence. All parameters, r, θ, Δθ, must be specified. Radius units are in quadrature counts. θ and Δθ have units of degrees. The parameter n is optional and describes the vector speed that is attached to the motion segment.

ARGUMENTS: CR r,θ,Δθ < n > o where

r is an unsigned real number in the range 10 to 6000000 decimal (radius)

θ a signed number in the range 0 to +/-32000 decimal (starting angle in degrees)

Δθ is a signed real number in the range 0.0001 to +/-32000 decimal (angle in degrees)

n specifies a vector speed to be taken into effect at the execution of the vector segment. n is an unsigned even integer between 0 and 12,000,000 for servo motor operation and between 0 and 3,000,000 for stepper motors.

o specifies a vector speed to be achieved at the end of the vector segment. o is an unsigned even integer between 0 and 8,000,000.

Note: The product r * Δθ must be limited to +/-4.5 10^8

USAGE:

While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	-
Command Line	Yes		
Controller Usage	**ALL CONTROLLERS, LOCAL AXES ONLY**		

DEFAULTS:

RELATED COMMANDS:

"VP"	Vector Position
"VS"	Vector Speed
"VD"	Vector Deceleration
"VA"	Vector Acceleration
"VM"	Vector Mode
"VE"	End Vector
"BG"	BGS - Begin Sequence

EXAMPLES:

VMAB	Specify vector motion in the A and B plane
VS 10000	Specify vector speed
CR 1000,0,360	Generate circle with radius of 1000 counts, start at 0 degrees and finish at 360 degrees
VE	End Sequence
BGS	Start motion

CS

FUNCTION: Clear Sequence

DESCRIPTION:

The CS command will remove VP, CR or LI commands stored in a motion sequence for the S coordinate system. After a sequence has been executed, the CS command is not necessary to put in a new sequence. This command is useful when you have incorrectly specified VP, CR or LI commands.

ARGUMENTS: CS is used to clear the sequence buffer for the "S" coordinate system.

USAGE:

While Moving	No
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	---
Default Format	---

OPERAND USAGE:

_CS contains the segment number in the sequence. This operand is valid in the Linear mode, LM, and Vector mode, VM

RELATED COMMANDS:

"CR"	Circular Interpolation Segment
"LI"	Linear Interpolation Segment
"LM"	Linear Interpolation Mode
"VM"	Vector Mode
"VP"	Vector Position

EXAMPLES:

#CLEAR	Label
VP 1000,5000	New vector
VP 8000,9000	New vector
CS	Clear vectors specified in S coordinate system

CW

FUNCTION: Copyright information / Data Adjustment bit on/off

DESCRIPTION:

The CW command has a dual usage. The CW command will return the copyright information when the argument, n is 0. Otherwise, the CW command is used as a communications enhancement for use by the Servo Design Kit software. When turned on, the communication enhancement sets the MSB of unsolicited, returned ASCII characters to 1. Unsolicited ASCII characters are those characters that are returned from the controller without being directly queried from the terminal. This is the case when a program has a command that requires the controller to return a value or string.

ARGUMENTS: CW n,m where

n = 0	Causes the controller to return the copyright information
n = 1	Causes the controller to set the MSB of unsolicited returned characters to 1
n = 2	Causes the controller to not set the MSB of unsolicited characters.
n = ?	Returns the copyright information for the controller.
m = 0	Causes the controller to pause program execution when output FIFO is full until FIFO no longer full.
m = 1	Causes the controller to continue program execution when output FIFO is full output characters after FIFO is full will be lost.

Note: The DMC-3425 has only a single byte UART (Buffer). Due to this, it is not recommended using the controller in the CW,1 mode if data is to be output. CW,1 should be used as a setting when messages exist in a program for debug, but are not currently needed by the user.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	2, 0
Default Format	

OPERAND USAGE:

_CW contains the value of the data adjustment bit. 2 = off, 1 = on

Note: *The CW command can cause garbled characters to be returned by the controller. The default state of the controller is to disable the CW command, however, the Galil Servo Design Kit software and terminal software may sometimes enable the CW command for internal usage. If the controller is reset while the Galil software is running, the CW command could be reset to the default value that would create difficulty for the software. It may be necessary to re-enable the CW command. The CW command status can be stored in EEPROM*

DA

FUNCTION: De-allocate the Variables & Arrays

DESCRIPTION:

The DA command frees the array and/or variable memory space. In this command, more than one array or variable can be specified for memory de-allocation. Different arrays and variables are separated by comma when specified in one command. The argument * de-allocates all the variables, and *[0] de-allocates all the arrays.

ARGUMENTS: DA c[0],variable-name where

c[0] = Defined array name

variable-name = Defined variable name

* - De-allocates all the variables

*[0] – De-allocates all the arrays

DA? Returns the number of arrays available on the controller.

USAGE:		**DEFAULTS:**	
While Moving	Yes		Default Value
In a Program	Yes		Default Format
Command Line	Yes		
Controller Usage	**ALL CONTROLLERS**		

OPERAND USAGE:

_DA contains the total number of arrays available. For example, before any arrays have been defined, the operand _DA is 14. If one array is defined, the operand _DA will return 13.

RELATED COMMANDS:

"DM"	Dimension Array

EXAMPLES: 'Cars' and 'Sales' are arrays and 'Total' is a variable.

DM Cars [400], Sales [50]	Dimension 2 arrays
Total=70	Assign 70 to the variable Total
DA Cars [0], Sales [0], Total	De-allocate the 2 arrays & variables
DA*[]	De-allocate all arrays
DA *,*[]	De-allocate all variables and all arrays

Note: *Since this command de-allocates the spaces and compacts the array spaces in the memory, it is possible that execution of this command may take longer time than 2 ms.*

DC

FUNCTION: Deceleration

DESCRIPTION:

The Deceleration command (DC) sets the linear deceleration rate of the motors for independent moves such as PR, PA and JG moves. The parameters will be rounded down to the nearest factor of 1024 and have units of counts per second squared.

ARGUMENTS: DC n,n,n,n,n,n,n,n or DCA=n where

n is an unsigned numbers in the range 1024 to 67107840

n = ? Returns the deceleration value for the specified axes.

USAGE:

While Moving	Yes*
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	256000
Default Format	8.0

* When moving, the DC command can only be specified while in the jog mode.

OPERAND USAGE:

_DCa contains the deceleration rate for the specified axis.

RELATED COMMANDS:

"AC"	Acceleration
"PR"	Position Relative
"PA"	Position Absolute
"SP"	Speed
"JG"	Jog

EXAMPLES:

PR 10000	Specify position
AC 2000000	Specify acceleration rate
DC 1000000	Specify deceleration rate
SP 5000	Specify slew speed
BG	Begin motion

Note: *The DC command may be changed during the move in JG move, but not in PR or PA move.*

DE

FUNCTION: Dual (Auxiliary) Encoder Position

DESCRIPTION:

The DE command defines the position of the auxiliary encoders.

The DE command defines the encoder position when used with stepper motors.

Note: The auxiliary encoders are not available for the stepper axis or for any axis where output compare is active.

ARGUMENTS: DE n,n,n,n,n,n,n,n or DEA=n where

n is a signed integers in the range -2147483647 to 2147483648 decimal

n = ? Returns the position of the auxiliary encoders for the specified axes.

n = ? returns the commanded reference position of the motor (in step pulses) when used with a stepper motor. Example: DE 0 This will define the TP or encoder position to 0. This will not effect the DE ? value. (To set the DE value when in stepper mode uses the DP command.)

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	0,0,0,0
Default Format	Position Format

OPERAND USAGE:

_DEa contains the current position of the specified auxiliary encoder.

RELATED COMMANDS:

"DP"	Define main encoder position
"TD"	Tell Dual Encoder position

EXAMPLES:

DE ,,,,,0	Set the current auxiliary encoder position to 0 on A axis
DE ,,,,,?	Return auxiliary encoder positions
DUALG=_DEG	Assign auxiliary encoder position of G-axis to the variable DUALG

Hint: Dual encoders are useful when you need an encoder on the motor and on the load. The encoder on the load is typically the auxiliary encoder and is used to verify the true load position. Any error in load position is used to correct the motor position.

DL

FUNCTION: Download

DESCRIPTION:

The DL command transfers a data file from the host computer to the controller. Instructions in the file will be accepted as a data stream without line numbers. The file is terminated using <control> Z, <control> Q, <control> D, or \. DO NOT insert spaces before each command.

If no parameter is specified, downloading a data file will clear all programs in the controllers RAM. The data is entered beginning at line 0. If there are too many lines or too many characters per line, the controller will return a?. To download a program after a label, specify the label name following DL. The argument # may be used with DL to append a file at the end of the program in RAM.

Using Galil DOS Terminal Software: The ED command puts the controller into the Edit subsystem. In the Edit subsystem, programs can be created, changed, or destroyed. The commands in the Edit subsystem are:

<cntrl>D	Deletes a line
<cntrl>I	Inserts a line before the current one
<cntrl>P	Displays the previous line
<cntrl>Q	Exits the Edit subsystem
<return>	Saves a line

ARGUMENTS: DL n where

n = no argument	Downloads program beginning at line 0. Erases programs in RAM.
n = #Label	Begins download at line following #Label
n = #	Begins download at end of program in RAM.

USAGE:

While Moving	Yes
In a Program	No
Command Line	Yes
Controller Usage	**ALL CONTROLLERS WITH DIRECT CONNECTION**

DEFAULTS:

Default Value	---
Default Format	---

OPERAND USAGE:

When used as an operand, _DL gives the number of available labels.

All Econo or E-Series controllers have 126 available labels

RELATED COMMANDS:

"UL"	Upload

EXAMPLES:

DL;	Begin download
#A;PR 4000;BGA	Data
AMA;MG DONE	Data
EN	Data
<control> Z	End download

DM

FUNCTION: Dimension

DESCRIPTION:

The DM command defines a single dimensional array with a name and the number of elements in the array. The first element of the defined array starts with element number 0 and the last element is at n-1.

ARGUMENTS: DM c[n] where

c is a name of up to eight characters, starting with an uppercase alphabetic character. n specifies the size of the array (number of array elements).

n = ? Returns the number of array elements available.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	---
Default Format	---

OPERAND USAGE:

_DM contains the available array space. For example, before any arrays have been defined, the operand _DM will return 2000. If an array of 100 elements is defined, the operand _DM will return 1900.

RELATED COMMANDS:

"DA"	De-allocate Array

EXAMPLES:

DM Pets[5],Dogs[2],Cats[3]	Define dimension of arrays, pets with 5 elements; Dogs with 2 elements; Cats with 3 elements
DM Tests[1600]	Define dimension of array Tests with 1600 elements

DP

FUNCTION: Define Position

DESCRIPTION:

The DP command sets the current motor position and current command positions to a user specified value. The units are in quadrature counts. This command will set both the TP and RP values.

The DP command sets the current commanded reference position for axes configured as steppers. The units are in steps. Example: DP0 This will set the registers for TD and RP to zero, but will not effect the TP register.

ARGUMENTS: DP n,n,n,n,n,n,n,n or DPA=n where

n is a signed integer in the range -2147483648 to 2147483647 decimal.

n = ? Returns the current position of the motor for the specified axes.

USAGE:

While Moving	No
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	0,0,0,0
Default Format	Position Format

OPERAND USAGE:

_DPa contains the current position of the specified axis.

RELATED COMMANDS:

"PF"	Position Formatting

EXAMPLES:

DP 0,100,200,400	Sets the current position of the A-axis to 0, the B-axis to 100, the C-axis to 200, and the D-axis to 400
DP ,-50000	Sets the current position of B-axis to -50000. The B,C and D axes remain unchanged.
DP ?,?,?,?	Interrogate the position of A, B, C and D axis.
0000000,-0050000,0000200,0000400	Returns all the motor positions
DP ?	Interrogate the position of A axis
0000000	Returns the A-axis motor position

Hint: *The DP command is useful to redefine the absolute position. For example, you can manually position the motor by hand using the Motor Off command, MO. Turn the servo motors back on with SH and then use DP0 to redefine the new position as your absolute zero.*

DT

FUNCTION: Delta Time

DESCRIPTION:

The DT command sets the time interval for Contour Mode. Sending the DT command once will set the time interval for all contour data until a new DT command is sent. 2^n milliseconds is the time interval. (Followed by CD0 command).

ARGUMENTS: DT n where

n is an integer in the range 0 to 8.

n=0 terminates the Contour Mode.

n=1 through 8 specifies the time interval of 2^n samples.

By default the sample period is 1 msec (set by the TM command); with n=1, the time interval would be 2 msec

n = ? Returns the value for the time interval for contour mode.

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	1.0
Command Line	Yes		
Controller Usage	**ALL CONTROLLERS, LOCAL AXES ONLY**		

OPERAND USAGE:

_DT contains the value for the time interval for Contour Mode

RELATED COMMANDS:

"CM"	Contour Mode
"CD"	Contour Data
"WC"	Wait for next data

EXAMPLES:

DT 4	Specifies time interval to be 16 msec
DT 7	Specifies time interval to be 128 msec
#CONTOUR	Begin
CMAB	Enter Contour Mode
DT 4	Set time interval
CD 1000,2000	Specify data
WC	Wait for contour
CD 2000,4000	New data
WC	Wait
DT0	Stop contour
CD0	Exit Contour Mode
EN	End

EA

FUNCTION: Choose ECAM master

DESCRIPTION:

The EA command selects the master axis for the electronic cam mode. Any axis may be chosen.

ARGUMENTS: EA n where

n is one of the axis specified as A,B,C,D,E,F,G or H

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	
In a Program	Yes	Default Format	
Command Line	Yes		
Controller Usage	**ALL CONTROLLERS, LOCAL AXES ONLY**		

RELATED COMMANDS:

"EB"	Enable ECAM
"EC"	Set ECAM table index
"EG"	Engage ECAM
“EM”	Specify ECAM cycle
"EP"	Specify ECAM table intervals & staring point
“EQ”	Disengage ECAM
"ET"	ECAM table

EXAMPLES:

EAB	Select B as a master for ECAM

EB

FUNCTION: Enable ECAM

DESCRIPTION:

The EB function enables or disables the cam mode. In this mode, the starting position of the master axis is specified within the cycle. When the EB command is given, the master axis is modularized.

ARGUMENTS: EB n where

n = 1 Starts ECAM mode

n = 0 Stops ECAM mode.

n = ? Returns 0 if ECAM is disabled and a 1 if enabled.

USAGE:

While Moving	Yes	Default Value
In a Program	Yes	Default Format
Command Line	Yes	
Controller Usage	**ALL CONTROLLERS, LOCAL AXES ONLY**	

DEFAULTS:

OPERAND USAGE:

_EB contains the state of Ecam mode. 0 = disabled, 1 = enabled

RELATED COMMANDS:

"EA"	Choose ECAM master
"EC"	Set ECAM table index
"EG"	Engage ECAM
“EM”	Specify ECAM cycle
"EP"	Specify ECAM table intervals & staring point
“EQ”	Disengage ECAM
"ET"	ECAM table

EXAMPLES:

EB1	Starts ECAM mode
EB0	Stops ECAM mode
B = _EB	Return status of cam mode

EC

FUNCTION: ECAM Counter

DESCRIPTION:

The EC function sets the index into the ECAM table. This command is only useful when entering ECAM table values without index values and is most useful when sending commands in binary. See the command, ET.

ARGUMENTS: EC n where

n is an integer between 0 and 256.

n = ? Returns the current value of the index into the ECAM table.

USAGE:

While Moving	Yes	Default Value
In a Program	Yes	Default Format
Command Line	Yes	
Controller Usage	**ALL CONTROLLERS, LOCAL AXES ONLY**	

DEFAULTS:

OPERAND USAGE:

_EC contains the current value of the index into the ECAM table.

RELATED COMMANDS:

"EA"	Choose ECAM master
"EB"	Enable ECAM
"EG"	Engage ECAM
“EM”	Specify ECAM cycle
"EP"	Specify ECAM table intervals & staring point
“EQ”	Disengage ECAM
"ET"	ECAM table

EXAMPLES:

EC0	Set ECAM index to 0
ET 200,400	Set first ECAM table entries to 200,400
ET 400,800	Set second ECAM table entries to 400,800

ED

FUNCTION: Edit

DESCRIPTION:

Using Galil DOS Terminal Software: The ED command puts the controller into the Edit subsystem. In the Edit subsystem, programs can be created, changed, or destroyed. The commands in the Edit subsystem are:

<cntrl>D	Deletes a line
<cntrl>I	Inserts a line before the current one
<cntrl>P	Displays the previous line
<cntrl>Q	Exits the Edit subsystem
<return>	Saves a line

Using Galil Windows Terminal Software: The ED command causes the Windows terminal software to open the terminal editor.

OPERAND USAGE:

_ED contains the line number of the last line to have an error.

EXAMPLES:

```
ED
000 #START
001 PR 2000
002 BGA
003 SLKJ                              Bad line
004 EN
005 #CMDERR                           Routine which occurs upon a command error
006 V=_ED
007 MG "An error has occurred" {n}
008 MG "In line", V{F3.0}
009 ST
010 ZS0
011 EN
```

Hint: *Remember to quit the Edit Mode prior to executing or listing a program.*

EG

FUNCTION: ECAM go (engage)

DESCRIPTION:

The EG command engages an ECAM slave axis at a specified position of the master. If a value is specified outside of the master's range, the slave will engage immediately. Once a slave motor is engaged, its position is redefined to fit within the cycle.

ARGUMENTS: EG n,n,n,n,n,n,n,n or EGA=n where

n is the ECAM master position at which the ECAM slave axis must be engaged.

n = ? Returns 1 if specified axis is engaged and 0 if disengaged.

USAGE:

		DEFAULTS:	
While Moving	Yes		Default Value
In a Program	Yes		Default Format
Command Line	Yes		
Controller Usage	**ALL CONTROLLERS, LOCAL AXES ONLY**		

OPERAND USAGE:

_EGa contains ECAM status for specified axis. 0 = axis is not engaged, 1 = axis is engaged.

RELATED COMMANDS:

"EA"	Choose ECAM master
"EB"	Enable ECAM
"EC"	Set ECAM table index
"EM"	Specify ECAM cycle
"EP"	Specify ECAM table intervals & staring point
"EQ"	Disengage ECAM
"ET"	ECAM table

EXAMPLES:

EG 700,1300	Engages the A and B axes at the master position 700 and 1300 respectively.
B = _EGB	Return the status of B axis, 1 if engaged

Note: *This command is not a trippoint. This command will not hold the execution of the program flow. If the execution needs to be held until master position is reached, use MF or MR command.*

ELSE

FUNCTION: Else function for use with IF conditional statement

DESCRIPTION:

The ELSE command is an optional part of an IF conditional statement. The ELSE command must occur after an IF command and it has no arguments. It allows for the execution of a command only when the argument of the IF command evaluates False. If the argument of the IF command evaluates false, the controller will skip commands until the ELSE command. If the argument for the IF command evaluates true, the controller will execute the commands between the IF and ELSE command.

ARGUMENTS: ELSE

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	No
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	
Default Format	

RELATED COMMANDS:

"ENDIF"	End of IF conditional Statement

EXAMPLES:

IF (@IN[1]=0)	IF conditional statement based on input 1
IF (@IN[2]=0)	2nd IF conditional statement executed if 1st IF conditional true
MG "INPUT 1 AND INPUT 2 ARE ACTIVE"	Message to be executed if 2nd IF conditional is true
ELSE	ELSE command for 2nd IF conditional statement
MG "ONLY INPUT 1 IS ACTIVE	Message to be executed if 2nd IF conditional is false
ENDIF	End of 2nd conditional statement
ELSE	ELSE command for 1st IF conditional statement
MG"ONLY INPUT 2 IS ACTIVE"	Message to be executed if 1st IF conditional statement
ENDIF	End of 1st conditional statement

EM

FUNCTION: Cam cycles

DESCRIPTION:

The EM command is part of the ECAM mode. It is used to define the change in position over one complete cycle of the master. The field for the master axis is the cycle of the master position. For the slaves, the field defines the net change in one cycle. If a slave will return to its original position at the end of the cycle, the change is zero. If the change is negative, specify the absolute value.

ARGUMENTS: EM n,n,n,n,n,n,n,n or EMX=n where

n is a positive integer in the range between 1 and 8,388,607 for the master axis and between 1 and 2,147,483,647 for a slave axis.

USAGE:		**DEFAULTS:**
While Moving	Yes	Default Value
In a Program	Yes	Default Format
Command Line	Yes	
Controller Usage	**ALL CONTROLLERS, LOCAL AXES ONLY**	

OPERAND USAGE:

_EMa contains the cycle of the specified axis.

RELATED COMMANDS:

"EA"	Choose ECAM master
"EB"	Enable ECAM
"EC"	Set ECAM table index
"EG"	Engage ECAM
"EP"	Specify ECAM table intervals & staring point
“EQ”	Disengage ECAM
"ET"	ECAM table

EXAMPLES:

EAC	Select C axis as master for ECAM.
EM 0,3000,2000	Define the changes in A and B to be 0 and 3000 respectively. Define master cycle as 2000.
V = _EMA	Return cycle of A

EN

FUNCTION: End

DESCRIPTION:

The EN command is used to designate the end of a program or subroutine. If a subroutine was called by the JS command, the EN command ends the subroutine and returns program flow to the point just after the JS command.

The EN command is used to end the automatic subroutines #MCTIME, #CMDERR, and #COMINT. When the EN command is used to terminate the #COMINT communications interrupt subroutine, there are two arguments; the first determines whether trippoints will be restored upon completion of the subroutine and the second determines whether the communication interrupt will be re-enabled.

ARGUMENTS: EN m, n where

m = 0: Return from subroutine without restoring trippoint

m = 1: Return from subroutine and restore trippoint

n = 0: Return from #COMINT without restoring interrupt

n = 1: Return from communications interrupt #COMINT and restore interrupt

Note1: The default values for the arguments are 0. For example EN,1 and EN0,1 have the same effect.

