DMC-1400 Series

Command Reference

Manual Rev. 2.9

By Galil Motion Control, Inc.

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

Rev 7/08

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

CC	aux serial
CF	unsolicited
CI	interrupt
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
P2CD	port 2 code
P2CH	character
P2NM	number
P2ST	string
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
#COMINT; N1,1	
#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
TK	peak torque
TL	torque limit
TM	sample time
TT	tell torque

ECAM

EA	master axis
EB	enable
EC	counter
EG	engage slave
EM	modulus
EP	master positions
EQ	disengage slave
ET	table

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
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
PT	position tracking
RP	desired position
SP	speed
ST	stop
~a	axis variable

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

VECTOR

AV	wait for arc length
_AVS	arc length
CA	2nd vector
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
TN	tangent scale
_TN	1st position
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

Contents

Overview

Controller Notation

This command reference is a supplement to the Galil Motion Control User Manual. For proper controller operation, consult the Users Manual. This manual describes commands to be used with the Galil Econo Series Motion Controllers: DMC-1410, DMC-1411, DMC-1412, DMC-1414, DMC-1415, DMC-1416, DMC-1417, and DMC-1425. Commands are listed in alphabetical order.

This command summary includes all executable commands, which can be used with the DMC-1400 series motion controller. These commands are common to all the controllers in that series with certain exceptions. These exceptions are noted on each corresponding command as “Controller Usage”. An example is Ethernet commands for the DMC-1415, DMC-1416, and DMC-1425.

Servo and Stepper Motor Notation:

Your motion controller has been designed to work with both servo and stepper type motors. Installation and system setup will vary depending upon whether the controller will be used with stepper motors, or servo motors. To make finding the appropriate instructions faster and easier, icons will be next to any information that applies exclusively to one type of system. Otherwise, assume that the instructions apply to all types of systems. The icon legend is shown below.

⬭	Attention: Pertains to servo motor use.
⎍	Attention: Pertains to stepper motor use.

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 left corner. Below the Opcode is a description of the command and required arguments.

Arguments

As arguments, some commands require actual values to be specified following the instruction. These commands are followed by lower case n where n is replaced by an actual value.

A “?” returns the specified value for that axis. For example, AC? returns the acceleration of the axis.

Other commands require action on the axis to be specified. These commands do not have an operand action for the axis or are specified by writing the command only, such as BG or ST. When downloading commands to the DMC-141X, do not insert a space prior to any command. For example, ST; AM is invalid because there is a space after the semicolon.

The DMC-1425 is the only controller in the DMC-1400 Econo Series that supports two axes. For this controller, arguments are specified for the X and Y 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 motor can be assigned to the variable 'V' with the command:

V=_TP

Usage Description

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

"While Moving" states whether or not the command is valid while the controller is performing a previously defined motion.

"In a program" states whether the command may be used as part of a user-defined program.

"Command Line" states whether the command may be used other than in a user-defined program.

"Can be Interrogated" states whether or not the command can be interrogated by using the "?" as a command argument.

"Used as an Operand" states whether the command has an associated operand.

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.

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, which are returned when the command is interrogated. The format value represents the number of digits before and after the decimal point.

Controller Usage

The controller usage indicates which models within the DMC-1400 Econo Series line support the current command. Controllers referenced in this manual are the DMC-1410, DMC-1411, DMC-1412, DMC-1414, DMC-1415, DMC-1416, DMC-1417 and DMC-1425. ALL indicates that all Econo controllers support the specific command.

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 down to 250 usec per sample.

Fast Firmware (DMC-1415/1416/1425)

The DMC-1415, DMC-1416 and DMC-1425 motion controllers can operate in 'fast mode' that allows operation down to 125 usec per sample.

In order to run the motion controller in fast mode, the fast firmware must be uploaded. This can be done through the Galil terminal software such as DMCTERM and WSDK. Use the menu option, "Update Flash EEPROM" to change the controller firmware. The fast firmware is included with the controller utilities.

When operating in fast mode, there are functions that are disabled and/or altered.

Commands which are not Allowed when Operating in Fast Mode:

Command
Gearing Mode
Ecam Mode
Analog Feedback (AF)
Stepper Motor Operation (MT 2, -2, 2.5, -2.5)
Trippoints allowed only in thread 0
Tell Velocity Interrogation Command (TV)

Commands which are Altered when Operating in Fast Mode:

Command	**Modification**
MT	Command argument 2, 2.5, -2, -2.5 not valid
AD, AI, AM, AP, AR, AS, AT, AV, MC, MF, MR, WC	Commands not allowed in thread 1

# #

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: **DEFAULTS:**

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

RELATED COMMANDS:

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

EXAMPLES:

```
#Loop; JP#Loop, x=10       ;'wait until x becomes 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:		**DEFAULTS:**	
While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	-
Command Line	Yes		
Controller Usage	**ALL**		

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

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

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

RELATED COMMANDS:

NO or '	comment

EXAMPLES:

```
BGX ;'comment

PRX=1000;BGX;AMX        ;'Save program line space

#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: **DEFAULTS:**

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

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: **DEFAULTS:**

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

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

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").

ARGUMENTS: AB n where

n = no argument or 1

1 aborts motion without aborting program, 0 aborts motion and program

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

USAGE:

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

OPERAND USAGE:

_AB gives the state of the Abort Input

RELATED COMMANDS:

SH	Turns servos back on if they were shut-off by Abort and OE1.