Note2: The arguments will specify how the #COMINT routine handles trippoints. Trippoints cause a program to wait for a particular event. The AM command, for example, waits for motion on all axes to complete. If the #COMINT subroutine is executed due to a communication interrupt while the program is waiting for a trippoint, the #COMINT can end and by continue to wait for the trippoint, or clear the trippoint and continue executing the program at the command just after the trippoint. .

Note3: Use the RE command to return from the interrupt handling subroutines #LIMSWI and #POSERR. Use the RI command to return from the #ININT subroutine.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	No
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	m=0, n=0
Default Format	

RELATED COMMANDS:

“RE”	Return from error subroutine
"RI"	Return from interrupt subroutine

EXAMPLES:

#A	Program A
PR 500	Move A axis forward 500 counts
BGA	Pause the program until the A axis completes the motion
AMA	Move A axis forward 1000 counts
PR 1000	Set another Position Relative move
BGA	Begin motion
EN	End of Program

ENDIF

FUNCTION: End of IF conditional statement

DESCRIPTION:

The ENDIF command is used to designate the end of an IF conditional statement. An IF conditional statement is formed by the combination of an IF and ENDIF command. An ENDIF command must always be executed for every IF command that has been executed. It is recommended that the user not include jump commands inside IF a conditional statement since this causes re-direction of command execution. In this case, the command interpreter may not execute an ENDIF command.

ARGUMENTS: ENDIF

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	No
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Yes	Default Value
Yes	Default Format

RELATED COMMANDS:

"IF"	Command to begin IF conditional statement
"ELSE"	Optional command to be used only after IF command
"JP"	Jump command
"JS"	Jump to subroutine command

EXAMPLES:

IF (@IN[1]=0)	IF conditional statement based on input 1
MG " INPUT 1 IS ACTIVE"	Message to be executed if "IF" conditional is false
ENDIF	End of conditional statement

Note: *Instead of EN, use the RE command to end the error subroutine and limit subroutine. Use the RI command to end the input interrupt) subroutine.*

EO

FUNCTION: Echo

DESCRIPTION:

The EO command turns the echo on or off. If the echo is off, characters input over the bus will not be echoed back.

ARGUMENTS: EO n where

n = 0 0 turns echo off

n = 1 1 turns echo on.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**SERIAL CONNECTIONS ONLY**

DEFAULTS:

Default Value	0
Default Format	1.0

EXAMPLES:

EO 0	Turns echo off
EO 1	Turns echo on

EP

FUNCTION: Cam table master interval and phase shift

DESCRIPTION:

The EP command defines the ECAM table master interval and phase shift. The interval m is the difference in master position between table entries. The phase shift n instantaneously moves the graph of slave position versus master position left (negative) or right (positive) and is used to make on-the-fly corrections to the slaves. Up to 257 points may be specified.

ARGUMENTS: EP m,n where

m is the master interval and is a positive integer in the range between 1 and 32,767 master counts. m cannot be changed while ECAM is running.

m = ? Returns the value of the interval, m.

n is the phase shift and is an integer between -2,147,483,648 and 2,147,483,647 master counts.. n can be changed while ECAM is running.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS, LOCAL AXES ONLY**

DEFAULTS:

Default Value	256,0
Default Format	

OPERAND USAGE:

_EP contains the value of the interval m.

RELATED COMMANDS:

"EA"	Choose ECAM master
"EB"	Enable ECAM
"EC"	Set ECAM table index
"EG"	Engage ECAM
“EM"	Specify ECAM cycle
“EQ”	Disengage ECAM
"ET"	ECAM table

EXAMPLES:

EP 20	Sets the cam master points to 0,20,40 . . .
D = _EP	Set the variable D equal to the ECAM internal master interval
EP, 100	Phase shift all slaves by 100 master counts

EQ

FUNCTION: ECAM quit (disengage)

DESCRIPTION:

The EQ command disengages an electronic cam slave axis at the specified master position. Separate points can be specified for each axis. If a value is specified outside of the master's range, the slave will disengage immediately.

ARGUMENTS: EQ n,n,n,n,n,n,n,n or EQA=n where

n is the master positions at which the axes are to be disengaged.

n = ? Returns 1 if engage command issued and axis is waiting to engage, 2 if disengage command issued and axis is waiting to disengage, and 0 if ECAM engaged or disengaged.

USAGE:

While Moving	Yes	Default Value
In a Program	Yes	Default Format
Command Line	Yes	
Controller Usage	**ALL CONTROLLERS, LOCAL AXES ONLY**	

DEFAULTS:

OPERAND USAGE:

_EQa contains 1 if engage command issued and axis is waiting to engage, 2 if disengage command issued and axis is waiting to disengage, and 0 if ECAM engaged or disengaged.

RELATED COMMANDS:

"EA"	Choose ECAM master
"EB"	Enable ECAM
"EC"	Set ECAM table index
"EG"	Engage ECAM
"EM"	Specify ECAM cycle
"EP"	Specify ECAM table intervals & staring point
"ET"	ECAM table

EXAMPLES:

EQ 300,700 Disengages the A and B motors at master positions 300 and 700 respectively.

Note: *This command is not a trippoint. This command will not hold the execution of the program flow. If the execution needs to be held until master position is reached, use MF or MR command.*

ER

FUNCTION: Error Limit

DESCRIPTION:

The ER command sets the magnitude of the position errors for each axis that will trigger an error condition. When the limit is exceeded, the Error output will go low (true). If the Off On Error (OE1) command is active, the motors will be disabled.

ARGUMENTS: ER n,n,n,n,n,n,n,n or ERA=n where

n is an unsigned numbers in the range 1 to 32767 which represents the error limit in encoder counts. A value of -1 will disable the position error limit for the specified axis.

n = ? Returns the value of the Error limit for the specified axis.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	16384
Default Format	Position Format

OPERAND USAGE:

_ERa contains the value of the Error limit for the specified axis.

RELATED COMMANDS:

"OE"	Off-On Error
#POSERR	Automatic Error Subroutine

EXAMPLES:

ER 200,300,400,600	Set the A-axis error limit to 200, the B-axis error limit to 300, the C-axis error limit to 400, and the D-axis error limit to 600.
ER ,1000	Sets the B-axis error limit to 1000, leave the A-axis error limit unchanged.
ER ?,?,?,?	Return A,B,C and D values
00200,00100,00400,00600	
ER ?	Return A value
00200	
V1=_ERA	Assigns V1 value of ERA
V1=	Returns V1
00200	

Hint: *The error limit specified by ER should be high enough as not to be reached during normal operation. Examples of exceeding the error limit would be a mechanical jam, or a fault in a system component such as encoder or amplifier.*

ES

FUNCTION: Ellipse Scale

DESCRIPTION:

The ES command divides the resolution of one of the axes in a vector mode (VM). This function allows for the generation of circular motion when encoder resolutions differ. It also allows for the generation of an ellipse instead of a circle.

The command has two parameters, m and n. The arguments, m and n apply to the axes designated by the command VM. When m>n, the resolution of the first axis, a, will be divided by the ratio m/n. When m<n, the resolution of the second axis, b, will be divided by n/m. The resolution change applies for the purpose of generating the VP and CR commands, effectively changing the axis with the higher resolution to match the coarser resolution.

ARGUMENTS: ES m,n where

m and n are positive integers in the range between 1 and 65,535.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS, LOCAL AXES ONLY**

DEFAULTS:

Default Value	1,1
Default Format	

RELATED COMMANDS:

"VM"	Vector Mode
"CR"	Circle move
"VP"	Vector position

EXAMPLES:

VMAB;ES3,4	Divide B resolution by 4/3

***Note**: ES must be issued after VM*

ET

FUNCTION: Electronic cam table

DESCRIPTION:

The ET command sets the ECAM table entries for the slave axes.. The values of the master axes are not required. The slave entry (n) is the position of the slave axes when the master is at the point (n ∗ i) + o, where i is the interval and o is the offset as determined by the EP command.

ARGUMENTS: ET[n] = n,n,n,n,n,n,n,n where

n is an integer in the range between -2,147,438,648, and 2,147,438,647.

The value n can be left out of the command if the index count has been set using the command, EC. In this mode, each ET command will automatically increment the index count by 1.

USAGE:

While Moving	Yes	Default Value
In a Program	Yes	Default Format
Command Line	Yes	
Controller Usage	**ALL CONTROLLERS, LOCAL AXES ONLY**	

DEFAULTS:

RELATED COMMANDS:

"EA"	Choose ECAM master
"EB"	Enable ECAM
"EC"	Set ECAM table index
"EG"	Engage ECAM
"EM"	Specify ECAM cycle
"EP"	Specify ECAM table intervals & staring point
"EQ"	Disengage ECAM

EXAMPLES:

ET[0]=0,,0	Specifies the position of the slave axes A and C to be synchronized with the starting point of the master.
ET[1]=1200,,400	Specifies the position of the slave axes A and C to be synchronized with the second point of the master
EC0	Set the table index value to 0, the first element in the table
ET 0,,0	Specifies the position of the slave axes A and C to be synchronized with the starting point of the master.
ET 1200,,400	Specifies the position of the slave axes A and C to be synchronized with the second point of the master

FA

FUNCTION: Acceleration Feed forward

DESCRIPTION:

The FA command sets the acceleration feed forward coefficient. This coefficient, when scaled by the acceleration, adds a torque bias voltage during the acceleration phase and subtracts the bias during the deceleration phase of a motion.

Acceleration Feed forward Bias = $FA \cdot AC \cdot 1.5 \cdot 10^{-7}$

Deceleration Feed forward Bias = $FA \cdot DC \cdot 1.5 \cdot 10^{-7}$

The Feed forward Bias product is limited to 10 Volts. FA operates when commanding motion with PA, PR and JG.

ARGUMENTS: FA n,n,n,n,n,n,n,n or FAA=n where

n is an unsigned number in the range 0 to 8191 decimal with a resolution of 0.25.

n = ? Returns the value of the feed forward acceleration coefficient for the specified axis.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	0
Default Format	4.0

OPERAND USAGE:

_FAa contains the value of the feed forward acceleration coefficient for the specified axis.

RELATED COMMANDS:

"FV"	Velocity feed forward

EXAMPLES:

AC 500000,1000000	Set feed forward coefficient to 10 for the A-axis
FA 10,15	and 15 for the B-axis. The effective bias will be 0.75V for A and 2.25V for B.
FA ?,?	Return A and B values
010,015	

Note: *If the feed forward coefficient is changed during a move, then the change will not take effect until the next move.*

FE

FUNCTION: Find Edge

DESCRIPTION:

The FE command moves a motor until a transition is seen on the homing input for that axis. The direction of motion depends on the initial state of the homing input (use the CN command to configure the polarity of the home input). Once the transition is detected, the motor decelerates to a stop.

This command is useful for creating your own homing sequences.

ARGUMENTS: FE nnnnnnnn where

n is A,B,C,D,E,F,G or H or any combination to specify the axis or axes

No argument specifies all axes.

USAGE:		**DEFAULTS:**	
While Moving	No	Default Value	
In a Program	Yes	Default Format	
Command Line	Yes		
Controller Usage	**ALL CONTROLLERS**		

RELATED COMMANDS:

"FI"	Find Index
"HM"	Home
"BG"	Begin
"AC"	Acceleration Rate
"DC"	Deceleration Rate
"SP"	Speed for search

EXAMPLES:

FE	Set find edge mode
BG	Begin all axes
FEA	Only find edge on A
BGA	
FEB	Only find edge on B
BGB	
FECD	Find edge on C and D
BGCD	

Hint: *Find Edge only searches for a change in state on the Home Input. Use FI (Find Index) to search for the encoder index. Use HM (Home) to search for both the Home input and the Index. Remember to specify BG after each of these commands.*

FI

FUNCTION: Find Index

DESCRIPTION:

The FI and BG commands move the motor until an encoder index pulse is detected. The controller looks for a transition from low to high. When the transition is detected, motion stops and the position is defined as zero. To improve accuracy, the speed during the search should be specified as 500 counts/s or less. The FI command is useful in custom homing sequences. The sign of the JG command specifies the direction of motion.

ARGUMENTS: FI nnnnnnnn where

n is A,B,C,D,E,F,G or H or any combination to specify the axis or sequence

No argument specifies all axes.

USAGE:	**DEFAULTS:**	
While Moving	No	Default Value
In a Program	Yes	Default Format
Command Line	Yes	
Controller Usage	**ALL CONTROLLERS**	

RELATED COMMANDS:

"FE"	Find Edge
"HM"	Home
"BG"	Begin
"AC"	Acceleration Rate
"SP"	Search Speed

EXAMPLES:

#HOME	Home Routine
JG 500	Set speed and forward direction
FIA	Find index
BGA	Begin motion
AMA	After motion
MG "FOUND INDEX"	

Hint: *Find Index only searches for a change in state on the Index. Use FE to search for the Home. Use HM (Home) to search for both the Home input and the Index. Remember to specify BG after each of these commands.*

FL

FUNCTION: Forward Software Limit

DESCRIPTION:

The FL command sets the forward software position limit. If this limit is exceeded during motion, motion on that axis will decelerate to a stop. Forward motion beyond this limit is not permitted. The forward limit is activated at A+1, B+1, C+1, D+1. The forward limit is disabled at 2147483647. The units are in counts.

When the reverse software limit is activated, the automatic subroutine #LIMSWI will be executed if it is included in the program and a program is executing. See User's Manual, Automatic Subroutine.

ARGUMENTS: FL n,n,n,n,n,n,n,n or FLA=n where

n is a signed integers in the range -2147483648 to 2147483647, n represents the absolute position of axis.

n = 2147483647 turns off the forward limit

n = ? Returns the value of the forward limit switch for the specified axis.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	2147483647
Default Format	Position Format

OPERAND USAGE:

_FLa contains the value of the forward software limit for the specified axis.

RELATED COMMANDS:

"BL"	Reverse Limit

EXAMPLES:

FL 150000	Set forward limit to 150000 counts on the A-axis
#TEST	Test Program
AC 1000000	Acceleration Rate
DC 1000000	Deceleration Rate
FL 15000	Forward Limit
JG 5000	Jog Forward
BGA	Begin
AMA	After Limit
TPA	Tell Position
EN	End

Hint: *Galil controllers also provide hardware limits.*

@FRAC[n]

FUNCTION: Fractional part

DESCRIPTION:

Returns the fractional part of the given number

ARGUMENTS: @FRAC[n]

n is a signed number in the range -2147483648 to 2147483647.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

@INT	Integer part

EXAMPLES:

```
:MG @FRAC[1.2]
 0.2000
:MG @FRAC[-2.4]
 -0.4000
:
```

FV

FUNCTION: Velocity Feed forward

DESCRIPTION:

The FV command sets the velocity feed forward coefficient, or returns the previously set value. This coefficient generates an output bias signal in proportions to the commanded velocity.

Velocity feed forward bias = $1.22 \cdot 10^{-6} \cdot FV \cdot$ Velocity [in cts/s].

FV operates when commanding motion with PA, PR, JG, VM, LM, and CM.

For example, if FV=10 and the velocity is 200,000 count/s, the velocity feedf orward bias equals 2.44 volts.

ARGUMENTS: FV n,n,n,n,n,n,n,n or FVA=n where

n is an unsigned numbers in the range 0 to 8191 decimal

n = ? Returns the feed forward velocity for the specified axis.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	0
Default Format	3.0

OPERAND USAGE:

_FVa contains the feed forward velocity for the specified axis.

RELATED COMMANDS:

"FA"	Acceleration feed forward

EXAMPLES:

FV 10,20	Set feed forward coefficients to 10 and 20 for A
JG 30000,80000	and B respectively. This produces 0.366 volts for A and 1.95 volts for B.
FV ?,?	Return the A and B values.
010,020	

GA

FUNCTION: Master Axis for Gearing

DESCRIPTION:

The GA command specifies the master axes for electronic gearing. Multiple masters for gearing may be specified. The masters may be the main encoder input, auxiliary encoder input, or the commanded position of any axis. The master may also be the commanded vector move in a coordinated motion of LM or VM type. When the master is a simple axis, it may move in any direction and the slave follows. When the master is a commanded vector move, the vector move is considered positive and the slave will move forward if the gear ratio is positive, and backward if the gear ratio is negative. The slave axes and ratios are specified with the GR command and gearing is turned off by the command GR0.

ARGUMENTS: GA x,x,x,x,x,x,x,x or GAA=x where

x can be A,B,C,D,E,F,G or H. The value of x is used to set the specified encoder axis as the gearing master. The slave axis is specified by the position of the argument. The first position of the argument corresponds to the 'A' axis, the second position corresponds to the 'B' axis, etc. A comma must be used in place of an argument if the corresponding axes will not be a slave.

x can be CA,CB,CC,CD,CE,CF,CG or CH. The value of x is used to set the commanded position of the specified axis as the gearing master.

x can be S to specify the vector motion of the coordinated system as the gearing master.

x=? returns the current GA setting

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS, LOCAL AXES ONLY**

DEFAULTS:

Default Value	
Default Format	

RELATED COMMANDS:

"GR"	Gear Ratio
"GM"	Gantry Mode

EXAMPLES:

#GEAR	Gear program
GA ,A,S	Specify A axis as master for B and vector motion on S as master for C
GR ,.5,-2.5	Specify B and C ratios
JG 5000	Specify master jog speed
BGA	Begin motion
WT 10000	Wait 10000 msec
STA	Stop

Hint: *Using the command position, as the master axis is useful for gantry applications. Using the vector motion as master is useful in generating helical motion.*

GM

FUNCTION: Gantry mode

DESCRIPTION:

The GM command specifies the axes in which the gearing function is performed in the Gantry mode. In this mode, the gearing will not stop by the ST command or by limit switches. Only GR0 will stop the gearing in this mode.

ARGUMENTS: GM n,n,n,n,n,n,n,n or GMA=n where

n = 0 Disables gantry mode function

n = 1 Enables the gantry mode

n = ? Returns the state of gantry mode for the specified axis: 0 gantry mode disabled, 1 gantry mode enabled

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS, LOCAL AXES ONLY**

DEFAULTS:

Default Value	0
Default Format	1.0

OPERAND USAGE:

_GMa contains the state of gantry mode for the specified axis: 0 gantry mode disabled, 1 gantry mode enabled

RELATED COMMANDS:

"GR"	Gear Ratio
"GA"	Master Axis for Gearing

EXAMPLES:

GM 1,1,1,1	Enable GM on all axes
GM 0	Disable GM on A-axis other axes remain unchanged
GM ,,1,1	Enable GM on C-axis and D-axis other axes remain unchanged
GM 1,0,1,0	Enable GM on A and C-axis Disable GM on B and D axis

Hint: *The GM command is useful for driving heavy load on both sides (Gantry Style).*

GR

FUNCTION: Gear Ratio

DESCRIPTION:

GR specifies the Gear Ratios for the geared axes in the electronic gearing mode. The master axis is defined by the GA command. The gear ratio may be different for each geared axis. The master can go in both directions. A gear ratio of ø disables gearing for each axis. A limit switch also disables the gearing unless gantry mode has been enabled (see GM command).

ARGUMENTS: GR n,n,n,n,n,n,n,n or GRA=n where

n is a signed numbers in the range +/-127, with a fractional resolution of $\frac{1}{2}^{16}$.

n = 0 Disables gearing

n = ? Returns the value of the gear ratio for the specified axis.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS, LOCAL AXES ONLY**

DEFAULTS:

Default Value	0
Default Format	3.4

OPERAND USAGE:

_GRa contains the value of the gear ratio for the specified axis.

RELATED COMMANDS:

"GA"	Master Axis
"GM"	Gantry Mode

EXAMPLES:

#GEAR	
MOB	Turn off servo to B motor
GAB	Specify master axis as B
GR .25,,-5	Specify A and C gear ratios
EN	End program

Now when the B motor is rotated by hand, the A will rotate at 1/4th the speed and C will rotate 5 times the speed in the opposite direction.

***Hint**: when the geared motors must be coupled "strongly" to the master, use the gantry mode GM.*

HC

FUNCTION: Handle Configuration

DESCRIPTION:

The HC command configures and establishes communications for a master/slave system. The command is executed in the master controller and addresses all slaves and IOC modules in the system. After the HC command is initiated, the master responds to the slave and IOC BOOTP requests and assigns corresponding IP addresses. The master then opens handles and initiates the slave update packets (QW).

The IP address for the master controller must be established with the IA command or DMCNet software prior to the HC command being issued. If no IP addresses have been issued for the slaves or IOC's, the master will assign addresses to these controllers as it receives the bootp packets. This is the recommended method. If IP addresses are to be assigned to the slaves manually, these must be assigned based on the master IP address. See Chapter 2 "Step 10: Configure the Distributed Control System" for manual address selection information.

Jumpers on the slave and IOC modules must be set to indicate axes configurations and IOC number. See Chapter 2 "Step 2: Configuring Jumpers on the DMC-3425" for more information.

ARGUMENTS: HCa,b,c,d where

a is the total number of axes in the system

b is the slave update interval (QW) in milliseconds

c is the communication protocol for the slave communications

1 = UDP (1 handle used)

2 = TCP/IP (2 handles used)

3 = TCP/IP used for Command Handle, UDP used for QW update Handle

d is the total number of IOC-7007 modules in the system

HC? returns the present setting of the HC command

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	2,20,1,0
Default Format	--

OPERAND USAGE:

_HC contains a 1 if the handle configuration is in progress

contains a 2 if the handle configuration has completed successfully

contains a 0 if the handle configuration failed or has not been issued

RELATED COMMANDS:

"CH"	Configure handles
"IA"	Internet Address
"IH"	Internet Handle

"NA"	Number of axes
"QW"	Slave data records

EXAMPLES:

IA 151, 12, 53, 89	Assigns the controller with the address 151.12.53.89
HC4,20,2,0	Configures a 4 axis system with two TCP/IP handles per slave. The data update interval is set to 20 milliseconds. For each slave, one TCP/IP handle will be used for sending commands while the other TCP/IP handle will be used for the data update.
HC6,30,1,0	Configures a 6 axis system with a single UDP handle per slave at updates of 30 msec. The single UDP handle is used for both sending commands and receiving data packets.

#AUTO HC 3, 250, 2	Example program that will automatically run when controller is powered up (#AUTO). HC command configures a 3 axis system with a 250 msec update rate.
#Loop;JP#Loop,_HC<>2 MG "Connected" EN	# Loop routine causes controller to wait for successful connection before continuing execution of code.

Hint: Use a WT (Wait) or #Loop; JP#Loop,_HC<>2 when issuing the HC command in a program to allow enough time for slaves to be configured correctly before executing any other commands.

HM

FUNCTION: Home

DESCRIPTION:

The HM command performs a three-stage homing sequence for servo systems and two stage sequence for stepper motor operation.

During first stage of the motor moves at the user programmed speed until detecting a transition on the homing input for that axis. The direction for this first stage is determined by the initial state of the Homing Input. Once the homing input changes state, the motor decelerates to a stop. The state of the homing input can be configured using the CN command.

At the second stage, the motor change directions and slowly approaches the transition again. When the transition is detected, the motor is stopped instantaneously..

At the third stage, the motor slowly moves forward until it detects an index pulse from the encoder. It stops at this point and defines it as position 0.

For stepper mode operation, the sequence consists of the first two stages. The frequency of the motion in stage 2 is 256 cts./sec.

USAGE:

While Moving	No
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	
Default Format	

OPERAND USAGE:

_HMx contains the state of the home switch for the specified axis

RELATED COMMANDS:

"CN"	Configure Home
"FI"	Find Index Only
"FE"	Find Home Only

EXAMPLES:

HM	Set Homing Mode for all axes
BG	Home all axes
BGA	Home only the A-axis
BGB	Home only the B-axis
BGC	Home only the C-axis
BGD	Home only the D-axis

Hint: *You can create your own custom homing sequence by using the FE (Find Home Sensor only) and FI (Find Index only) commands.*

HR

FUNCTION: Handle Restore

DESCRIPTION:

The HR command is used to enable the automatic restoration of handles that have closed during distributed control communications. Once enabled with the HC command, handles that have been assigned as data or communication channels for distributed control are monitored in the master. If a handle closes, attempts are made to re-establish connection and restore communications with the handle. This command is executed in the master controller as it controls the handle assignments and monitoring of those handles.

Note: This command only valid when the automatic setup method, HC, has been used.

ARGUMENTS: HRn where

n = 0 to disable automatic handle restore

n = 1 to enable automatic handle restore

HR? returns the present setting of the handle restore command.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	n = 0
Default Format	--

RELATED COMMANDS:

"HC"	Automatic Handle Connect
"IH"	Internet Handle

EXAMPLES:

HR1	Enable automatic handle restore
HR?	Request handle restore setting
: 1	Handle restore is enabled

HS

FUNCTION: Handle Assignment Switch

DESCRIPTION:

The HS command is used to switch the handle assignments between two handles. The controller assigns handles when the handles are opened with the HC command, or are assigned explicitly with the IH command. Should those assignments need modifications, the HS command allows the handles to be reassigned.

Note: This command only valid when the automatic setup method, HC, has been used.

ARGUMENTS: HSh=i where

h is the first handle of the switch (A through H, S)

i is the second handle of the switch (A through H, S)

S is used to represent the current handle executing the command

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	--
Default Format	--

RELATED COMMANDS:

"HC"	Automatic Handle Connect
"HR"	Handle Restore
"IH"	Internet Handle

EXAMPLES:

HSC=D	Connection for handle C is assigned to handle D. Connection for handle D is assigned to handle C.
HSS=E	Executing handle connection is assigned to handle E. Connection for handle E is assigned to executing handle.

HW

FUNCTION: Handle response wait

DESCRIPTION:

This command is used to set the master to wait on responses from slave for each command sent. With this command enabled, the master controller will wait until the slave responds to a command before sending the colon or ? to the host. If an error is generated on the slaves, the master will indicate the error with a ? to the host. With this command disabled, the master controller will immediately acknowledge a command sent to the slave and will not wait for the slave to respond to the command.

If an error is generated on a slave while in the HW1 mode, the master will respond with a "?". Issuing the TC on the master will respond with the error code from the slave.

Issuing TCA through TCH will respond with the text of the error from the slave on a specified handle.

_TCA through _TCH will respond with the error code from the slave on a specified handle.

ARGUMENTS: HWn where

n = 0 turns handle response wait off

n = 1 turns handle response wait on

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	1
Default Format	--

OPERAND USAGE:

_HW contains the current value of the handle response wait parameter.

EXAMPLES:

HW1	Turn on handle response wait
HW0	Turn off handle response wait

**Note: HW command is only valid when HC is used to set up master-slave communication.*

HX

FUNCTION: Halt Execution

DESCRIPTION:

The HX command halts the execution of any program that is running.

ARGUMENTS: HXn where

n is an integer in the range of 0 to 1 and indicates the thread number.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS WITH DIRECT CONNECTION**

DEFAULTS:

Default Value	n = 0
Default Format	

OPERAND USAGE:

When used as an operand, _HXn contains the running status of thread n with:

0	Thread not running
1	Thread is running
2	Thread has stopped at trippoint

RELATED COMMANDS:

"XQ"	Execute program
"ST"	Stop all threads of motion

EXAMPLES:

XQ #A	Execute program #A, thread zero
XQ #B,1	Execute program #B, thread one
HX0	Halt thread zero
HX1	Halt thread one

IA

FUNCTION: IP Address

DESCRIPTION:

The IA command assigns the controller with an IP address.

The IA command may also be used to specify the time out value. This is only applicable when using the TCP/IP protocol.

The IA command can only be used via RS-232. Since it assigns an IP address to the controller, communication with the controller via internet cannot be accomplished until after the address has been assigned.

ARGUMENTS: IA ip0,ip1,ip2, ip3 **or** IA n **or** IA<t>u where

ip0, ip1, ip2, ip3 are 1 byte numbers separated by commas and represent the individual fields of the IP address.

n is the IP address for the controller which is specified as an integer representing the signed 32 bit number (two's complement).

<t specifies the time in update samples between TCP retries.

>u specifies the multicast IP address where u is an integer between 0 and 63.

IA? will return the IP address of the controller

USAGE:

While Moving	No
In a Program	No
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	n = 0, t=250, u=0
Default Format	--

OPERAND USAGE:

_IA0	contains the IP address representing a 32 bit signed number (Two's complement)
_IA1	contains the value for t (retry time)
_IA2	contains the number of available handles
_IA3	contains the number of the handle using this operand where the number is 0 to 7. 0 represents handle A, 1 handle B, etc.
_IA4	contains the handle that lost communication last. Contains –1 on reset to indicate no handles lost.

RELATED COMMANDS:

"IH"	Internet Handle

EXAMPLES:

IA 151, 12, 53, 89	Assigns the controller with the address 151.12.53.89
IA 2534159705	Assigns the controller with the address 151.12.53.89
IA < 500	Sets the timeout value to 500msec

IF

FUNCTION: IF conditional statement

DESCRIPTION:

The IF command is used in conjunction with an ENDIF command to form an IF conditional statement. The arguments are one or more conditional statements. If the conditional statement(s) evaluates true, the command interpreter will continue executing commands which follow the IF command. If the conditional statement evaluates false, the controller will ignore commands until the associated ENDIF command OR an ELSE command occurs in the program.

ARGUMENTS: IF condition where

Conditions are tested with the following logical operators:

< less than or equal to

> greater than

= equal to

<= less than or equal to

>= greater than or equal to

<> not equal

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	No
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

"ELSE"	Optional command to be used only after IF command
"ENDIF"	End of IF conditional Statement

EXAMPLES:

IF (_TEA<1000)	IF conditional statement based on A motor position
MG "Motor is within 1000 counts of zero"	Message to be executed if “IF” conditional statement
ENDIF	End of IF conditional statement

IH

FUNCTION: Open Internet Handle

DESCRIPTION:

The IH command establishes a communication channel between a master device and a slave device. Each communication channel is known as a handle, and the master controller opens a handle by identifying the desired handle number and the IP address of the slave.

Each controller may have 8 handles open at any given time. They are designated by the letters A through H. To open a handle, the user must specify:

1. The IP address of the slave
2. The type of session: TCP/IP or UDP/IP
3. The port number of the slave. This number is not necessary if the slave device does not require a specific port value. If not specified, the controller will specify the port value as 1000.

This command is not necessary when using the automatic setup procedure HC.

ARGUMENTS: IHh= ip0,ip1,ip2,ip3 <p >q **or** IHh=n <p >q **or** IHh= >r where

h is the handle, specified as A,B,C,D,E,F,G or H

ip0,ip1,ip2,ip3 are integers between 0 and 255 and represent the individual fields of the IP address of the slaves. Commas must separate these values.

n is a signed integer between - 2147483648 and 2147483647. This value is the 32 bit IP address and can be used instead of specifying the 4 address fields.

IHS => C closes the handle that sent the command where C=-1 for UDP/IP and C=-2 for TCP/IP.

IHN => C closes all handles except for the one sending the command where C=–1 for UDP and C=–2 for TCP/IP.

<p specifies the port number of the slave where p is an integer between 0 and 65535. This value is not required for opening a handle.

>q specifies the connection type where q is 0 for no connection, 1 for UDP and 2 for TCP

>r specifies that the connection be terminated and the handle be freed, where r is –1 for UDP close, –2 for standard TCP/IP close and –3 for TCP/IP close and TCP/IP reset

IHh= will restore the previously closed handle

"?" returns the IP address as 4 1-byte numbers

OPERAND USAGE:

_IHh0 contains the IP address as a 32 bit number

_IHh1 contains the slave port number

_IHh2 contains a 0 if the handle is free

contains a 1 if it is for a UDP slave

contains a 2 if it is for a TCP slave

contains a –1 if it is for a UDP master

contains a –2 if it is for a TCP master

contains a –5 while attempting to establish a UDP handle

contains a –6 while attempting to establish a TCP/IP handle

_IHh3 contains a 0 if the ARP was successful or has timed out

contains a 1 if it is still in progress.

_IHh4 contains a 1 if the master controller is waiting for acknowledgment from the slave after issuing a command.

contains a 2 if the master controller received a colon from the slave after issuing a command.

contains a 3 if the master controller received a question mark from the slave after issuing a command.

contains a 4 if the master controller timed-out while waiting for a response from the slave after issuing a command.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	--
Default Format	--

RELATED COMMANDS:

“CH”	Configure Handle
"IA"	Internet Address

EXAMPLES:

IHA=251,29,51,1	Open handle A at IP address 251.29.51.1
IHA= -2095238399	Open handle A at IP address 251.29.51.1

Note: When the IH command is given, the controller initializes an ARP on the slave device before opening a handle. This operation can cause a small time delay before the controller responds.

II

FUNCTION: Input Interrupt

DESCRIPTION:

The II command enables the interrupt function for the specified inputs. By default, input interrupts are configured for activation with a logic "0" but can be configured for activation with a logic "1" signal.

If any of the specified inputs are activated during program execution, the program will jump to the subroutine with label #ININT. Any trippoints set by the program will be cleared but can be re-enabled by the proper termination of the interrupt subroutine using RI. The RI command is used to return from the #ININT routine.

Note: *An application program must be running on the controller for the interrupt function to work.*

ARGUMENTS: II m,n,o,p where

m is an integer between 0 and 7 decimal. 0 disables interrupt. The value of m specifies the lowest input to be used for the input interrupt. When the 2nd argument, n, is omitted, only the input specified by m will be enabled.

n is an integer between 2 and 7. This argument is optional and is used with m to specify a range of values for input interrupts. For example, II 2,4 specifies interrupts occurring for Input 2, Input 3 and Input 4.

o is an integer between 1 and 127. Using this argument is an alternative to specifying an input range with m,n. If m and n are specified, o will be ignored. The argument o is an integer value and represents a binary number. For example, if o = 15, the binary equivalent is 000 1111 where the bottom 4 bits are 1 (bit 0 through bit 3) and the top 3 bits are 0 (bit 4 through bit 6). Each bit represents an interrupt to be enabled - bit0 for interrupt 1, bit 1 for interrupt 2, etc. If o=15, the inputs 1,2,3 and 4 would be enabled.

p is an integer between 1 and 127. The argument p is used to specify inputs that will be activated with a logic "1". This argument is an integer value and represents a binary number. This binary number is used to logically "AND" with the inputs which have been specified by the parameters m and n or the parameter o. For example, if m=1 and n=4, the inputs 1,2,3 and 4 have been activated. If the value for p is 2 (the binary equivalent of 2 is 000 0010), input 2 will be activated by a logic '1' and inputs 1,3, and 4 will be activated with a logic "0".

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	No
Controller Usage	**ALL CONTROLLERS, LOCAL AXES ONLY**

DEFAULTS:

Default Value	
Default Format	3.0 (mask only)

RELATED COMMANDS:

"AI"	Trippoint for input
"RI"	Return from Interrupt
#ININT	Interrupt Subroutine

EXAMPLES:

#A	Program A
II 1	Specify interrupt on input 1
JG 5000;BGA	Specify jog and begin motion on A axis
#LOOP;JP #LOOP	Loop
EN	End Program
#ININT	Interrupt subroutine
STA;MG "INTERRUPT";AMA	Stop A, print message, wait for motion to complete
#CLEAR;JP#CLEAR,@IN[1]=0	Check for interrupt clear
BGA	Begin motion
RI0	Return to main program, don't re-enable trippoints

IL

FUNCTION: Integrator Limit

DESCRIPTION:

The IL command limits the effect of the integrator function in the filter to a certain voltage. For example, IL 2 limits the output of the integrator of the A-axis to the +/-2 Volt range.

A negative parameter also freezes the effect of the integrator during the move. For example, IL -3 limits the integrator output to +/-3V. If, at the start of the motion, the integrator output is 1.6 Volts, that level will be maintained through the move. Note, however, that the KD and KP terms remain active in any case.

ARGUMENTS: IL n,n,n,n,n,n,n,n or ILA=n where

n is a number in the range -10 to 10 Volts with a resolution of 0.0003.

n = ? Returns the value of the integrator limit for the specified axis.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	9.9982
Default Format	1.4

OPERAND USAGE:

_ILa contains the value of the integrator limit for the specified axis.

RELATED COMMANDS:

"KI"	Integrator

EXAMPLES:

KI 2,3,5,8	Integrator constants
IL 3,2,7,2	Integrator limits
IL ?	Returns the A-axis limit
3.0000	

@IN[n]

FUNCTION: Read digital input

DESCRIPTION:

Returns the value of the given digital input (either 0 or 1)

ARGUMENTS: @IN[n] where

n is an unsigned integer in the range 1 to 96

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

@AN	Read analog input
@OUT	Read digital output
SB	Set digital output bit
CB	Clear digital output bit
OP	Output port

EXAMPLES:

```
:MG @IN[1] ;'print digital input 1
 1.0000
  :x = @IN[1] ;'assign digital input 1 to a variable
```

#ININT

FUNCTION: Input interrupt automatic subroutine

DESCRIPTION:

#ININT runs upon a state transition of digital inputs 1 to 8 and is configured with II. #ININT runs in thread 0 and requires something running in thread 0 to be active.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	No
Controller Usage	**ALL**

RELATED COMMANDS:

II	Input interrupt
@IN	Read digital input
RI	Return from interrupt

EXAMPLES:

```
II1           ;'arm digital input 1

#MAIN         ;'print a message every second
  MG "MAIN"
  WT1000
JP #MAIN

#ININT        ;'runs when input 1 goes low
  MG "ININT"
  AI1
RI
```

***NOTE**: An application program must be executing for the automatic subroutine to function, which runs in thread 0.*

***NOTE**: Use RI to end the routine*

@INT[n]

FUNCTION: Integer part

DESCRIPTION:

Returns the integer part of the given number. Note that the modulus operator can be implemented with @INT (see example below).

ARGUMENTS: @INT[n]

n is a signed number in the range -2147483648 to 2147483647.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

@FRAC	Fractional part

EXAMPLES:

```
:MG @INT[1.2]
 1.0000
:MG @INT[-2.4]
 -2.0000
:
```

```
#AUTO    ;'modulus example
  x = 10 ;'prepare arguments
  y = 3
  JS#mod ;'call modulus
  MG z   ;'print return value
EN


'subroutine:  integer remainder of x/y (10 mod 3 = 1)
'arguments are x and y.  Return is in z
#mod
  z = x - (y * @INT[x/y])
EN
```

IP

FUNCTION: Increment Position

DESCRIPTION:

The IP command allows for a change in the command position while the motor is moving. This command does not require a BG. The command has three effects depending on the motion being executed. The units of this are quadrature.

Case 1: Motor is standing still

An IP a,b,c,d command is equivalent to a PR a,b,c,d and BG command. The motor will move to the specified position at the requested slew speed and acceleration.

Case 2: Motor is moving towards a position as specified by PR, PA, or IP.

An IP command will cause the motor to move to a new position target, which is the old target plus the specified increment. The incremental position must be in the same direction as the existing motion.

Case 3: Motor is in the Jog Mode

An IP command will cause the motor to instantly try to servo to a position that is the current instantaneous position plus the specified increment position. The SP and AC parameters have no effect. This command is useful when synchronizing 2 axes in which one of the axis' speeds is indeterminate due to a variable diameter pulley.

Warning: When the mode is in jog mode, an IP will create an instantaneous position error. In this mode, the IP should only be used to make small incremental position movements.

ARGUMENTS: IP n,n,n,n,n,n,n,n or IPA=n where

n is a signed numbers in the range -2147483648 to 2147483647 decimal.

n = ? Returns the current position of the specified axis.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	
Default Format	7.0

RELATED COMMANDS:

"PR"	Position Relative
“PA”	Position Absolute

EXAMPLES:

IP 50	50 counts with set acceleration and speed
#CORRECT	Label
AC 100000	Set acceleration
JG 10000;BGA	Jog at 10000 counts/sec rate
WT 1000	Wait 1000 msec
IP 10	Move the motor 10 counts instantaneously
STA	Stop Motion

IT

FUNCTION: Independent Time Constant - Smoothing Function

DESCRIPTION:

The IT command filters the acceleration and deceleration functions of independent moves such as JG, PR, PA to produce a smooth velocity profile. The resulting profile, known as smoothing, has continuous acceleration and results in reduced mechanical vibrations. IT sets the bandwidth of the filter where 1 means no filtering and 0.004 means maximum filtering. Note that the filtering results in longer motion time.

The use of IT will not affect the trippoints AR and AD. The trippoints AR & AD monitor the profile prior to the IT filter and therefore can be satisfied before the actual distance has been reached if IT is NOT 1.

ARGUMENTS: IT n,n,n,n,n,n,n,n or ITA=n where

n is a positive numbers in the range between 0.004 and 1.0 with a resolution of 1/256.

n = ? Returns the value of the independent time constant for the specified axis.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	1
Default Format	7.0

OPERAND USAGE:

_ITx contains the value of the independent time constant for the specified ‘x’ axis.

RELATED COMMANDS:

"VT"	Vector Time Constant for smoothing vector moves

EXAMPLES:

IT 0.8, 0.6, 0.9, 0.1	Set independent time constants for a,b,c,d axes
IT ?	Return independent time constant for A-axis
0.8	

JG

FUNCTION: Jog

DESCRIPTION:

The JG command sets the jog mode and the jog slew speed of the axes.

ARGUMENTS: JG n,n,n,n,n,n,n,n or JGA=n where

n is a signed even integer in the range 0 to +/-12,000,000 decimal. The units of this are counts/second. (Use JGN=n for virtual axis)

For stepper motor operation, the maximum value is 3,000,000 steps/ second

n = ? Returns the absolute value of the jog speed for the specified axis.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	25000
Default Format	Position Format

OPERAND USAGE:

_JGa contains the absolute value of the jog speed for the specified axis.

RELATED COMMANDS:

"BG"	Begin
"ST"	Stop
"AC"	Acceleration
"DC"	Deceleration
"IP"	Increment Position
"TV"	Tell Velocity

EXAMPLES:

JG 100,500,2000,5000	Set for jog mode with a slew speed of 100 counts/sec for the A-axis, 500 counts/sec for the B-axis, 2000 counts/sec for the C-axis, and 5000 counts/sec for D-axis.
BG	Begin Motion
JG ,,-2000	Change the C-axis to slew in the negative direction at -2000 counts/sec.

***Note**: JG2 is the minimum non-zero speed.*

JP

FUNCTION: Jump to Program Location

DESCRIPTION:

The JP command causes a jump to a program location on a specified condition. The program location may be any program line number or label. The condition is a conditional statement which uses a logical operator such as equal to or less than. A jump is taken if the specified condition is true.

Multiple conditions can be used in a single jump statement. The conditional statements are combined in pairs using the operands "&" and "|". The "&" operand between any two conditions, requires that both statements must be true for the combined statement to be true. The "|" operand between any two conditions, requires that only one statement be true for the combined statement to be true. *Note: Each condition must be placed in parenthesis for proper evaluation by the controller.*

ARGUMENTS: JP location,condition where

location is a program line number or label

condition is a conditional statement using a logical operator

The logical operators are:

< less than

> greater than

= equal to

<= less than or equal to

>= greater than or equal to

<> not equal to

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	No
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	
Default Format	

RELATED COMMANDS:

"JS"	Jump to Subroutine
"IF"	If conditional statement
"ELSE"	Else function for use with IF conditional statement
"ENDIF"	End of IF conditional statement

EXAMPLES:

JP #POS1,V1<5	Jump to label #POS1 if variable V1 is less than 5
JP #A,V7*V8=0	Jump to #A if V7 times V8 equals 0
JP #B	Jump to #B (no condition)

Hint: *JP is similar to an IF, THEN command. Text to the right of the comma is the condition that must be met for a jump to occur. The destination is the specified label before the comma.*

JS

FUNCTION: Jump to Subroutine

DESCRIPTION:

The JS command will change the sequential order of execution of commands in a program. If the jump is taken, program execution will continue at the line specified by the destination parameter, which can be either a line number or label. The line number of the JS command is saved and after the next EN command is encountered (End of subroutine), program execution will continue with the instruction following the JS command. There can be a JS command within a subroutine.

Multiple conditions can be used in a single jump statement. The conditional statements are combined in pairs using the operands "&" and "|". The "&" operand between any two conditions, requires that both statements must be true for the combined statement to be true. The "|" operand between any two conditions, requires that only one statement be true for the combined statement to be true. *Note: Each condition must be placed in parenthesis for proper evaluation by the controller.*

Note: Subroutines may be nested 16 deep in the controller.

A jump is taken if the specified condition is true. Conditions are tested with logical operators. The logical operators are:

< less than or equal to	<= less than or equal to
> greater than	>= greater than or equal to
= equal to	<> not equal

ARGUMENTS: JS destination, condition where

destination is a line number or label

condition is a conditional statement using a logical operator

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	No
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	
Default Format	

RELATED COMMANDS:

"EN"	End

EXAMPLES:

JS #SQUARE,V1<5	Jump to subroutine #SQUARE if V1 is less than 5
JS #LOOP,V1<>0	Jump to #LOOP if V1 is not equal to 0
JS #A	Jump to subroutine #A (no condition)

KD

FUNCTION: Derivative Constant

DESCRIPTION:

KD designates the derivative constant in the control filter. The filter transfer function is

$D(z) = 4 \cdot KP + 4 \cdot KD(z-1)/z + KIz/2\ (z-1)$

For further details on the filter see the section Theory of Operation.

ARGUMENTS: KD n,n,n,n,n,n,n,n or KDA=n where

n is an unsigned numbers in the range 0 to 4095.875 with a resolution of 1/8.

n = ? Returns the value of the derivative constant for the specified axis.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	64
Default Format	4.2

OPERAND USAGE:

_KDa contains the value of the derivative constant for the specified axis.

RELATED COMMANDS:

"KI"	Integrator
"KP"	Proportional

EXAMPLES:

KD 100,200,300,400.25	Specify KD
KD ?,?,?,?	Return KD
0100.00,0200.00,0300.00,0400.25	

KI

FUNCTION: Integrator

DESCRIPTION:

The KI command sets the integral gain of the control loop. It fits in the control equation as follows:

$D(z) = 4 \cdot KP + 4 \cdot KD(z-1)/z + KI\ z/2(z-1)$

The integrator term will reduce the position error at rest to zero.

ARGUMENTS: KI n,n,n,n,n,n,n,n or KIA=n where

n is an unsigned numbers in the range 0 to 2047.875 with a resolution of 1/128.

n = ? Returns the value of the derivative constant for the specified axis.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	0
Default Format	4.0

OPERAND USAGE:

_KIa contains the value of the derivative constant for the specified axis.

RELATED COMMANDS:

"KP"	Proportional Constant
"KI"	Integrator
"IL"	Integrator Limit

EXAMPLES:

KI 12,14,16,20	Specify a,b,c,d-axis integral
KI 7	Specify a-axis only
KI ,,8	Specify c-axis only
KI ?,?,?,?	Return A,B,C,D
0007,0014,0008,0020	KI values

KP

FUNCTION: Proportional Constant

DESCRIPTION:

KP designates the proportional constant in the controller filter. The filter transfer function is

$$D(z) = 4 \cdot KP + 4 \cdot KD(z-1)/z + KI\ z/2(z-1)$$

For further details see the section Theory of Operation.

ARGUMENTS: KP n,n,n,n,n,n,n,n or KPA=n where

n is an unsigned numbers in the range 0 to 1023.875 with a resolution of 1/8.

n = ? Returns the value of the proportional constant for the specified axis.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	6
Default Format	4.2

OPERAND USAGE:

_KPa contains the value of the proportional constant for the specified axis.

RELATED COMMANDS:

"KP"	Proportional Constant
"KI"	Integrator
"IL"	Integrator Limit

LA

FUNCTION: List Arrays

DESCRIPTION:

The LA command returns a list of all arrays in memory. The listing will be in alphabetical order. The size of each array will be included next to each array name in square brackets.

ARGUMENTS: None

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS WITH DIRECT CONNECTION**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

"LL"	List Labels
"LS"	List Program
"LV"	List Variable

EXAMPLES:

```
: LA
CA [10]
LA [5]
NY [25]
VA [17]
```

LE

FUNCTION: Linear Interpolation End

DESCRIPTION: LE

Signifies the end of a linear interpolation sequence. It follows the last LI specification in a linear sequence. After the LE specification, the controller issues commands to decelerate the motors to a stop. The VE command is interchangeable with the LE command.

ARGUMENTS:

n = ? Returns the total vector move length in encoder counts.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS, LOCAL AXES ONLY**

DEFAULTS:

Default Value	-
Default Format	-

OPERAND USAGE:

_LEa contains the total vector move length in encoder counts.

RELATED COMMANDS:

"LI"	Linear Distance
"BG"	BGS - Begin Sequence
"LM"	Linear Interpolation Mode
"VS"	Vector Speed
"VA"	Vector Acceleration
"VD"	Vector Deceleration

EXAMPLES:

CAS	Specify S coordinated motion system
LM CD	Specify linear interpolation mode for C and D axes
LI ,,100,200	Specify linear distance
LE	End linear move
BGS	Begin motion

_LF*

FUNCTION: Forward Limit Switch Operand (Keyword)

DESCRIPTION:

The _LF operand contains the state of the forward limit switch for the specified axis.

The operand is specified as: _LFa where a is the specified axis.

Note: This operand is affected by the configuration of the limit switches set by the command CN:

For CN -1:

_LFa = 1 when the limit switch input is inactive*

_LFa = 0 when the limit switch input is active*

For CN 1:

_LFa = 0 when the limit switch input is inactive*

_LFa = 1 when the limit switch input is active*

* The term “active” refers to the condition when at least 1ma of current is flowing through the input circuitry. The input circuitry can be configured to sink or source current to become active. See Chapter 3 for further details.

RELATED COMMANDS:

“_LR”	Reverse limit status

EXAMPLES:

MG _LF A	Display the status of the A axis forward limit switch

* This is an Operand - Not a command.

LI

FUNCTION: Linear Interpolation Distance

DESCRIPTION:

The LI a,b,c,d command specifies the incremental distance of travel for each axis in the Linear Interpolation (LM) mode. LI parameters are relative distances given with respect to the current axis positions. Up to 511 LI specifications may be given ahead of the Begin Sequence (BGS) command. Additional LI commands may be sent during motion when the controller sequence buffer frees additional spaces for new vector segments. The Linear End (LE) command must be given after the last LI specification in a sequence. This command tells the controller to decelerate to a stop at the last LI command. It is the responsibility of the user to keep enough LI segments in the controller's sequence buffer to ensure continuous motion.

LM ? Returns the available spaces for LI segments that can be sent to the buffer. 511 returned means the buffer is empty and 511 LI segments can be sent. A zero means the buffer is full and no additional segments can be sent. It should be noted that the controller computes the vector speed based on the axes specified in the LM mode. For example, LM ABC designates linear interpolation for the A,B and C axes. The speed of these axes will be computed from $VS^2=AS^2+BS^2+CS^2$ where AS, BS and CS are the speed of the A,B, and C axes. If the LI command specifies only A and B, the speed of C will still be used in the vector calculations. The controller always uses the axis specifications from LM, not LI, to compute the speed. The parameter n is optional and can be used to define the vector speed that is attached to the motion segment.

ARGUMENTS: LI n,n,n,n,n,n,n,n <o >p or LIA=n where

n is a signed integer in the range -8,388,607 to 8,388,607 and represents the incremental move distance (at least one n must be non-zero).

o specifies a vector speed to be taken into effect at the execution of the linear segment. s is an unsigned even integer between 0 and 12,000,000 for servo motor operation and between 0 and 3,000,000 for stepper motors.

p specifies a vector speed to be achieved at the end of the linear segment. o is an unsigned even integer between 0 and 8,000,000.

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	-
Command Line	Yes		
Controller Usage	**ALL CONTROLLERS, LOCAL AXES ONLY**		

(LI cont.)

RELATED COMMANDS:

"LE"	Linear end
"BG"	BGS - Begin sequence
"LM"	Linear Interpolation Mode
"CS"	Clear Sequence
"VS"	Vector Speed
"VA"	Vector Acceleration
"VD"	Vector Deceleration

EXAMPLES:

LM ABC	Specify linear interpolation mode
LI 1000,2000,3000	Specify distance
LE	Last segment
BGS	Begin sequence

#LIMSWI

FUNCTION: Limit switch automatic subroutine

DESCRIPTION:

Without #LIMSWI defined, the controller will effectively issue the STn on the axis when it's limit switch is tripped. With #LIMSWI defined, the axis is still stopped, and in addition, code is executed. #LIMSWI is most commonly used to turn the motor off when a limit switch is tripped (see example below). For #LIMSWI to run, code must be running in thread 0 AND the switch corresponding to the direction of motion must be tripped (forward limit switch for positive motion and negative limit switch for negative motion). #LIMSWI interrupts thread 0 when it runs.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	No
Controller Usage	**ALL**

RELATED COMMANDS:

_LFX	State of forward limit switch
_LRX	State of reverse limit switch
RE	Return from error routine

EXAMPLES:

```
#Main         ;'print a message every second
  MG "Main"
  WT1000
JP#Main
EN

#LIMSWI       ;'runs when a limit switch is tripped
  IF (_LFX = 0) | (_LRX = 0)
    MG "X"
    DCX=67107840
    STX
    AMX
    MOX
  ELSE; IF (_LFY = 0) | (_LRY = 0)
    MG "Y"
    DCY=67107840
    STY
    AMY
    MOY
  ENDIF; ENDIF
RE1
```

***NOTE**: An application program must be executing for the automatic subroutine to function, which runs in thread 0.*

***NOTE**: Use RE to end the routine*

LL

FUNCTION: List Labels

DESCRIPTION:

The LL command returns a listing of all of the program labels in memory. The listing will be in alphabetical order.

ARGUMENTS: None

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS WITH DIRECT CONNECTION**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

"LA"	List Arrays
"LS"	List Program
"LV"	List Variables

EXAMPLES:

```
: LL
# FIVE
# FOUR
# ONE
# THREE
# TWO
```

LM

FUNCTION: Linear Interpolation Mode

DESCRIPTION:

The LM command specifies the linear interpolation mode and specifies the axes for linear interpolation. Any set of 1 thru 8 axes may be used for linear interpolation. LI commands are used to specify the travel distances for linear interpolation. The LE command specifies the end of the linear interpolation sequence. Several LI commands may be given as long as the controller sequence buffer has room for additional segments. Once the LM command has been given, it does not need to be given again unless the VM command has been used.

It should be noted that the controller computes the vector speed based on the axes specified in the LM mode. For example, LM ABC designates linear interpolation for the A,B and C axes. The speed of these axes will be computed from $VS^2=AS^2+BS^2+CS^2$, where AS, BS and CS are the speed of the A, B and C axes. In this example, if the LI command specifies only A and B, the speed of C will still be used in the vector calculations. The controller always uses the axis specifications from LM, not LI, to compute the speed.

ARGUMENTS: LM xxxxxxxxxx where

x is A,B,C,D,E,F,G or H or any combination to specify the axis or axes

x = ? Returns the number of spaces available in the sequence buffer for additional LI commands.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS, LOCAL AXES ONLY**

DEFAULTS:

Default Value	-
Default Format	-

OPERAND USAGE:

_LMa contains the number of spaces available in the sequence buffer for the coordinate system.

RELATED COMMANDS:

"LE"	Linear end
"LI"	Linear Distance
"VA"	Vector acceleration
"VS"	Vector Speed
"VD"	Vector deceleration
"AV"	Vector distance
"CS"	_CS - Sequence counter

EXAMPLES:

LM ABCD	Specify linear interpolation mode
VS 10000; VA 100000;VD 1000000	Specify vector speed, acceleration and deceleration
LI 100,200,300,400	Specify linear distance
LI 200,300,400,500	Specify linear distance
LE; BGS	Last vector, then begin motion

LO

FUNCTION: Lockout

DESCRIPTION:

The lockout command is used to lockout a particular handle or serial port with the master controller on a distributed control system. This function ignores all data received to the master on the specified communication channel.

ARGUMENTS: LO h,n where

h is the handle, A thru H, or the letter S for the serial port. This identifies the communication channel to be locked out.

n = 1 or no argument to enable the lockout.

n = -1 to remove the lockout.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	--
Default Format	--

OPERAND USAGE:

_LOh contains the state of the lockout for Handle A-H or S

RELATED COMMANDS:

"CH"	Connect to Internet Handles for slaves
“IH”	Set Internet Handles
"NA"	Set Number of Axes for Distributed Control System
“QW”	Set Slave Data Record Update Rate
“SA”	Send Command to Slave

EXAMPLES:

LOS	Lockout information received from the serial port
WT10000	Wait 10 seconds
LOS,-1	Re-enable the serial port

LR

FUNCTION: Launch Slave Record

DESCRIPTION:

The LR command causes the slave controller in a distributed control system to launch a data record to the master controller. This command is executed on the slave controller and will be ignored if the QW command has not already setup the data record.

ARGUMENTS: LR

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	---
Default Format	---

RELATED COMMANDS:

"QW"	Set Slave Data Record Update Rate
"ZA"	Set first and second user variables
"ZB"	Set third and fourth user variables

EXAMPLES:

CHC=A,B	Using one DMC-3425 as a master and one DMC-3425 as a slave. This command assigns the slave, identified by the C axis designator, with Handle A for commands and Handle B for status returned from the slave.
QWB=20	Sets the update rate for the slave controller to 20 msec (TM=1000)

_LR*

FUNCTION: Reverse Limit Switch Operand (Keyword)

DESCRIPTION:

The _LR operand contains the state of the reverse limit switch for the specified axis.

The operand is specified as: _LRa where a is the specified axis.

Note: This operand is affected by the configuration of the limit switches set by the command CN:

For CN -1:

_LRa = 1 when the limit switch input is inactive*

_LRa = 0 when the limit switch input is active*

For CN 1:

_LRa = 0 when the limit switch input is inactive*

_LRax = 1 when the limit switch input is active*

* The term “active” refers to the condition when at least 1ma of current is flowing through the input circuitry. The input circuitry can be configured to sink or source current to become active. See Chapter 3 for further details.

RELATED COMMANDS:

“_LF”	Forward limit status

EXAMPLES:

MG _LR A	Display the status of the A axis reverse limit switch

**Note: This is an Operand - Not a command*

LS

FUNCTION: List Program

DESCRIPTION:

The LS command returns a listing of the programs in memory.

ARGUMENTS: LS n,m where

n and m are valid numbers from 0 to 499, or labels. n is the first line to be listed, m is the last.

n is an integer in the range of 0 to 499 or a label in the program memory. n is used to specify the first line to be listed.

m is an integer in the range of 1 to 499 or a label on the program memory. m is used to specify the last line to be listed.

USAGE:

While Moving	Yes	Default Value	0, Last Line
In a Program	No	Default Format	-
Command Line	Yes		
Controller Usage	**ALL CONTROLLERS WITH DIRECT CONNECTION**		

DEFAULTS:

RELATED COMMANDS:

"LA"	List Arrays
"LL"	List Labels
"LV"	List Variables

EXAMPLES:

:LS #A,6	List program starting at #A through line 6
002 #A	
003 PR 500	
004 BGA	
005 AM	
006 WT 200	

Hint: *Remember to quit the Edit Mode <cntrl> Q prior to giving the LS command.*

LV

FUNCTION: List Variables

DESCRIPTION:

The LV command returns a listing of all of the program variables in memory. The listing will be in alphabetical order.

ARGUMENTS: None

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	-
Command Line	Yes		
Controller Usage	**ALL CONTROLLERS WITH DIRECT CONNECTION**		

RELATED COMMANDS:

"LA"	List Arrays
"LS"	List Program
"LL"	List Labels

EXAMPLES:

```
: LV
APPLE  = 60.0000
BOY    = 25.0000
ZEBRA = 37.0000
```

LZ

FUNCTION: Leading Zeros

DESCRIPTION:

The LZ command is used for formatting the values returned from interrogation commands or interrogation of variables and arrays. By enabling the LZ function, all leading zeros of returned values will be removed.

ARGUMENTS: LZ n where

n = 1 Removes leading zeros

n = 0 Does not remove leading zeros.

n = ? Returns the state of the LZ function. '0' does not remove and '1' removes zeros

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	0
Default Format	-

OPERAND USAGE:

_LZ contains the state of the LZ function. '0' is disabled and '1' is enabled.

RELATED COMMANDS:

"PF"	Position Format

EXAMPLES:

LZ 0	Disable the LZ function
TPA	Interrogate the controller for current position of A axis
0000021645.0000	Value returned by the controller
VAR1=	Request value of variable "VAR1" (previously set to 10)
0000000010.0000	Value of variable returned by controller
LZ1	Enable LZ function
TPA	Interrogate the controller for current position of A axis
21645.0000	Value returned by the controller
VAR1=	Request value of variable "VAR1" (previously set to 10)
10.0000	Value of variable returned by controller

MB

FUNCTION: Modbus

DESCRIPTION:

The MB command is used to communicate with I/O devices using the first two levels of the Modbus protocol.

The format of the command varies depending on each function code. The function code, -1, designates that the first level of Modbus is used (creates raw packets and receives raw data). The other codes are the 10 major function codes of the second level that the DMC-3425 supports.

FUNCTION CODE	DEFINITION
01	Read Coil Status (Read Bits)
02	Read Input Status (Read Bits)
03	Read Holding Registers (Read Words)
04	Read Input Registers (Read Words)
05	Force Single Coil (Write One Bit)
06	Preset Single Register (Write One Word)
07	Read Exception Status (Read Error Code)
15	Force Multiple Coils (Write Multiple Bits)
16	Preset Multiple Registers (Write Words)
17	Report Slave ID

Note: For those command formats that have “addr”, this is the slave address. The slave address may be designated or defaulted to the device handle number.

Note: All the formats contain an h parameter. This designates the connection handle number (A thru H).

ARGUMENTS:

MBh = -1, len, array[] where

len is the number of the bytes

Array[] is the name of array containing data

MBh = addr, 1, m, n, array[] where

m is the starting bit number

n is the number of bits

array[] of which the first element will hold result

MBh = addr, 2, m, n, array[] where

m is the starting bit number

n is the number of bits

array[] of which the first element will hold result

MBh = addr, 3, m, n, array[] where

m is the starting register number

n is the number of registers

array[] will hold the response

MBh = addr, 4, m, n, array[] where

m is the starting register number

n is the number of registers

array[] will hold the response

MBh = addr, 5, m, n where

m is the starting bit number

n is 0 or 1 and represents the coil set to off or on.

MBh = addr, 6, m, n where

m is the register number

n is the 16 bit value

MBh = addr, 7, array[] where

array[] is where the returned data is stored (one byte per element)

MBh = addr, 15, m, n, array[] where

m is the starting bit number

n is the number of bits

array[] contains the data (one byte per element)

MBh = addr, 16, m, n, array[] where

m is the starting register number

n is the number of registers

array[] contains the data (one 16 bit word per element)

MBh = addr, 17, array[] where

array[] is where the returned data is stored

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

"IA"	IP Address
"IH"	Internet Handle

MC

FUNCTION: Motion Complete - "In Position"

DESCRIPTION:

The MC command is a trippoint used to control the timing of events. This command will hold up execution of the following commands until the current move on the specified axis or axes is completed and the encoder reaches or passes the specified position. Any combination of axes may be specified with the MC command. For example, MC AB waits for motion on both the A and B axis to be complete. MC with no parameter specifies that motion on all axes is complete. The command TW sets the timeout to declare an error if the encoder is not in position within the specified time. If a timeout occurs, the trippoint will clear and the stopcode will be set to 99. An application program will jump to the special label.

When used in stepper mode, the controller will hold up execution of the proceeding commands until the controller has generated the same number of steps as specified in the commanded position. Using the interrogation command TD can monitor the actual number of steps that have been generated. Note: The MC command is recommended when operating with stepper motors since the generation of step pulses can be delayed due to the stepper motor smoothing function, KS. In this case, the MC command would only be satisfied after all steps are generated.

ARGUMENTS: MC nnnnnnnn where

n is A,B,C,D,E,F,G or H or any combination to specify the axis or axes

No argument specifies that motion on all axes is complete.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

"BG"	Begin
"AM"	After Move
"TW"	Timeout

EXAMPLES:

#MOVE	Program MOVE
PR2000,4000	Independent Move on A and B axis
BG AB	Start the A and B-axes
MC AB	After the move is complete on T coordinate system,
MG "DONE"; TP	Print message
EN	End of Program

Hint: MC can be used to verify that the actual motion has been completed.

#MCTIME

FUNCTION: MC command timeout automatic subroutine

DESCRIPTION:

#MCTIME runs when the MC command is used to wait for motion to be complete and the actual position TP does not reach or pass the target _PA + _PR within the specified timeout TW.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	No
Controller Usage	**ALL**

RELATED COMMANDS:

MC	Wait for motion complete trip point
TW	MC timeout
EN	End routine

EXAMPLES:

```
#BEGIN                ;'Begin main program
  TWX=1000            ;'Set the time out to 1000 ms
  PRX=10000           ;'Position relative
  BGX                 ;'Begin motion
  MCX                 ;'Motion Complete trip point
EN                    ;'End main program

#MCTIME               ;'Motion Complete Subroutine
  MG "X fell short"   ;'Send out a message
EN1                   ;'End subroutine
```

***NOTE**: An application program must be executing for the automatic subroutine to function, which runs in thread 0.*

***NOTE**: Use EN to end the routine*

MF

FUNCTION Forward Motion to Position

DESCRIPTION:

The MF command is a trippoint used to control the timing of events. This command will hold up the execution of the following command until the specified motor moves forward and crosses the position specified*. The units of the command are in quadrature counts. Only one axis may be specified at a time. The MF command only requires an encoder and does not require that the axis be under servo control.

* When using a stepper motor, this condition is satisfied when the stepper position (as determined by the output buffer) has crossed the specified Forward Motion Position. For further information see Chapter 6 of the User Manual *"Stepper Motor Operation"*.

ARGUMENTS: MF n,n,n,n,n,n,n,n or MFA=n where

n is a signed integer in the range -2147483648 to 2147483647 decimal

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

"AD"	Trippoint for after Relative Distances
"AP"	Trippoint for after Absolute Position

EXAMPLES:

#TEST	Program B
DP0	Define zero
JG 1000	Jog mode (speed of 1000 counts/sec)
BG A	Begin move
MF 2000	After passing the position 2000
V1=_TPA	Assign V1 A position
MG "Position is",V1;ST	Print Message; Stop
EN	End of Program

Hint: *The accuracy of the MF command is the number of counts that occur in 2 msec. Multiply the speed by 2 msec to obtain the maximum error. MF tests for absolute position. The MF command can also be used when the specified motor is driven independently by an external device.*

MG

FUNCTION: Message

DESCRIPTION:

The MG command sends data out the bus. This can be used to alert an operator, send instructions or return a variable value.

ARGUMENTS: MG "m", {^n}, V {Fm.n or $m,n} {N} {Pn} where

"m" is a text message including letters, numbers, symbols or <ctrl>G (up to 72 characters).

{^n} is an ASCII character specified by the value n

{Ex}for Ethernet and 'x' specifies the Ethernet handle (A,B,C,D,E,F,G or H).

V is a variable name or array element where the following formats can be used:

{Fm.n} Display variable in decimal format with m digits to left of decimal, and n to the right.

{Zm.n} Same as {Fm.n} but suppresses the leading zeros.

{$m,n} Display variable in hexadecimal format with m digits to left of decimal, and n to the right.

{Sn} Display variable as a string of length n where n is 1 through 6

{N} Suppress carriage return line feed.

Note: Multiple text, variables, and ASCII characters may be used, each must be separated by a comma.

Note: The order of arguments is not important.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	--
Default Format	Variable Format

EXAMPLES:

Case 1: Message command displays ASCII strings

MG "Good Morning" Displays the string

Case 2: Message command displays variables or arrays

MG "The Answer is", Total {F4.2} Displays the string with the content of variable TOTAL in local format of 4 digits before and 2 digits after the decimal point.

Case 3: Message command sends any ASCII characters to the port.

MG {^13}, {^10}, {^48}, {^055} displays carriage return and the characters 0 and 7.

MO

FUNCTION: Motor Off

DESCRIPTION:

The MO command shuts off the control algorithm. The controller will continue to monitor the motor position. To turn the motor back on use the Servo Here command (SH).

ARGUMENTS: MO nnnnnnnn where

n is A,B,C,D,E,F,G or H or any combination to specify the axis or axes.

No argument specifies all axes.

USAGE:

While Moving	No
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	0
Default Format	1.0

OPERAND USAGE:

_MOa contains the state of the motor for the specified axis.

RELATED COMMANDS:

"SH"	Servo Here

EXAMPLES:

MO	Turn off all motors
MOA	Turn off the A motor. Leave the other motors unchanged
MOB	Turn off the B motor. Leave the other motors unchanged
MOCA	Turn off the C and A motors. Leave the other motors unchanged
SH	Turn all motors on
Bob=_MOA	Sets Bob equal to the A-axis servo status
Bob=	Return value of Bob. If 1, in motor off mode, If 0, in servo mode

Hint: *The MO command is useful for positioning the motors by hand. Turn them back on with the SH command.*

MR

FUNCTION: Reverse Motion to Position

DESCRIPTION:

The MR command is a trippoint used to control the timing of events. This command will hold up the execution of the following command until the specified motor moves backward and crosses the position specified*. The units of the command are in quadrature counts. Only one axis may be specified at a time. The MR command only requires an encoder and does not require that the axis be under servo control.

* When using a stepper motor, this condition is satisfied when the stepper position (as determined by the output buffer) has crossed the specified Reverse Motion Position. For further information see Chapter 6 of the User Manual *"Stepper Motor Operation"*.

ARGUMENTS: MR n,n,n,n,n,n,n,n or MRA=n where

n is a signed integers in the range -2147483648 to 2147483647 decimal

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	
In a Program	Yes	Default Format	
Command Line	Yes		
Controller Usage	**ALL CONTROLLERS**		

RELATED COMMANDS:

"AD"	Trippoint for Relative Distances
"AP"	Trippoint for after Absolute Position

EXAMPLES:

#TEST	Program B
DP0	Define zero
JG -1000	Jog mode (speed of 1000 counts/sec)
BG A	Begin move
MR -3000	After passing the position -3000
V1=_TPA	Assign V1 A position
MG "Position is", V1	Print Message
ST	Stop
EN	End of Program

Hint: *The accuracy of the MR command is the number of counts that occur in 2 msec. Multiply the speed by 2 msec to obtain the maximum error. MR tests for absolute position. The MR command can also be used when an external device drives the specified motor independently.*

MT

FUNCTION: Motor Type

DESCRIPTION:

The MT command selects the type of the motor and the polarity of the drive signal. Motor types include standard servomotors, which require a voltage in the range of +/- 10 Volts, and step motors, which require pulse and direction signals. The polarity reversal inverts the analog signals for servomotors, and inverts logic level of the pulse train, for step motors.

ARGUMENTS: MT n,n,n,n,n,n,n,n or MTA=n where

n = 1	Specifies Servo motor
n = -1	Specifies Servo motor with reversed polarity
n = -2	Specifies Step motor with active high step pulses
n = 2	Specifies Step motor with active low step pulses
n = -2.5	Specifies Step motor with reversed direction and active high step pulses
n = 2.5	Specifies Step motor with reversed direction and active low step pulses
n = ?	Returns the value of the motor type for the specified axis.

USAGE:

While Moving	No
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS ***

DEFAULTS:

Default Value	1,1,1,1
Default Format	1

*** Note**: For two axes of stepper control on a single DMC-3425, the card must be ordered as a DMC-3425-Stepper.

OPERAND USAGE:

_MTa contains the value of the motor type for the specified axis.

RELATED COMMANDS:

"CE"	Configure encoder type

EXAMPLES:

MT 1,-1,2,2	Configure A as servo, B as reverse servo, C and D as steppers
MT ?,?	Interrogate motor type
V=_MTA	Assign motor type to variable

MW

FUNCTION: Modbus Wait

DESCRIPTION:

Enabling the MW command causes the controller to hold up execution of the program after sending a Modbus command until a response from the Modbus device has been received. If the response is never received, then the #TCPERR subroutine will be triggered and an error code of 123 will occur on _TC.

ARGUMENTS: MWn where

n = 0 Disables the Modbus Wait function

n = 1 Enables the Modbus Wait function

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	0
Default Format	1.0

OPERAND USAGE:

MW? contains the state of the Modbus Wait.

RELATED COMMANDS:

"MB"	Modbus

EXAMPLES:

MW1	Enables Modbus Wait
SB1001	Set Bit 1 on Modbus Handle A
CB1001	Clear Bit 1 on Modbus Handle A

Hint: *The MW command ensures that the command that was sent to the Modbus device was successfully received before continuing program execution. This prevents the controller from sending multiple commands to the same Modbus device before it has a chance to execute them.*

NA

FUNCTION: Number of Axes

DESCRIPTION:

NA defines the total number of axes used in a distributed control system. This command is issued on the master controller. For example; Using 3 DMC-3425 controllers (1 master and 2 slaves), there would be 6 axes; the command NA6 would be given to the master controller.

This command is not necessary when using the automatic setup procedure HC.

ARGUMENTS: NA n where

n = an integer between 1 and 8. This number represents the number of axes in a distributed control system.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	1 (DMC-3415)
	2 (DMC-3425)
Default Format	--

OPERAND USAGE:

_NA contains the number of axes configured in the distributed control system.

RELATED COMMANDS:

"CH"	Connect to internet handles for slave operation
"IH"	Set Internet Handles
"QW"	Set Slave Data Record Update Rate

EXAMPLES:

NA 4	Command given to DMC-3425 master controller with 1 DMC-3425 slave
NA 5	Master controller command with 1 DMC-3425 slave and 1 DMC3415 slave

NB

FUNCTION: Notch Bandwidth

DESCRIPTION:

The NB command sets real part of the notch poles

ARGUMENTS: NB n,n,n,n,n,n,n,n or NBA=n where

n is ranges from 0 Hz to $\frac{1}{(16 \cdot TM)}$

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	0.5
Default Format	

RELATED COMMANDS:

"NF"	Notch Filter
"NZ"	Notch Zeros

EXAMPLES:

_NBA = 10	Sets the real part of the notch pole to 10/2 Hz
NOTCH = _NBA	Sets the variable "NOTCH" equal to the notch bandwidth value for the A axis

NF

FUNCTION: Notch Frequency

DESCRIPTION:

The NF command sets the frequency of the notch filter, which is placed in series with the PID compensation.

ARGUMENTS: NF n,n,n,n,n,n,n,n or NFA=n where

n ranges from 1 Hz to $\frac{1}{(4 \cdot TM)}$ where TM is the update rate (default TM is 1 msec).

n = ? Returns the value of the Notch filter for the specified axis.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	0
Default Format	

OPERAND USAGE:

_NFa contains the value of notch filter for the specified axis.

RELATED COMMANDS:

"NB"	Notch bandwidth
"NZ"	Notch Zero

EXAMPLES:

NF, 20	Sets the notch frequency of B axis to 20 Hz

NO (‘ apostrophe also accepted)

FUNCTION: No Operation

DESCRIPTION:

The NO or an apostrophe (‘) command performs no action in a sequence, but can be used as a comment in a program. This helps to document a program.

ARGUMENTS: NO m where

m is any group of letters and numbers

up to 77 characters can follow the NO command

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	
Default Format	

EXAMPLES:

#A	Program A
NO	No Operation
NO This Program	No Operation
NO Does Absolutely	No Operation
NO Nothing	No Operation
EN	End of Program

NZ

FUNCTION: Notch Zero

DESCRIPTION:

The NZ command sets the real part of the notch zero.

ARGUMENTS: NZ n,n,n,n,n,n,n,n or NZA=n where

n is ranges from 1 Hz to $\frac{1}{(16 \cdot TM)}$

n = ? Returns the value of the Notch filter zero for the specified axis.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	0.5
Default Format	

OPERAND USAGE:

_NZa contains the value of the Notch filter zero for the specified axis.

RELATED COMMANDS:

"NB"	Notch Bandwidth
"NF"	Notch Filter

EXAMPLES:

NZA = 10	Sets the real part of the notch pole to 10/2 Hz

OB

FUNCTION: Output Bit

DESCRIPTION:

The OB n, logical expression command defines an output as either 0 or 1 depending on the result from the logical expression. Any non-zero value of the expression results in a one on the output. OB can be used to set outputs of extended I/O (when the I/O has been configured to operate as outputs). A master controller to set the outputs on slave controllers or IOC-7007 controllers when used in a distributed control system can also use OB.

ARGUMENTS: OB n, *expressio*n where

n denotes the output bit

expression is any valid logical expression, variable or array element.

The outputs of the slave devices are calculated using the following formula:

n = (HandleNum*100) + (Bitnum)

HandleNum is the number associated with the handle specifier 1 for handle A, 2 for handle B, etc.

BitNum is the specific output of the slave device. This includes extended I/O that has been configured to operate as outputs.

The outputs on an IOC-7007 I/O controller may also be defined through the master controller. The outputs for the IOC-7007's IOM modules are calculated using the following formula:

n = ((HandleNum*1000) + (Bitnum) where

HandleNum is the number associated with the handle specifier for the particular IOC controller. 1 for handle A, 2 for handle B, etc.

Bitnum is the specific output bit on the IOC controller to be set or cleared.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Yes	Default Value
Yes	Default Format

RELATED COMMANDS:

"SB"	Set Bit
"CB"	Clear Bit
"OP"	Set Output Port

EXAMPLES:

OB 1, POS=1	If POS 1 is non-zero, Bit 1 is high.
	If POS 1 is zero, Bit 1 is low
OB 2, @IN[1]&@IN[2]	If Input 1 and Input 2 are both high, then
	Output 2 is set high
OB 3, COUNT[1]	If the element 1 in the array is zero, clear bit 3
OB N, COUNT[1]	If element 1 in the array is zero, clear bit N

OC

FUNCTION: Output Compare

DESCRIPTION:

The OC command allows the generation of output pulses based on one of the main encoder positions. For circular compare, the output is a low-going pulse with duration of approximately 600 nanoseconds and is available at the output compare signal. For one shot, the output goes low until OC is called again.

This function cannot be used with any axis configured for a step motor and the auxiliary encoder of the corresponding axis cannot be used while using this function.

ARGUMENTS: OCa = m, n where

a = A,B,C,D,E,F,G H specifies which encoder input to be used.

m = Absolute position for first pulse. Integer between $-2 \cdot 10^9$ and $2 \cdot 10^9$

n = Incremental distance between pulses. Integer between -65535 and 65535, 0 one shot.

Notes:

OCx = 0 will disable the Circular Compare function.

The sign of the parameter, n, will designate the expected direction of motion for the output compare function. When moving in the opposite direction, output compare pulses will occur at the incremental distance of 65536-|n| where |n| is the absolute value of n.

When changing to CEx = 2, if the original command was OCx = m,n and the starting position was _TPx, the new command is OCx = 2*_TPx-m, -n. For pulses to occur under CEx =2, the following conditions must be met:

m >_TPx and n > 0 for negative moves (e.g. JGx = -1000)

m <_TPx and n < 0 for positive moves (e.g. JGx = 1000)

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	-
Default Format	-

OPERAND USAGE:

_OCa contains the state of the OC function

_OCa = 0 : OC function has been enabled but not generated any pulses.

_OCa = 1: OC function not enables or has generated the first output pulse.

EXAMPLES:

OCA=300,100	Select A encoder as position sensor. First pulse at 300. Following pulses at 400, 500…

OE

FUNCTION: Off on Error

DESCRIPTION:

The OE command causes the controller to shut off the motor command if a position error exceeds the limit specified by the ER command occurs or an abort occurs from either the abort input or on AB command.

If a position error is detected on an axis, and the motion was executing an independent move, only that axis will be shut off. If the motion is a part of coordinated mode of the types VM, LM or CM, all participating axes will be stopped.

ARGUMENTS: OE n,n,n,n,n,n,n,n or OEA=n where

n = 0	Disables the Off-On-Error function.
n = 1	Enables the Off-On-Error function.
n = ?	Returns the state of the off on error function for the specified axis.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	0
Default Format	1.0

OPERAND USAGE:

_OEa contains the status of the off-on-error function for the specified axis. 0 = off, 1 = on

RELATED COMMANDS:

"AB"	Abort
"ER"	Error limit
"SH"	Servo Here
#POSERR	Error Subroutine

EXAMPLES:

OE 1,1,1,1	Enable OE on all axes
OE 0	Disable OE on A-axis other axes remain unchanged
OE ,,1,1	Enable OE on C-axis and D-axis other axes remain unchanged
OE 1,0,1,0	Enable OE on A and C-axis Disable OE on B and D axis

Hint: *The OE command is useful for preventing system damage on excessive error.*

OF

FUNCTION: Offset

DESCRIPTION:

The OF command sets a bias voltage in the motor command output or returns a previously set value. This can be used to counteract gravity or an offset in an amplifier.

ARGUMENTS: OF n,n,n,n,n,n,n,n or OFA=n where

n is a signed number in the range -9.998 to 9.998 volts with resolution of 0.0003.

n = ? Returns the offset for the specified axis.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	0
Default Format	1.0

OPERAND USAGE:

_OFa contains the offset for the specified axis.

EXAMPLES:

OF 1,-2,3,5	Set A-axis offset to 1, the B-axis offset to -2, the C-axis to 3, and the D-axis to 5
OF -3	Set A-axis offset to -3 Leave other axes unchanged
OF ,0	Set B-axis offset to 0 Leave other axes unchanged
OF ?,?,?,?	Return offsets
-3.0000,0.0000,3.0000,5.0000	
OF ?	Return A offset
-3.0000	
OF ,?	Return B offset
0.0000	

OP

FUNCTION: Output Port

DESCRIPTION:

The OP command sends data to the output ports of the controller. You can use the output port to control external switches and relays.

ARGUMENTS: OP m,a,b,c,d where

m is an integer in the range 0 to 7 decimal.

m is the decimal representation of the general output bits.

a,b,c,d represent the DB-14064 extended I/O in consecutive groups of 16 bits, (values from 0 to 65535). Arguments that are given for I/O points that are configured, as inputs will be ignored. The following table describes the arguments used to set the state of outputs.

Arguments	Blocks	Bits	Description
m	0	1-3	General Outputs
a	2,3	17-32	Extended I/O
b	4,5	33-48	Extended I/O
c	6,7	49-64	Extended I/O
d	8,9	65-80	Extended I/O

n = ? returns the value of the argument

Note: The OP command is valid in the Local Command mode only. If this command is to be sent through the master to a slave, the SA command must be used.

USAGE:		DEFAULTS:	
While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	3.0
Command Line	Yes		
Controller Usage	**ALL CONTROLLERS.**		

OPERAND USAGE:

_OP0 contains the value of the first argument, m

_OP1 contains the value of the first argument, a

_OP2 contains the value of the first argument, b

_OP3 contains the value of the first argument, c

_OP4 contains the value of the first argument, d

RELATED COMMANDS:

"SB"	Set output bit
"CB"	Clear output bit
"OB"	Output Bit

EXAMPLES:

OP 0	Clear Output Port -- all bits
OP 5	Set outputs 1,3; clear the others
MG _OP0	Returns the first parameter "m"

OQ

FUNCTION: Output Data

DESCRIPTION:

The OQ command writes data to the output port of the IOC-7007 controller when used in a distributed system. These output ports correspond to IOM output modules mounted on the IOC-7007 I/O controller.

ARGUMENTS: OQ a,b where

a is an integer representing the handle and slot number of the output module. This integer is calculated as follows:

a = (HandleNum * 1000) + SlotNum where

HandleNum is the number associated with the handle specifier for the particular IOC-7007. 1 for handle A, 2 for handle B, etc.

SlotNum is the number associated with the slot location of the IOM output to be set, 0 – 6.

b is an integer representing the data to be written to the particular IOC-7007 output slot. The data written will depend on the IOM module in each particular slot and whether they have 8 or 16 outputs.

IOM-70208	b = 0 – 255
IOM-70308	b = 0 - 255
IOM-70404	b = 0 – 15

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**IOC-7007 MODULE ONLY. CAN BE SENT THROUGH MASTER.**

DEFAULTS:

Default Value	0
Default Format	--

OPERAND USAGE:

_OQa where a is the HandleNum and SlotNum as calculated above, contains the value of the output data at the specified location.

RELATED COMMANDS:

"SB"	Set output bit
"CB"	Clear output bit
"OB"	Output Bit

EXAMPLES:

OQ 6001,128	6001 represents handle F, slot 1 of the IOC-7007. 128 is the data written, which will set bit 7 of the specified IOM module.
OQ 4003,0	4003 represents handle D, slot 3 of the IOC-7007. 0 is the data written, which will clear all outputs of the specified IOM module.

@OUT[n]

FUNCTION: Read digital output

DESCRIPTION:

Returns the value of the given digital output (either 0 or 1)

ARGUMENTS: @OUT[n] where

n is an unsigned integer in the range 1 to 80

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

@AN	Read analog input
@IN	Read digital input
SB	Set digital output bit
CB	Clear digital output bit
OF	Set analog output offset

EXAMPLES:

```
:MG @OUT[1] ;'print digital output 1
 1.0000
:x = @OUT[1] ;'assign digital output 1 to a variable
```

PA

FUNCTION: Position Absolute

DESCRIPTION:

The PA command will set the final destination of each axis. The position is referenced to the absolute zero.

ARGUMENTS: PA n,n,n,n,n,n,n,n or PAA=n where

n is a signed integers in the range -2147483647 to 2147483648 decimal. Units are in encoder counts.

n = ? Returns the commanded position at which motion stopped.

USAGE:

While Moving	No
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	-
Default Format	Position Format

OPERAND USAGE:

_PAa contains the last commanded position at which motion stopped.

RELATED COMMANDS:

“PR”	Position relative
“SP”	Speed
“AC”	Acceleration
“DC”	Deceleration
“BG”	Begin
“PF”	Position Formatting

EXAMPLES:

:PA 400,-600,500,200	A-axis will go to 400 counts B-axis will go to –600 counts C-axis will go to 500 counts D-axis will go to 200 counts
BG;AM	Execute Motion and Wait for Motion Complete
:PA ?,?,?,?	Returns the current commanded position after motion has completed
400, -600, 500, 200	
:BG	Start the move
:PA 700	A-axis will go to 700 on the next move while the
:BG	B,C and D-axis will travel the previously set relative distance if the preceding move was a PR move, or will not move if the preceding move was a PA move.

PF

FUNCTION: Position Format

DESCRIPTION:

The PF command allows the user to format the position numbers such as those returned by TP. The number of digits of integers and the number of digits of fractions can be selected with this command. An extra digit for sign and a digit for decimal point will be added to the total number of digits. If PF is minus, the format will be hexadecimal and a dollar sign will precede the characters. Hex numbers are displayed as 2's complement with the first bit used to signify the sign.

If a number exceeds the format, the number will be displayed as the maximum possible positive or negative number (i.e. 999.99, -999, $8000 or $7FF).

The PF command can be used to format values returned from the following commands:

BL ?	LE ?
DE ?	PA ?
DP ?	PR ?
EM ?	TN ?
FL ?	VE ?
IP ?	TE
TP	

ARGUMENTS: PF m,n where

m is an integer between -8 and 10 which represents the number of places preceding the decimal point. A negative sign for m specifies hexadecimal representation.

n is an integer between 0 and 4 which represent the number of places after the decimal point.

n = ? Returns the value of m.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	10.0
Default Format	10.0

OPERAND USAGE:

_PF contains the value of 'm' position format parameter.

EXAMPLES:

:TPA	Tell position of A
0000000000	Default format
:PF 5.2	Change format to 5 digits of integers and 2 of fractions
:TPA	Tell Position
00021.00	
PF-5.2	New format Change format to hexadecimal*
:TPA	Tell Position
$00015.00	Report in hex

#POSERR

FUNCTION: Position error automatic subroutine

DESCRIPTION:

The factory default behavior of the Galil controller upon a position error (TE > ER) is to do nothing more than turn on the red light. If OE is set to 1, the motor whose position error ER was exceeded will be turned off MO. #POSERR can be used if the programmer wishes to run code upon a position error (for example to notify a host computer).

The #POSERR label causes the statements following it to be automatically executed if the error TE on any axis exceeds the error limit specified by ER. The error routine must be closed with the RE command. The RE command returns from the error subroutine to the main program.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	No
Controller Usage	**ALL**

RELATED COMMANDS:

OE	Off on error
TE	Tell error
ER	Error limit
RE	Return from error routine

EXAMPLES:

```
#A              ;'"Dummy" program
JP #A

#POSERR         ;'Position error routine
  MG "TE > ER" ;'Send message
RE1             ;'Return to main program
```

***NOTE**: An application program must be executing for the automatic subroutine to function, which runs in thread 0.*

***NOTE**: Use RE to end the routine*

PR

FUNCTION: Position Relative

DESCRIPTION:

The PR command sets the incremental distance and direction of the next move. The move is referenced with respect to the current position. .

ARGUMENTS: PR n,n,n,n,n,n,n,n or PRA=n where

n is a signed integer in the range -2147483648 to 2147483647 decimal. Units are in encoder counts

n = ? Returns the current incremental distance for the specified axis.

USAGE:

While Moving	No
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	0
Default Format	Position Format

OPERAND USAGE:

_PRa contains the current incremental distance for the specified axis.

RELATED COMMANDS:

"PA"	Position Absolute
"BG"	Begin
"AC"	Acceleration
"DC"	Deceleration
"SP"	Speed
"IP"	Increment Position
"PF"	Position Formatting

EXAMPLES:

:PR 100,200,300,400	On the next move the A-axis will go 100 counts,
:BG	the B-axis will go to 200 counts forward, C-axis will go 300 counts and the D-axis will go 400 counts.
:PR ?,?,?	Return relative distances
0000000100,0000000200,0000000300	
:PR 500	Set the relative distance for the A axis to 500
:BG	The A-axis will go 500 counts on the next move while the B-axis will go its previously set relative distance.

QD

FUNCTION: Download Array

DESCRIPTION:

The QD command transfers array data from the host computer to the controller. QD array [], start, end requires that the array name be specified along with the first element of the array and last element of the array. The array elements can be separated by a comma (,) or by <CR> <LF>. The downloaded array is terminated by a \.

ARGUMENTS: QD array[],start,end where

array[] is valid array name start is first element of array (default=0) end is last element of array (default = last element)

USAGE:

While Moving	No
In a Program	No
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	0
Default Format	Position Format

RELATED COMMANDS:

"QU" Upload array

HINT:

Using Galil terminal software, the command can be used in the following manner:

1. Set the timeout to 0
2. Send the command QD

3a. Use the send file command to send the data file.

OR

3b. Enter data manually from the terminal. End the data entry with the character '\'

QR

FUNCTION: Data Record

DESCRIPTION:

The QR command causes the controller to return a record of information regarding controller status. This status information includes 4 bytes of header information and specific blocks of information as specified by the command arguments. The detail of the status information is described in Chapter 4 of the user's manual.

ARGUMENTS: QR nnnnnnnnnn where

n is A,B,C,D,E,F,G,H,S or I or any combination to specify the axis, axes, sequence, or I/O status

S represents the coordinated motion plane

I represents the status of the I/O

Chapter 4 of the users manual provides the definition of the data record information.

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	-
Command Line	Yes		
Controller Usage	**ALL CONTROLLERS**		

RELATED COMMANDS:

"QZ" Return DMA / Data Record information

Note: The Galil windows terminal will not display the results of the QR command since the results are in binary format.

QU

FUNCTION: Upload Array

DESCRIPTION:

The QU command transfers array data from the controller to a host computer. The QU requires that the array name be specified along with the first element of the array and last element of the array. The uploaded array will be followed by a <control>Z as an end of text marker.

ARGUMENTS: QU array[],start,end,delim where

"array[]" is a valid array name

"start" is the first element of the array (default=0)

"end" is the last element of the array (default = last element)

"delim" specifies the character used to delimit the array elements. If delim is 1, then the array elements will be separated by a comma. Otherwise, the elements will be separated by a carriage return.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	0
Default Format	Position Format

RELATED COMMANDS:

"QD" Download array

QW

FUNCTION: Slave Record Update Rate

DESCRIPTION:

The QW command is given to the master controller of a distributed system. This value establishes the update rate for data records to be sent from the slave controllers to the master controller. This command is executed on the master controller.

ARGUMENTS: QWh=n where

h is the handle being used to send commands to the slave controller.

n = an even integer between 4 and 16000. This sets the period at which the slave controller updates the master controller. The value of n represents the number of servo update cycles (default update cycle is 1msec, see the TM command). The slave controller will always wait for this period after a data record has been sent before generating a new record.

Notes:

1. The update period should be 8msec or greater.

2. The servo update period of the master must be greater than or equal to the update period of the slave (see TM command).

3. A slave record is sent when a trippoint condition has been met.

4. The slave will wait until the elapsed update period after each record before automatically sending a new record.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	---
Default Format	---

RELATED COMMANDS:

"CH"	Configure Handle
"NA"	Number of Axes

EXAMPLES:

CHC=A,B	Using one DMC-3425 as a master and one DMC-3425 as a slave. This command assigns the slave, identified by the C axis designator, with Handle A for commands and Handle B for status returned from the slave.
QWB=20	Sets the update rate for the slave controller to 20 msec (TM=1000)

QZ

FUNCTION: Return DMA / Data Record information

DESCRIPTION:

The QZ command is an interrogation command that returns information regarding the Data Record. The controller's response to this command will be the return of 4 integers separated by commas. The four fields represent the following:

First field returns the number of axes.

Second field returns the number of bytes to be transferred for general status

Third field returns the number bytes to be transferred for coordinated move status

Fourth field returns the number of bytes to be transferred for axis specific information

ARGUMENTS: QZ

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	---
Default Format	

RELATED COMMANDS:

"QR" Data Record

RA

FUNCTION: Record Array

DESCRIPTION:

The RA command selects one through eight arrays for automatic data capture. The selected arrays must be dimensioned by the DM command. The data to be captured is specified by the RD command and time interval by the RC command.

ARGUMENTS: RA n [],m [],o [],p [] RA n[],m[],o[],p[],q[],r[],s[],t[] where

n,m,o and p are dimensioned arrays as defined by DM command. The [] contain nothing.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

"DM"	Dimension Array
"RD"	Record Data
"RC"	Record Interval

EXAMPLES:

#Record	Label
DM POS[100]	Define array
RA POS[]	Specify Record Mode
RD _TPA	Specify data type for record
RC 1	Begin recording at 2 msec intervals
PR 1000;BG	Start motion
EN	End

Hint: *The record array mode is useful for recording the real-time motor position during motion. The data is automatically captured in the background and does not interrupt the program sequencer. The record mode can also be used for a teach or learn of a motion path.*

RC

FUNCTION: Record

DESCRIPTION:

The RC command begins recording for the Automatic Record Array Mode (RA). RC 0 stops recording .

ARGUMENTS: RC n,m where

n is an integer 1 thru 8 and specifies 2^n samples between records. RC 0 stops recording.

m is optional and specifies the number of records to be recorded. If m is not specified, the DM number will be used. A negative number for m causes circular recording over array addresses 0 to m-1. The address for the array element for the next recording can be interrogated with _RD.

n = ? Returns status of recording. '1' if recording, '0' if not recording.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	-
Default Format	-

OPERAND USAGE:

_RC contains status of recording. '1' if recording, '0' if not recording.

RELATED COMMANDS:

"DM"	Dimension Array
"RD"	Record Data
"QZ"	Record Array Mode

EXAMPLES:

#RECORD	Record
DM Torque[1000]	Define Array
RA Torque[]	Specify Record Mode
RD _TTA	Specify Data Type
RC 2	Begin recording and set 4 msec between records
JG 1000;BG	Begin motion
#A;JP #A,_RC=1	Loop until done
MG "DONE RECORDING"	Print message
EN	End program

RD

FUNCTION: Record Data

DESCRIPTION:

The RD command specifies the data type to be captured for the Record Array (RA) mode. The command type includes:

_AFa	Analog Input Value (+32767 to -32768). The analog inputs are limited to those which correspond to an axis on the controller.
_TPa	Position
_TEa	Position error
_SHa	Commanded position
_RLa	Latched position
_TI	Inputs
_OP	Outputs
_TSa	Switches, only 0-4 bits valid
_SCa	Stop code
_TTa	Tell torque (Note: the values recorded for torque are in the range of +/- 32767 where 0 is 0 torque, -32767 is -10 volt command output, and +32767 is +10 volt.
_TVn	Filtered velocity (Note: Will be 65 times greater than TV command)

where 'a' is the axis specifier.

ARGUMENTS: RD m_1, m_2, m_3, m_4, m_5, m_6, m_7, m_8 where

the arguments are data types to be captured using the record Array feature. The order is important. Each data type corresponds with the array specified in the RA command.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	-
Default Format	-

OPERAND USAGE:

_RD contains the address for the next array element for recording.

RELATED COMMANDS:

"RA"	Record Array
"RC"	Record Interval
"DM"	Dimension Array

EXAMPLES:

DM ERRORA[50],ERRORB[50]	Define array
RA ERRORA[],ERRORB[ ]	Specify record mode
RD _TEA,_TEBS	Specify data type
RC1	Begin record
JG 1000;BG	Begin motion

RE

FUNCTION: Return from Error Routine

DESCRIPTION:

The RE command is used to end a position error handling subroutine or limit switch handling subroutine. The error handling subroutine begins with the #POSERR label. The limit switch handling subroutine begins with the #LIMSWI. An RE at the end of these routines causes a return to the main program. Care should be taken to be sure the error or limit switch conditions no longer occur to avoid re-entering the subroutines. If the program sequencer was waiting for a trippoint to occur, prior to the error interrupt, the trippoint condition is preserved on the return to the program if RE1 is used. RE0 clears the trippoint. To avoid returning to the main program on an interrupt, use the ZS command to zero the subroutine stack.

ARGUMENTS: RE n where

n = 0	Clears the interrupted trippoint
n = 1	Restores state of trippoint

no argument clears the interrupted trippoint

USAGE:

While Moving	No
In a Program	Yes
Command Line	No
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

#POSERR	Error Subroutine
#LIMSWI	Limit Subroutine

EXAMPLES:

#A;JP #A;EN	Label for main program
#POSERR	Begin Error Handling Subroutine
MG "ERROR"	Print message
SB1	Set output bit 1
RE	Return to main program and clear trippoint

Hint: *An applications program must be executing for the #LIMSWI and #POSERR subroutines to function.*

REM

FUNCTION: Remark

DESCRIPTION:

REM is used for comments. The REM statement is NOT a controller command. Rather, it is recognized by Galil PC software, which strips away the REM lines before downloading the DMC file to the controller. REM differs from NO (or ‘) in the following ways:

(1) NO comments are downloaded to the controller and REM comments aren’t

(2) NO comments take up execution time and REM comments don’t; therefore, REM should be used for code that needs to run fast.

(3) REM comments cannot be recovered when uploading a program but NO comments are recovered. Thus the uploaded program is less readable with REM.

(4) NO comments take up program line space and REM lines don’t.

(5) REM comments must be the first and only thing on a line, whereas NO can be used to place comments to the right of code on the same line.

NO (or ‘) should be used instead of REM unless speed or program space is an issue.

ARGUMENTS: REM n where

n is a text string comment

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	No
Controller Usage	**ALL**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

NO (or ‘)	No operation (comment)

EXAMPLES:

```
REM This comment will be stripped when downloaded to the controller
'This comment will be downloaded and takes some execution time
PRX=1000 ;'this comment is to the right of the code
```

RI

FUNCTION: Return from Interrupt Routine

DESCRIPTION:

The RI command is used to end the interrupt subroutine beginning with the label #ININT. An RI at the end of this routine causes a return to the main program. The RI command also re-enables input interrupts. If the program sequencer was interrupted while waiting for a trippoint, such as WT, RI1 restores the trippoint on the return to the program. RI0 clears the trippoint. To avoid returning to the main program on an interrupt, use the command ZS to zero the subroutine stack. This turns the jump subroutine into a jump only.

ARGUMENTS: RI n where

n = 0 Clears the interrupted trippoint

n = 1 Restores state of trippoint

no argument clears the interrupted trippoint

USAGE:

While Moving	No
In a Program	Yes
Command Line	No
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

#ININT	Input interrupt subroutine
"II"	Enable input interrupts

EXAMPLES:

#A;II1;JP #A;EN	Program label
#ININT	Begin interrupt subroutine
MG "INPUT INTERRUPT"	Print Message
SB 1	Set output line 1
RI 1	Return to the main program and restore trippoint

Hint: *An applications program must be executing for the #ININT subroutine to function.*

RL

FUNCTION: Report Latched Position

DESCRIPTION:

The RL command will return the last position captured by the latch. The latch must first be armed by the AL command and then a 0 must occur on the appropriate input. Each axis has a position latch and can be activated through the general inputs. For a single axis master or slave, Input 1 is the latch for the axis. For a 2 axis master or slave, Input 1 is the latch for the first axis and Input 2 is the latch for the second axis.

The armed state of the latch can be configured using the CN command.

Note: The Latch Function does not work with stepper motors.

ARGUMENTS: RL nnnnnnnn where

n is A,B,C,D,E,F,G or H or any combination to specify the axis or axes

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	0
Default Format	Position Format

OPERAND USAGE:

_RLa contains the latched position of the specified axis.

RELATED COMMAND:

"AL"	Arm Latch

EXAMPLES:

JG ,5000	Set up to jog the B-axis
BGB	Begin jog
ALB	Arm the B latch; assume that after about 2 seconds, input goes low
RLB	Report the latch
10000	

@RND[n]

FUNCTION: Round

DESCRIPTION:

Rounds the given number to the nearest integer

ARGUMENTS: @RND[n]

n is a signed number in the range -2147483648 to 2147483647.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

@INT	Truncates to the nearest integer

EXAMPLES:

```
:MG @RND[1.2]
 1.0000
:MG @RND[5.7]
 6.0000
:MG @RND[-1.2]
 -1.0000
:MG @RND[-5.7]
 -6.0000
:MG @RND[5.5]
 6.0000
:MG @RND[-5.5]
 -5.0000
:
```

RP

FUNCTION: Reference Position

DESCRIPTION:

This command returns the commanded reference position of the motor(s).

ARGUMENTS: RP nnnnnnnn where

n is A,B,C,D,E,F,G,H or N, or any combination to specify the axis or axes

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	0
Default Format	Position Format

OPERAND USAGE:

_RPa contains the commanded reference position for the specified axis.

RELATED COMMAND:

"TP"	Tell Position

Note: The relationship between RP, TP and TE: TEA equals the difference between the reference position, RPA, and the actual position, _TPA.

EXAMPLES: Assume that ABC and D axes are commanded to be at the positions 200, -10, 0, -110 respectively. The returned units are in quadrature counts.

:PF 7	Position format of 7
0:RP	
0000200,-0000010,0000000,-0000110	Return A,B,C,D reference positions
RPA	
0000200	Return the A motor reference position
RPB	
-0000010	Return the B motor reference position
PF-6.0	Change to hex format
RP	
$0000C8,$FFFFF6,$000000,$FFFF93	Return A,B,C,D in hex
Position =_RPA	Assign the variable, Position, the value of RPA

RS

FUNCTION: Reset

DESCRIPTION:

The RS command resets the state of the processor to its power-on condition. The previously saved state of the controller, along with parameter values, and saved sequences are restored.

The RS-1 command resets the state of the processor to its factory default without modifying the EEPROM.

USAGE:

While Moving	Yes
In a Program	No
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	0
Default Format	-

OPERAND USAGE:

_RS contains the power up error status

BIT	ERROR CONDITION
Bit 3	Master Reset error
Bit 2	Program checksum error
Bit 1	Parameter checksum error
Bit 0	Variable checksum error

<control>R<control>S

FUNCTION: Master Reset

DESCRIPTION:

This command resets the controller to factory default settings and erases EEPROM.

A master reset can also be performed by installing a jumper on the controller at the location labeled MRST and resetting the controller (power cycle or pressing the reset button). Remove the jumper after this procedure.

USAGE:

While Moving	Yes
In a Program	No
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	-
Default Format	-

Note: A master reset is not supported on the Ethernet connection. Any attempt will hang up the host.

<control>R<control>V

FUNCTION: Revision Information

DESCRIPTION:

The Revision Information command causes the controller to return firmware revision information.

USAGE:

While Moving	Yes
In a Program	No
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	-
Default Format	-

SA

FUNCTION: Send Command

DESCRIPTION:

SA sends a command from the master to the slave controller of a distributed control system. Any command can be sent to a slave controller and will be interpreted by the slave as a "local" command. Some commands are only "local" commands and must be sent with the SA command. Please refer to the discussion of local vs global commands at the beginning of this manual.

***Note**: A wait statement (e.g. WT5) must be inserted between successive calls to SA.*

ARGUMENTS: SAh= arg or SAh=arg,arg,arg,arg,arg,arg,arg,arg where

h is the handle being used to send commands to the slave controller.

arg is a command, number, controller operand, variable, mathematical function, or string; The range for numeric values is 4 bytes of integer (2^{31})followed by two bytes of fraction (+/-2,147,483,647.9999). The maximum number of characters for a string is 6 characters. Strings are identified by quotations.

Typical usage would have the first argument as a string such as "KI" and the subsequent arguments as the arguments to the command: Example SAF= "KI",1,2 would send the command

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	--
Default Format	--

OPERAND USAGE:

_SAhn gives the value of the response to the command sent with an SA command. The h value represents the handle A thru H and the n value represents the specific field returned from the controller (1-8). If the specific field is not used, the operand will be –2^31.

RELATED COMMANDS:

"CH"	Connect to Internet Handles for slaves
"IH"	Set Internet Handles
"LO"	Lock out communication channels
"NA"	Set Number of Axes for Distributed Control System
"QW"	Set Slave Data Record Update Rate

EXAMPLES:

IHA=10,0,0,12	Configures handle A to be connected to a controller with IP address 10.0.0.12
SAA="KI",1,2	Sends the command to handle A (slave controller): KI 1,2
WT5	
SAA="KI?,?"	Sends the command to handle A (slave controller): KI?,?
MG _SAA0	Display the content of the operand _SAA (first response to KI?,? command)
: 1	
MG _SAA1	Display the content of the operand _SAA (2nd response to KI?,? command)
: 2	

Note: The SA command does not wait for a response from the slave controller before continuing code execution. Therefore, a WTxx is required between two SA commands or between an SA command

and querying the response using _SAHn. There is a 38 character maximum string length for the SA command. It is helpful for timing to keep the SA command query as short as possible.

SB

FUNCTION: Set Bit

DESCRIPTION:

The SB command sets an output bit high. SB can be used to set outputs of extended I/O (when the I/O has been configured to operate as outputs). A master controller to set the outputs on slave controllers and IOC-7007 controllers when used in a distributed control system can also use SB.

ARGUMENTS: SB n where

n is an integer corresponding to a specific output on the controller to be set (set to 1). The first output on the controller is denoted as output 1.

The outputs of the slave devices are calculated using the following formula:

n = (HandleNum*100) + (Bitnum)

HandleNum is the number associated with the handle specifier for the particular slave. 1 for handle A, 2 for handle B, etc.

BitNum is the specific output of the slave device. This includes extended I/O that has been configured to operate as outputs.

The outputs on an IOC-7007 I/O controller may also be set through the master controller. The outputs for the IOC-7007's IOM modules are calculated using the following formula:

n = ((HandleNum*1000) + (Bitnum) where

HandleNum is the number associated with the handle specifier for the particular IOC controller. 1 for handle A, 2 for handle B, etc.

Bitnum is the specific output bit on the IOC controller to be set or cleared.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	--
Default Format	--

RELATED COMMAND

"CB"	Clear Bit
"OB"	Set Output on Condition
"OP"	Set Output Port

EXAMPLES:

SB 5	Set output line 5
SB 1	Set output line 1

SC

FUNCTION: Stop Code

DESCRIPTION:

The SC command allows the user to determine why a motor stops. The controller responds with the stop code as follows:

CODE	MEANING	CODE	MEANING
0	Motors are running, independent mode	9	Stopped after Finding Edge (FE)
1	Motors decelerating or stopped at commanded independent position	10	Stopped after homing (HM)
2	Decelerating or stopped by FWD limit switch or soft limit FL	11	Stopped by Selective Abort Input
3	Decelerating or stopped by REV limit switch or soft limit BL	50	Contour running
4	Decelerating or stopped by Stop Command (ST)	51	Contour Stop
6	Stopped by Abort input	99	MC timeout
7	Stopped by Abort command (AB)	100	Motors are running, vector sequence
8	Decelerating or stopped by Off-on-Error (OE1)	101	Motors stopped at commanded vector

ARGUMENTS: SC aaaaaaaa where

a is A,B,C,D,E,F,G or H or any combination to specify the axis or axes

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	-
Default Format	3.0

OPERAND USAGE:

_SCa contains the value of the stop code for the specified axis.

EXAMPLES:

Tom =_SCD	Assign the Stop Code of D to variable Tom

SH

FUNCTION: Servo Here

DESCRIPTION:

The SH commands tells the controller to use the current motor position as the command position and to enable servo control here.

This command can be useful when the position of a motor has been manually adjusted following a motor off (MO) command.

ARGUMENTS: SH aaaaaaaa where

a is A,B,C,D,E,F,G or H or any combination to specify the axis or axes

USAGE:

While Moving	No
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

"MO"	Motor-off

EXAMPLES:

SH	Servo A,B,C,D motors
SHA	Only servo the A motor, the B,C and D motors remain in its previous state.
SHB	Servo the B motor; leave the A,C and D motors unchanged
SHC	Servo the C motor; leave the A,B and D motors unchanged
SHD	Servo the D motor; leave the A,B and C motors unchanged

Note: *The SH command changes the coordinate system. Therefore, all position commands given prior to SH must be repeated. Otherwise, the controller produces incorrect motion.*

@SIN[n]

FUNCTION: Sine

DESCRIPTION:

Returns the sine of the given angle in degrees

ARGUMENTS: @SIN[n] where

n is a signed number in degrees in the range of -32768 to 32767, with a fraction resolution of 16-bit..

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

@ASIN	Arc sine
@COS	Cosine
@ATAN	Arc tangent
@ACOS	Arc cosine
@TAN	Tangent

EXAMPLES:

```
:MG @SIN[0]
 0.0000
:MG @SIN[90]
 1.0000
:MG @SIN[180]
 0.0000
:MG @SIN[270]
 -1.0000
:MG @SIN[360]
 0.0000
:
```

SL

FUNCTION: Single Step

DESCRIPTION:

For debugging purposes. Single Step through the program after execution has paused at a breakpoint (BK). Optional argument allows user to specify the number of lines to execute before pausing again. The BK command resumes normal program execution.

ARGUMENTS: SL n where

n is an integer representing the number of lines to execute before pausing again

USAGE:

While Moving	Yes
In a Program	No
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	1

RELATED COMMANDS:

"BK"	Breakpoint
"TR"	Trace

EXAMPLES:

BK 3	Pause at line 3 (the 4th line) in thread 0
BK 5	Continue to line 5
SL	Execute the next line
SL 3	Execute the next 3 lines
BK	Resume normal execution

SP

FUNCTION: Speed

DESCRIPTION:

This command sets the slew speed of any or all axes for independent moves.

Note: Negative values will be interpreted as the absolute value.

ARGUMENTS: SP n,n,n,n,n,n,n,n or SPA=n where

n is an unsigned even integer in the range 0 to 12,000,000 for servo motors. The units are encoder counts per second.

OR

n is an unsigned number in the range 0 to 3,000,000 for stepper motors

n = ? Returns the speed for the specified axis.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	25000
Default Format	Position Format

OPERAND USAGE:

_SPa contains the speed for the specified axis.

RELATED COMMANDS:

"AC"	Acceleration
"DC"	Deceleration
“PA”	Position Absolute
"PR"	Position Relation
"BG"	Begin

EXAMPLES:

PR 2000,3000,4000,5000	Specify A,B,C,D parameter
SP 5000,6000,7000,8000	Specify A,B,C,D speeds
BG	Begin motion of all axes
AM C	After C motion is complete

Note: *For vector moves, use the vector speed command (VS) to change the speed. SP is not a "mode" of motion like JOG (JG).*

Note: *SP2 is the minimum non-zero speed.*

@SQR[n]

FUNCTION: Square Root

DESCRIPTION:

Takes the square root of the given number. If the number is negative, the absolute value is taken first.

ARGUMENTS: @SQR[n] where

n is a signed number in the range -2147483648 to 2147483647.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

@ABS	Absolute value

EXAMPLES:

```
:MG @SQR[2]
 1.4142
:MG @SQR[-2]
 1.4142
  :
```

ST

FUNCTION: Stop

DESCRIPTION:

The ST command stops motion on the specified axis. Motors will come to a decelerated stop.

If ST is sent from the host without an axis specification, program execution will stop in addition to motion.

ARGUMENTS: ST aaaaaaaaaa where

a is A,B,C,D,E,F,G,H,S or any combination to specify the axis or sequence. If the specific axis or sequence is specified, program execution will not stop.

No argument will stop motion on all axes and stop any programs that are executing.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

"BG"	Begin Motion
"AB"	Abort Motion
"DC"	Deceleration rate

EXAMPLES:

ST A	Stop A-axis motion
ST S	Stop coordinated sequence
ST ABCD	Stop A,B,C,D motion
ST	Stop ABCD motion
ST SCD	Stop coordinated sequence, and C and D motion

Hint: *Use the after motion complete command, AM, to wait for motion to be stopped.*

@TAN[n]

FUNCTION: Tangent

DESCRIPTION:

Returns the tangent of the given angle in degrees

ARGUMENTS: @TAN[n] where

n is a signed number in degrees in the range of -32768 to 32768, with a fractional resolution of 16-bit.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL**

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

@ASIN	Arc sine
@COS	Cosine
@ATAN	Arc tangent
@ACOS	Arc cosine
@SIN	Tangent

EXAMPLES:

```
:MG @TAN[-90]
 -2147483647.0000
:MG @TAN[0]
 0.0000
:MG @TAN[90]
 2147483647.0000
  :
```

TB

FUNCTION: Tell Status Byte

DESCRIPTION:

The TB command returns status information from the controller as a decimal number. Each bit of the status byte denotes the following condition when the bit is set (high):

BIT	STATUS
Bit 7	Executing application program
Bit 6	N/A
Bit 5	Contouring
Bit 4	Executing error or limit switch routine
Bit 3	Input interrupt enabled
Bit 2	Executing input interrupt routine
Bit 1	N/A
Bit 0	Echo on

ARGUMENTS:

TB ? returns the status byte

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	-
Default Format	1.0

OPERAND USAGE:

_TB Contains the status byte

EXAMPLES:

TB	Tell status information from the controller
65	Executing program and Echo is on ($2^6 + 2^0 = 64 + 1 = 65$)

TC

FUNCTION: Tell Error Code

DESCRIPTION:

The TC command returns a number between 1 and 255. This number is a code that reflects why a command was not accepted by the controller. This command is useful when the controller halts execution of a program at a command or when the response to a command is a question mark. The TC command will provide the user with a diagnostic tool. After TC has been read, the error code is set to zero.

ARGUMENTS: TC n where

n = 0 Returns code only

n = 1 Returns code and message

n = ? Returns the error code

No argument will provide the error code for all axes

CODE	EXPLANATION	CODE	EXPLANATION
1	Unrecognized command	59	Mismatched parentheses
2	Command only valid from program	60	Download error - line too long or too many lines
3	Command not valid in program	61	Duplicate or bad label
4	Operand error	62	Too many labels
5	Input buffer full	63	IF statement without ENDIF
6	Number out of range	65	IN command must have a comma
7	Command not valid while running	66	Array space full
8	Command not valid when not running	67	Too many arrays or variables
9	Variable error	68	Not valid from USB Port
10	Empty program line or undefined label	71	IN only valid in task #0
11	Invalid label or line number	80	Record mode already running
12	Subroutine more than 16 deep	81	No array or source specified
13	JG only valid when running in jog mode	82	Undefined Array
14	EEPROM check sum error	83	Not a valid number
15	EEPROM write error	84	Too many elements
16	IP incorrect sign during position move or IP given during forced deceleration	90	Only X Y Z W valid operand
17	ED, BN and DL not valid while program running	96	SM jumper needs to be installed for stepper motor operation
18	Command not valid when contouring	97	Bad Binary Command Format
19	Application strand already executing	98	Binary Commands not valid in application program
20	Begin not valid with motor off	99	Bad binary command number
21	Begin not valid while running	100	Not valid when running ECAM
22	Begin not possible due to Limit	101	Improper index into ET

	Switch		(must be 0-256)
24	Begin not valid because no sequence defined	102	No master axis defined for ECAM
25	Variable not given in IN command	103	Master axis modulus greater than 256*EP value
28	S operand not valid	104	Not valid when axis performing ECAM
29	Not valid during coordinated move	105	EB1 command must be given first
30	Sequence segment too short	110	No hall effect sensors detected
31	Total move distance in a sequence > 2 billion	111	Must be made brushless by BA command
32	More than 511 segments in a sequence	112	BZ command timeout
33	VP or CR commands cannot be mixed with LI commands	113	No movement in BZ command
41	Contouring record range error	114	BZ command runaway
42	Contour data being sent too slowly	118	Controller has GL1600 not GL1800
46	Gear axis both master and follower	120	Bad Ethernet transmit
50	Not enough fields	121	Bad Ethernet packet received
51	Question mark not valid	122	Ethernet input buffer overrun
52	Missing " or string too long	123	TCP lost sync
53	Error in {}	124	Ethernet handle already in use
54	Question mark part of string	125	No ARP response from IP address
55	Missing [or []	126	Closed Ethernet handle
56	Array index invalid or out of range	127	Illegal Modbus function code
57	Bad function or array	128	IP address not valid
58	Not a valid Command Operand (i.e._GNX)	129	HC already executed
		130	Serial port hardware handshake timeout

USAGE:

While Moving	Yes
In a Program	Yes
Not in a Program	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	--
Default Format	3.0

USAGE:

_TC contains the error code

EXAMPLES:

:GF32	Bad command
?TC	Tell error code
001	Unrecognized command

#TCPERR

FUNCTION: Ethernet communication error automatic subroutine

DESCRIPTION:

The following error (see TC) occurs when a command such as MG “hello” {EA} is sent to a failed Ethernet connection:

123 TCP lost sync or timeout

This error means that the client on handle A did not respond with a TCP acknowledgement (for example because the Ethernet cable was disconnected). Handle A is closed in this case.

#TCPERR allows the application programmer to run code (for example to reestablish the connection) when error 123 occurs.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	No
Controller Usage	**DMC-21x2/3, 2100, 2200 (not 2000)**

RELATED COMMANDS:

TC	Tell error code
_IA4	Last dropped handle
MG	Print message
SA	Send ASCII command via Ethernet

EXAMPLES:

```
#L
  MG {EA} "L"
  WT1000
JP#L

#TCPERR
  MG {P1} "TCPERR.  Dropped handle", _IA4
RE
```

***NOTE**: An application program must be executing for the automatic subroutine to function, which runs in thread 0.*

***NOTE**: Use RE to end the routine*

TD

FUNCTION: Tell Dual Encoder

DESCRIPTION::

This command returns the current position of the dual (auxiliary) encoder(s). Auxiliary encoders are not available for stepper axes or for the axis where output compare is used.

When operating with stepper motors, the TD command returns the number of counts that have been output by the controller.

ARGUMENTS: TD aaaaaaaa where

a is A,B,C,D,E,F,G or H or any combination to specify the axis or axes

No argument will provide the dual encoder position for all axes

USAGE:

While Moving	Yes
In a Program	Yes
Not in a Program	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	0
Default Format	Position Format

OPERAND USAGE:

_TDA contains value of dual encoder register.

RELATED COMMANDS:

"DE" Dual Encoder

EXAMPLES:

:PF 7	Position format of 7
:TD	Return A,B,C,D Dual encoders
0000200,-0000010,0000000,-0000110	
TDA	Return the A motor Dual encoder
0000200	
DUAL=_TDA	Assign the variable, DUAL, the value of TDA

TE

FUNCTION: Tell Error

DESCRIPTION::

This command returns the current position error of the motor(s). The range of possible error is 2147483647. The Tell Error command is not valid for step motors since they operate open loop.

ARGUMENTS: TE aaaaaaaa where

a is A,B,C,D,E,F,G or H or any combination to specify the axis or axes

No argument will provide the position error for all axes

USAGE:

While Moving	Yes
In a Program	Yes
Not in a Program	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	0
Default Format	Position Format

RELATED COMMANDS:

"OE"	Off On Error
"ER"	Error Limit
#POSERR	Error Subroutine
"PF"	Position Formatting

EXAMPLES:

TE	Return all position errors
00005,-00002,00000,00006	
TEA	Return the A motor position error
00005	
TEB	Return the B motor position error
-00002	
Error =_TEA	Sets the variable, Error, with the A-axis position error

Hint: *Under normal operating conditions with servo control, the position error should be small. The position error is typically largest during acceleration.*

TH

FUNCTION: Tell Handle Status

DESCRIPTION:

The TH command is used to request the controllers' handle status. Data returned from this command indicates the IP address and Ethernet address of the current controller. This data is followed by the status of each handle indicating connection type, IP address and whether it is a QW or Command handle setup by the HC command.

ARGUMENTS: None

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	--
Default Format	--

RELATED COMMANDS:

"IH"	Internet Handle
"HR"	Handle Restore
"WH"	Which Handle

EXAMPLES:

:TH Tell current handle configuration

```
CONTROLLER IP ADDRESS 10,51,0,87 ETHERNET ADDRESS 00-50-4C-08-01-1F
IHA TCP PORT 1050 TO IP ADDRESS 10,51,0,89 PORT 1000 SLAVE CD COMMAND
IHB TCP PORT 1061 TO IP ADDRESS 10,51,0,89 PORT 1001 SLAVE CD QW
IHC TCP PORT 1012 TO IP ADDRESS 10,51,0,93 PORT 1002 SLAVE EF COMMAND
IHD TCP PORT 1023 TO IP ADDRESS 10,51,0,93 PORT 1003 SLAVE EF QW
IHE TCP PORT 1034 TO IP ADDRESS 10,51,0,101 PORT 1004 SLAVE IOC COMMAND
IHF TCP PORT 1045 TO IP ADDRESS 10,51,0,101 PORT 1005 SLAVE IOC QW
IHG AVAILABLE
IHH AVAILABLE
```

TI

FUNCTION: Tell Inputs

DESCRIPTION:

This command returns the state of the inputs including the extended I/O configured as inputs and IOC-7007 modules. The value returned by this command is decimal and represents an 8-bit value (decimal value ranges from 0 to 255). Each bit represents one input where the LSB is the lowest input number and the MSB is the highest input bit.

ARGUMENTS: TIn where

n = 0 Return Input Status for Inputs 1 through 7 of the master controller

n = 2 through 9 [see note 3]

where n represents the extended inputs ranging from (8*n)+1 through (8*(n+1))

For slave inputs, n = (HandleNum * 100) + InputBank where HandleNum is the numeric value of the slave handle (1 – 8) and InputBank is the bank to be read (0 – 9 as shown above).

For IOC-7007 inputs, n = (HandleNum * 1000) + SlotNum where Handlenum is the numeric value of the IOC-7007 handle (1 – 8) and SlotNum is the IOM input slot to be read (0 – 6). Please note for the IOM-70016 TTL input module, 16 bits of data will be returned for a value of 0 – 65535.

no argument will return the Input Status for Inputs 1 through 7 of the master controller

n = ? returns the Input Status for Inputs 1 through 7 of the master controller

note 3 These arguments only apply when using DB-14064 extended I/O configured as inputs

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	--
Default Format	1.0

OPERAND USAGE:

_TIn contains the status byte of the input block specified by ‘n’. Note that the operand can be masked to return only specified bit information - see section on Bit-wise operations.

EXAMPLES:

TI	
08	Input 4 is high, others low
TI602	
00	All inputs low on slave handle F, extended input block 2
Input =_TI	Sets the variable, Input, with the TI value
TI8001	
255	All inputs high on IOC-7007 at handle H, slot number 1

TIME*

FUNCTION: Time Operand (Keyword)

DESCRIPTION:

*The TIME operand returns the value of the internal free running, real time clock. The returned value represents the number of servo loop updates and is based on the TM command. The default value for the TM command is 1000. With this update rate, the operand TIME will increase by 1 count every update of approximately 1000usec. Note that a value of 1000 for the update rate (TM command) will actually set an update rate of 976 microsecond. Thus the value returned by the TIME operand will be off by 2.4% of the actual time.

The clock is reset to 0 with a standard reset or a master reset.

The keyword, TIME, does not require an underscore "_" as does the other operands.

EXAMPLES:

MG TIME	Display the value of the internal clock

TL

FUNCTION: Torque Limit

DESCRIPTION:

The TL command sets the limit on the motor command output. For example, TL of 5 limits the motor command output to 5 volts. Maximum output of the motor command is 9.998 volts.

ARGUMENTS: TL n,n,n,n,n,n,n,n or TLA=n where

n is an unsigned numbers in the range 0 to 9.998 volts with resolution of 0.0003 volts

n = ? Returns the value of the torque limit for the specified axis.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	-
Default Format	1.0

OPERAND USAGE:

_TLa contains the value of the torque limit for the specified axis.

EXAMPLES:

TL 1,5,9,7.5	Limit A-axis to 1 volt Limit B-axis to 5 volts Limit C-axis to 9 volts Limit D-axis to 7.5 volts
TL ?,?,?,?	Return limits
1.0000,5.0000,9.0000,	
7.5000	
TL ?	Return A-axis limit
1.0000	

TM

FUNCTION: Update Time

DESCRIPTION:

The TM command sets the sampling period of the control loop. Changing the sampling period will un-calibrate the speed and acceleration parameters. A negative number turns off the servo loop. The units of this command are μsec.

ARGUMENTS: TM n where

Using normal firmware the minimum sample time for the DMC-3425 is 375 usec and 250 usec for the DMC-3415.

n = ? returns the value of the sample time.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	-
Default Format	1.0

OPERAND USAGE:

_TM contains the value of the sample time.

EXAMPLES:

TM -1000	Turn off internal clock
TM 2000	Set sample rate to 2000 [EQN "[mu]"]sec (This will cut all speeds in half and all acceleration in fourths)
TM 1000	Return to default sample rate

TP

FUNCTION: Tell Position

DESCRIPTION:

This command returns the current position of the motor(s).

ARGUMENTS: TP aaaaaaaa where

a is A,B,C,D,E,F,G or H or any combination to specify the axis or axes

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	-
Default Format	--

OPERAND USAGE:

_TPa contains the current position value for the specified axis.

No argument will provide the encoder position for all axes

RELATED COMMANDS:

"PF" Position Formatting

EXAMPLES:

Assume the A-axis is at the position 200 (decimal), the B-axis is at the position -10 (decimal), the C-axis is at position 0, and the D-axis is at -110 (decimal). The returned parameter units are in quadrature counts.

:PF 7	Position format of 7
:TP	Return A,B,C,D positions
0000200,-0000010,0000000,-0000110	
TPA	Return the A motor position
0000200	
TPB	Return the B motor position
-0000010	
PF-6.0	Change to hex format
TP	Return A,B,C,D in hex
$0000C8,$FFFFF6,$000000,$FFFF93	
Position =_TPA	Assign the variable, Position, the value of TPA

TR

FUNCTION: Trace

DESCRIPTION:

The TR command causes each instruction in a program to be sent out the communications port prior to execution. TR1 enables this function and TR0 disables it. The trace command is useful in debugging programs.

ARGUMENTS: TR n where

n = 0 Disables the trace function

n = 1 Enables the trace function

No argument disables the trace function

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	TR0
Default Format	--

TS

FUNCTION: Tell Switches

DESCRIPTION:

TS returns status information of the Home switch, Forward Limit switch Reverse Limit switch, error conditions, motion condition and motor state. The value returned by this command is decimal and represents an 8 bit value (decimal value ranges from 0 to 255). Each bit represents the following status information:

BIT	STATUS
Bit 7	Axis in motion if high
Bit 6	Axis error exceeds error limit if high
Bit 5	A motor off if high
Bit 4	Undefined
Bit 3	Forward Limit Switch Status inactive if high
Bit 2	Reverse Limit Switch Status inactive if high
Bit 1	Home A Switch Status
Bit 0	Latched

Note: For active high or active low configuration (CN command), these bits are '1' when the switch is inactive and '0' when active.

ARGUMENTS: TS aaaaaaaa where

a is A,B,C,D,E,F,G or H or any combination to specify the axis or axes

No argument will provide the status for all axes

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	-
Default Format	3.0

OPERAND USAGE:

_TS contains the current status of the switches.

EXAMPLES:

V1=_TSB	Assigns value of TSB to the variable V1
V1=	Interrogate value of variable V1
015 (returned value)	Decimal value corresponding to bit pattern 00001111 Y axis not in motion (bit 7 - has a value of 0) Y axis error limit not exceeded (bit 6 has a value of 0) Y axis motor is on (bit 5 has a value of 0) Y axis forward limit is inactive (bit 3 has a value of 1) Y axis reverse limit is inactive (bit 2 has a value of 1) Y axis home switch is high (bit 1 has a value of 1) Y axis latch is not armed (bit 0 has a value of 1)

TT

FUNCTION: Tell Torque

DESCRIPTION:

The TT command reports the value of the analog output signal, which is a number between -9.998 and 9.998 volts.

ARGUMENTS: TT aaaaaaaa where

a is A,B,C,D,E,F,G or H or any combination to specify the axis or axes

No argument will provide the torque for all axes

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	-
Default Format	1.4

OPERAND USAGE:

_TTa contains the value of the torque for the specified axis.

RELATED COMMANDS:

"TL"	Torque Limit

EXAMPLES:

V1=_TTA	Assigns value of TTA to variable, V1
TTA	Report torque on A
-0.2843	Torque is -.2843 volts

TV

FUNCTION: Tell Velocity

DESCRIPTION:

The TV command returns the actual velocity of the axes in units of encoder count/s. The value returned includes the sign.

ARGUMENTS: TV aaaaaaaa where

a is A,B,C,D,E,F,G or H or any combination to specify the axis or axes

No argument will provide the auxiliary encoder velocity for all axes

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	-
Default Format	7.0

OPERAND USAGE:

_TVa contains the value of the velocity for the specified axis.

EXAMPLES:

VELA=_TVA	Assigns value of A-axis velocity to the variable VELA
TVB	Returns the B-axis velocity
0003420	

Note: *The TV command is computed using a special averaging filter (over approximately .25 sec). Therefore, TV will return average velocity, not instantaneous velocity.*

TW

FUNCTION: Timeout for IN-Position (MC)

DESCRIPTION:

The TW command sets the timeout in msec to declare an error if the MC command is active and the motor is not at or beyond the actual position within n msec after the completion of the motion profile. If a timeout occurs, then the MC trippoint will clear and the stopcode will be set to 99. An application program will jump to the special label #MCTIME. The RE command should be used to return from the #MCTIME subroutine.

ARGUMENTS: TW n,n,n,n,n,n,n,n or TWA=n where

n specifies the timeout in msec. n ranges from 0 to 32767 msec

n = -1 Disables the timeout.

n = ? Returns the timeout in msec for the MC command for the specified axis.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	32766
Default Format	

OPERAND USAGE:

_TWa contains the timeout in msec for the MC command for the specified axis.

RELATED COMMANDS:

"MC" Motion Complete trippoint

TZ

FUNCTION: Tell I/O Status

DESCRIPTION:

The TZ command is used to request the master or slave I/O status. This is returned to the user as a text string.

ARGUMENTS: TZh where

h (A through H) is the handle of the data communication channel to the slave

No argument returns the master's I/O status.

* returns master and all slave I/O

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	--
Default Format	--

RELATED COMMANDS:

"TI"	Tell Inputs
"SB"	Set output bit
"OP"	Output port
"CB"	Clear output bit

EXAMPLES:

:TZC Tell current slave I/O status on handle C

Block 300 (307-301) dedicated as input - value 127 (111_1111)

Block 300 (303-301) dedicated as output - value 0 (000)

UL

FUNCTION: Upload

DESCRIPTION:

The UL command transfers data from the controller to a host computer through port 1. Programs are sent without line numbers. The Uploaded program will be followed by a <control>Z or a \ as an end of text marker.

ARGUMENTS: None

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	0
In a Program	No	Default Format	-
Command Line	Yes		
Controller Usage	**ALL CONTROLLERS**		

OPERAND USAGE:

When used as an operand, _UL gives the number of available variables. The number of available variables is 254.

RELATED COMMAND:

"DL"	Download

EXAMPLES:

UL;	Begin upload
#A	Line 0
NO This is an Example	Line 1
NO Program	Line 2
EN	Line 3
<cntrl>Z	Terminator

VA

FUNCTION: Vector Acceleration

DESCRIPTION:

This command sets the acceleration rate of the vector in a coordinated motion sequence.

ARGUMENTS: VA s where

s is an unsigned integer in the range 1024 to 68,431,360. The parameter input will be rounded down to the nearest factor of 1024. The units of the parameter is counts per second squared.

s = ? Returns the value of the vector acceleration for the S coordinate plane.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS, LOCAL AXES ONLY**

DEFAULTS:

Default Value	262144
Default Format	Position Format

OPERAND USAGE:

_VA contains the value of the vector acceleration for the specified axis.

RELATED COMMANDS:

"VS"	Vector Speed
"VP"	Vector Position
"VE"	End Vector
"CR"	Circle
"VM"	Vector Mode
"BG"	Begin Sequence
"VD"	Vector Deceleration
"VT"	Vector smoothing constant - S-curve

EXAMPLES:

VA 1024	Set vector acceleration to 1024 counts/sec^2
VA ?	Return vector acceleration
00001024	
VA 20000	Set vector acceleration
VA ?	
0019456	Return vector acceleration
ACCEL=_VA	Assign variable, ACCEL, the value of VA

VD

FUNCTION: Vector Deceleration

DESCRIPTION:

This command sets the deceleration rate of the vector in a coordinated motion sequence.

ARGUMENTS: VD s where

s is an unsigned integer in the range 1024 to 68431360. The parameter input will be rounded down to the nearest factor of 1024. The units of the parameter is counts per second squared.

s = ? Returns the value of the vector deceleration for the S coordinate plane.

USAGE:

While Moving	No
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS, LOCAL AXES ONLY**

DEFAULTS:

Default Value	262144
Default Format	Position Format

OPERAND USAGE:

_VD contains the value of the vector deceleration.

RELATED COMMANDS:

"VA"	Vector Acceleration
"VS"	Vector Speed
"VP"	Vector Position
"CR"	Circle
"VE"	Vector End
"VM"	Vector Mode
"BG"	Begin Sequence
"VT"	Smoothing constant - S-curve

EXAMPLES:

#VECTOR	Vector Program Label
VMAB	Specify plane of motion
VA1000000	Vector Acceleration
VD 5000000	Vector Deceleration
VS 2000	Vector Speed
VP 10000, 20000	Vector Position
VE	End Vector
BGS	Begin Sequence

VE

FUNCTION: Vector Sequence End

DESCRIPTION:

VE is required to specify the end segment of a coordinated move sequence. VE would follow the final VP or CR command in a sequence. VE is equivalent to the LE command.

ARGUMENTS: VE n

No argument specifies the end of a vector sequence

n = ? Returns the length of the vector in counts.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS, LOCAL AXES ONLY**

DEFAULTS:

Default Value	-
Default Format	-

OPERAND USAGE:

_VE contains the length of the vector in counts.

RELATED COMMANDS:

"VM"	Vector Mode
"VS"	Vector Speed
"VA"	Vector Acceleration
"VD"	Vector Deceleration
"CR"	Circle
"VP"	Vector Position
"BG"	Begin Sequence
"CS"	Clear Sequence

EXAMPLES:

VM AB	Vector move in AB
VP 1000,2000	Linear segment
CR 0,90,180	Arc segment
VP 0,0	Linear segment
VE	End sequence
BGS	Begin motion

VF

FUNCTION: Variable Format

DESCRIPTION:

The VF command formats the number of digits to be displayed when interrogating the controller.

If a number exceeds the format, the number will be displayed as the maximum possible positive or negative number (i.e. 999.99, -999, $8000 or $7FF).

ARGUMENTS: VF m.n where

m and n are unsigned numbers in the range 0<m<10 and 0<n<4.

m represents the number of digits before the decimal point. A negative m specifies hexadecimal format. When in hexadecimal, the string will be preceded by a $ and Hex numbers are displayed as 2's complement with the first bit used to signify the sign.

n represents the number of digits after the decimal point.

m = ? Returns the value of the format for variables and arrays.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	10.4
Default Format	2.1

OPERAND USAGE:

_VF contains the value of the format for variables and arrays.

RELATED COMMANDS:

"PF"	Position format

EXAMPLES:

VF 5.3	Sets 5 digits of integers and 3 digits after the decimal point
VF 8.0	Sets 8 digits of integers and no fractions
VF -4.0	Specify hexadecimal format with 4 bytes to the left of the decimal

VM

FUNCTION: Coordinated Motion Mode

DESCRIPTION:

The VM command specifies the coordinated motion mode and the plane of motion. This mode may be specified for motion on any set of two axes.

The motion is specified by the instructions VP and CR, which specify linear and circular segments. Up to 511 segments may be given before the Begin Sequence (BGS) command. Additional segments may be given during the motion when the buffer frees additional spaces for new segments. It is the responsibility of the user to keep enough motion segments in the buffer to ensure continuous motion.

The Vector End (VE) command must be given after the last segment. This allows the controller to properly decelerate.

ARGUMENTS: VM n,m where

n and m specify plane of vector motion and can be any two axes. Vector Motion can be specified for one axis by specifying 2nd parameter, m, as N. Specifying one axis is useful for obtaining sinusoidal motion on 1 axis.

n = ? Returns the available spaces for motion segments that can be sent to the buffer. A value of zero means that the buffer is full and no additional segments may be sent.

USAGE:

While Moving	No
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS, LOCAL AXES ONLY**

DEFAULTS:

Default Value	A,B
Default Format	-

OPERAND USAGE:

_VM contains instantaneous commanded vector velocity.

RELATED COMMANDS:

"VP"	Vector Position
"VS"	Vector Speed
"VA"	Vector Acceleration
"VD"	Vector Deceleration
"CR"	Circle
"VE"	End Vector Sequence
"CS"	Clear Sequence
"VT"	Vector smoothing constant -- S-curve
"AV"	Trippoint for Vector distance

EXAMPLES:

VM A,B	Specify coordinated mode for A,B
CR 500,0,180	Specify arc segment
VP 100,200	Specify linear segment
VE	End vector
BGS	Begin sequence

VP

FUNCTION Vector Position

DESCRIPTION:

The VP command defines the target coordinates of a straightline segment in a 2 axis motion sequence which have been selected by the VM command. The units are in quadrature counts, and are a function of the vector scale factor set using the command VS.

For three or more axes linear interpolation, use the LI command.

ARGUMENTS: VP n,m < o > p where

n and m are signed integers in the range -2147483648 to 2147483647 The length of each segment must be limited to $8 \cdot 10^6$. The values for n and m will specify a coordinate system from the beginning of the sequence.

o specifies a vector speed to be taken into effect at the execution of the vector segment. n is an unsigned even integer between 0 and 12,000,000 for servo motor operation and between 0 and 3,000,000 for stepper motors.

p specifies a vector speed to be achieved at the end of the vector segment. p is an unsigned even integer between 0 and 8,000,000.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS, LOCAL AXES ONLY**

DEFAULTS:

Default Value	-
Default Format	-

OPERAND USAGE:

_VP contains the absolute coordinate of the axes at the last intersection along the sequence. For example, during the first motion segment, this instruction returns the coordinate at the start of the sequence. The use as an operand is valid in the linear mode, LM, and in the Vector mode, VM.

RELATED COMMANDS:

"CR"	Circle
"VM"	Vector Mode
"VA"	Vector Acceleration
"VD"	Vector Deceleration
"VE"	Vector End
"VS"	Vector Speed
"BG"	Begin Sequence
"VT"	Vector smoothing

EXAMPLES:

#A	Program A
VM A,B	Specify motion plane
VP 1000,2000	Specify vector position A,B
CR 1000,0,360	Specify arc
VE	Vector end
VS 2000	Specify vector speed
VA 400000	Specify vector acceleration
BGS	Begin motion sequence
EN	End Program

Hint: *The first vector in a coordinated motion sequence defines the origin for that sequence. All other vectors in the sequence are defined by their endpoints with respect to the start of the move sequence.*

VR

FUNCTION: Vector Speed Ratio

DESCRIPTION:

The VR sets a ratio to be used as a multiplier of the current vector speed. The vector speed can be set by the command VS or the operators < and > used with CR, VP and LI commands. VR takes effect immediately and will ratio all the following vector speed commands. VR doesn't ratio acceleration or deceleration, but the change in speed is accomplished by accelerating or decelerating at the rate specified by VA and VD.

ARGUMENTS: VR s where

s is between 0 and 10 with a resolution of .0001. The value specified by s is the vector ratio to apply to the S coordinate system.

s = ? Returns the value of the vector speed ratio for the S coordinate plane.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS, LOCAL AXES ONLY**

DEFAULTS:

Default Value	1
Default Format	-

OPERAND USAGE:

_VR contains the vector speed ratio of the S coordinate system

RELATED COMMANDS:

"VS"	Vector Speed

EXAMPLES:

#A	Vector Program
VMAB	Vector Mode
VP 1000,2000	Vector Position
CR 1000,0,360	Specify Arc
VE	End Sequence
VS 2000	Vector Speed
BGS	Begin Sequence
AMS	After Motion
JP#A	Repeat Move
#SPEED	Speed Override
VR@AN[1]*.1	Read analog input compute ratio
JP#SPEED	Loop
XQ#A,0;	Execute task 0 and 1 simultaneously

Note: *VR is useful for feedrate override, particularly when specifying the speed of individual segments using the operator '<' and '>'.*

VS

FUNCTION: Vector Speed

DESCRIPTION:

The VS command specifies the speed of the vector in a coordinated motion sequence in either the LM or VM modes. VS may be changed during motion.

Vector Speed can be calculated by taking the square root of the sum of the squared values of speed for each axis specified for vector or linear interpolated motion.

ARGUMENTS: VS s where

s is an unsigned even number in the range 2 to 12,000,000 for servo motors and 2 to 3,000,000 for stepper motors. The units are counts per second.

s = ? Returns the value of the vector speed for the S coordinate plane.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS, LOCAL AXES ONLY**

DEFAULTS:

Default Value	8192
Default Format	-

OPERAND USAGE:

_VS contains the vector speed of the S coordinate system.

RELATED COMMANDS:

"VA"	Vector Acceleration
"VP"	Vector Position
"CR"	Circle
"LM"	Linear Interpolation
"VM"	Vector Mode
"BG"	Begin Sequence
"VE"	Vector End

EXAMPLES:

VS 2000	Define vector speed of S coordinate system
VS ?	Return vector speed of S coordinate system
002000	

Hint: *Vector speed can be attached to individual vector segments. For more information, see description of VP, CR, and LI commands.*

VT

FUNCTION: Vector Time Constant - S curve

DESCRIPTION:

The VT command filters the acceleration and deceleration functions in vector moves of VM, LM type to produce a smooth velocity profile. The resulting profile, known as Smoothing, has continuous acceleration and results in reduced mechanical vibrations. VT sets the bandwidth of the filter, where 1 means no filtering and 0.004 means maximum filtering. Note that the filtering results in longer motion time.

ARGUMENTS: VT s where

s is an unsigned number in the range between 0.004 and 1.0, with a resolution of 1/256.

s = ? Returns the value of the vector time constant for the S coordinate plane.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS, LOCAL AXES ONLY**

DEFAULTS:

Default Value	1.0
Default Format	1.4

OPERAND USAGE:

_VT contains the vector time constant.

RELATED COMMANDS:

"IT"	Independent Time Constant for smoothing independent moves

EXAMPLES:

VT 0.8	Set vector time constant for S coordinate system
VT ?	Return vector time constant for S coordinate system
0.8	

WC

FUNCTION: Wait for Contour Data

DESCRIPTION:

The WC command acts as a flag in the Contour Mode. After this command is executed, the controller does not receive any new data until the internal contour data buffer is ready to accept new commands. This command prevents the contour data from overwriting on itself in the contour data buffer.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS, LOCAL AXES ONLY**

DEFAULTS:

Default Value	1.0
Default Format	1.4

RELATED COMMANDS:

"CM"	Contour Mode
"CD"	Contour Data
"DT"	Contour Time

EXAMPLES:

CM ABCD	Specify contour mode
DT 4	Specify time increment for contour
CD 200,350,-150,500	Specify incremental position on A,B,C and D. A-axis moves 200 counts B-axis moves 300 counts C-axis moves -150 counts D-axis moves 500 counts
WC	Wait for contour data to complete
CD 100,200,300,400	
WC	Wait for contour data to complete
DT 0	Stop contour
CD 0,0,0,0	Exit mode

WH

FUNCTION: Which Handle

DESCRIPTION:

The WH command is used to identify the handle in which the command is executed. The command returns IHA through IHH to indicate on which handle the command was executed. The command returns RS232 if communicating serially.

ARGUMENTS: None

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	--
Default Format	--

RELATED COMMANDS:

"TH"	Tell Handle
"IH"	Internet Handle

OPERAND USAGE:

_WH contains the numeric representation of the handle in which a command is executed. Handles A through H are indicated by the value 0 – 7, while a –1 indicates the serial port.

EXAMPLES:

:WH	Request handle identification
IHC	Command executed in handle C
:WH	Request handle identification
RS232	Command executed in RS232 port

WT

FUNCTION: Wait

DESCRIPTION:

The WT command is a trippoint used to time events. After this command is executed, the controller will wait for the number of samples specified before executing the next command. If the TM command has not been used to change the sample rate from 1 msec, then the units of the Wait command are milliseconds.

ARGUMENTS: WT n where

n is an integer in the range 0 to 2 Billion decimal

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	-
Command Line	Yes		
Controller Usage	**ALL CONTROLLERS**		

EXAMPLES: Assume that 10 seconds after a move is over a relay must be closed.

#A	Program A
PR 50000	Position relative move
BGA	Begin the move
AMA	After the move is over
WT 10000	Wait 10 seconds
SB 0	Turn on relay
EN	End Program

Hint: *To achieve longer wait intervals, just stack multiple WT commands.*

XQ

FUNCTION: Execute Program

DESCRIPTION:

The XQ command begins execution of a program residing in the program memory of the controller. Execution will start at the label or line number specified. Up to 2 programs may be executed with the controller.

ARGUMENTS: XQ #A,n XQm,n where

A is a program name of up to seven characters.

m is a line number

n is an integer representing the thread number for multitasking

n is an integer in the range of 0 to 1.

NOTE: The arguments for the command, XQ, are optional. If no arguments are given, the first program in memory will be executed as thread 0.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value of n:	0
Default Format	-

OPERAND USAGE:

_XQn contains the current line number of execution for thread n, and -1 if thread n is not running.

RELATED COMMANDS:

"HX"	Halt execution

EXAMPLES:

XQ #Apple,0	Start execution at label Apple, thread zero
XQ #data,1	Start execution at label data, thread one
XQ 0	Start execution at line 0

Hint: *Don't forget to quit the edit mode first before executing a program!*

ZA

FUNCTION: User Variable

DESCRIPTION:

ZA sets one or two user variables for use with a distributed control system. One user variable per local axis is automatically sent as part of the status record from the slave controller to the master controller. These variables provide a method for specific slave information to be passed to the master automatically.

ARGUMENTS: ZA n,n or ZAA=n where

n can be a number, controller operand, variable, mathematical function, or string; The range for numeric values is 4 bytes of integer (2^{31} or –2,147,483,648 to +2,147,483,647). The maximum number of characters for a string is 4 characters. Strings are identified by quotations.

Note: The number of arguments depends on the number of axes. Using a DMC-3425, there are 2 ZA variables. To set both values on the slave controller, the commands would be: ZAA=<value> and ZAB=<value>

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	--
In a Program	Yes	Default Format	--
Command Line	Yes		
Controller Usage	**ALL CONTROLLERS**		

OPERAND USAGE:

_ZAa is called on the master controller and contains the user variable defined by the axis a set with the ZA command on the slave controller. a is any globalaxis designator A,B,C,D,E,F,G and H.

RELATED COMMANDS:

"ZB"	Set third and fourth user variables

EXAMPLES:

ZA 2343,"CAT"	Sets the first user variable to a number (2343) and the second user variable to the string "CAT". This is called on the slave controller. Call _ZAa on the master controller to retrieve the value.

ZB

FUNCTION: User Variable, ZB

DESCRIPTION:

ZB sets one or two user variables for use with a distributed control system. One user variable per local axis is automatically sent as part of the status record from the slave controller to the master controller. These variables provide a method for specific slave information to be passed to the master automatically.

ARGUMENTS: ZB n,n or ZBA=n where

n can be a number, controller operand, variable, mathematical function, or string; The range for numeric values is 4 bytes of integer (2^{31} or –2,147,483,648 to +2,147,483,647). The maximum number of characters for a string is 4 characters. Strings are identified by quotations.

Note: The number of arguments depends on the number of axes. Using a DMC-3425, there are 2 ZB variables. To directly set both values on the slave controller, the commands would be ZBA=<value> and ZBB=<value>

USAGE:

While Moving	Yes	Default Value	--
In a Program	Yes	Default Format	--
Command Line	Yes		
Controller Usage	**ALL CONTROLLERS**		

DEFAULTS:

OPERAND USAGE:

_ZBa is called on the master controller and contains the user variable defined by the axis a set with the ZB command on the slave controller. a is any global axis designator A,B,C,D,E,F,G and H.

RELATED COMMANDS:

"ZA"	Set first and second user variables

EXAMPLES:

ZB "DOG",9999485	Sets the first user variable to the string "DOG" and the second to a number (9999845). This is called on the slave controller. Call _ZBa on the master controller to retrieve the value.

ZS

FUNCTION: Zero Subroutine Stack

DESCRIPTION:

The ZS command is only valid in an application program and is used to avoid returning from an interrupt (either input or error). ZS alone returns the stack to its original condition. ZS1 adjusts the stack to eliminate one return. This turns the jump to subroutine into a jump. Do not use RI (Return from Interrupt) when using ZS. To re-enable interrupts, you must use II command again.

The status of the stack can be interrogated with the operand _ZSn - see operand usage below.

ARGUMENTS: ZS n where

n = 0 Returns stack to original condition

n = 1 Eliminates one return on stack

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	No
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	0
Default Format	3.0

OPERAND USAGE:

_ZSn contains the stack level for the specified thread where n = 0,1,2 or 3. Note: n can also be specified using A (thread 0), B (thread1), C (thread2) or D (thread3) .

EXAMPLES:

II1	Input Interrupt on 1
#A;JP #A;EN	Main program
#ININT	Input Interrupt
MG "INTERRUPT"	Print message
S=_ZS	Interrogate stack
S=	Print stack
ZS	Zero stack
S=_ZS	Interrogate stack
S=	Print stack
EN	End

THIS PAGE WAS LEFT BLANK INTENTIONALLY

Index