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 X-axis
BGX	Begin jog on X-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:		**DEFAULTS:**	
While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	-
Command Line	Yes		
Controller Usage	**ALL**		

RELATED COMMANDS:

@SQR	Square Root

EXAMPLES:

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

AC

FUNCTION: Acceleration

DESCRIPTION:

The Acceleration (AC) command sets the linear acceleration rate for independent moves, such as PR, PA and JG moves. The parameters input will be rounded down to the nearest factor of 1024. The units of the parameters are counts per second squared. The acceleration rate may be changed during motion. The DC command is used to specify the deceleration rate.

ARGUMENTS: AC n where

n is an unsigned number in the range in the range 1024 to 67107840

"?" returns the acceleration value

USAGE:

While Moving	Yes	Default Value	256000
In a Program	Yes	Default Format	8.0
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_ACx contains the value of acceleration.

RELATED COMMANDS:

DC	Specifies deceleration rate.
FA	Feedforward Acceleration.
IT	Smoothing constant

EXAMPLES:

AC 150000	Set acceleration to 150000
AC ?	Request the current Acceleration setting
0149504	Return Acceleration (resolution, 1024)
V=_AC	Assigns the current acceleration setting to the variable V

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

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 which moves away from the specified position.

The units of the command are quadrature counts. The motion profiler must be on or the trippoint will automatically be satisfied.

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

ARGUMENTS: ADn where

n is an unsigned integer in the range 0 to 2147483647, with no commas.

USAGE:

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

RELATED COMMANDS:

AR	After distance for repetitive triggering

EXAMPLES:

#A;DP0,0,0,0	Begin Program
PR 10000	Specify positions
BG	Begin motion
AD 5000	After motor travels 5000 units
MG "Halfway" ;TP	Send message
EN	End Program

Hint: *The AD command is accurate to the number of counts that occur in 2 servo samples (2 msec for TM 1000). 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). As the analog feedback is decoded by a 12-bit A/D converter, 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. When using Analog Feedback mode, X axis feedback must be wired to analog input 1 and Y axis feedback must be wired to analog input 2.

ARGUMENTS: AF x,y AFX=x AF a,b where

x,y are integers

1 = Enables analog feedback

0 = Disables analog feedback and switches to digital feedback

"?" returns a 0 or 1 which states whether analog feedback is enabled for the specified axes.

USAGE:

While Moving	No	Default Value	0,0,0,0
In a Program	Yes	Default Format	-
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**DMC-1415/1416/1425 only**		

OPERAND USAGE:

_AFx contains the value of acceleration for the specified axis.

RELATED COMMANDS:

CE	Configure Encoder

EXAMPLES:

AF 1,0	Analog feedback on X axis
V1 = _AFX	Assign feedback type to variable
AF ?	Interrogate feedback type

AI

FUNCTION: After Input

DESCRIPTION:

The AI command is used in motion programs to wait until after the specified input has occurred. If n is positive, it waits for the input to go high. If n is negative, it waits for n to go low.

ARGUMENTS: AI +/-n where

n is an integer in the range 1 to 7 decimal for DMC-1410/1411/1417/1415/1416 and 0 to 3 decimal for the DMC-1425.

USAGE:

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

RELATED COMMANDS:

@IN[n]	Function to read input 1 through 7 or 1 through 3 for DMC-1425
II	Input interrupt
#ININT	Label for input interrupt

EXAMPLES:

#A	Begin Program
AI 7	Wait until input 7 is high
SP 10000	Speed is 10000 counts/sec
AC 20000	Acceleration is 20000 counts/sec2
PR 400	Specify position
BG	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 position capture) of the controller. When the AL command is used to arm the position latch, the encoder position of the main encoder input will be captured upon a low going signal on Input 1. The command RL returns the captured position value. When interrogated or used in an operand the AL command will return a 1 if the latch is armed or a zero after the latch has occurred. The CN command will change the polarity of the latch.

ARGUMENTS: ALn where

n is X (or Y for the DMC-1425) or

n is SX for the aux encoder on DMC-1415/16

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	1.0
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

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

RELATED COMMANDS:

RL	Report Latch

EXAMPLES:

#START	Start program
AL	Arm latch
JG 50000	Set up jog at 50000 counts/sec
BG	Begin the move
#LOOP	Loop until latch has occurred
JP #LOOP,_AL=1	
RL	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. AM occurs when the profiler is finished generating the last position command. However, the servo motor may not be in the final position. Use TE to verify position error for servos, or use the MC trippoint to wait until final actual position is recorded.

ARGUMENTS: AM

USAGE:

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

RELATED COMMANDS:

BG	(_BG returns a 0 if motion complete)
MC	Actual Motion Complete (In-Position)

EXAMPLES:

#MOVE	Program MOVE
PR 5000	Position relative moves
BG	Start
AM	After the move is complete
EN	End of Program
#F;DP 0	Program F
PR 5000	Position relative moves
BG	Start
AM	After motion complete on all axes
MG "DONE";TP	Print message
EN	End of Program

***HINT**: AM is a very important command for controlling the timing between multiple move sequences. For example, if the motor is in the middle of a position relative move (PR) you cannot make a position absolute move (PA, BG) until the first move is complete. Use AM to halt the program sequences until the first motion is complete. AM tests for profile completion. The actual motor may still be moving. Another method for testing motion complete is to query the operand, _BG. This is equal to 1 during motion, and 0 when motion profiling is complete.*

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

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 F (1 to 6).

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	Default Value	---
In a Program	Yes	Default Format	---
Command Line	Yes		
Controller Usage	**DMC-1415/1416/1425**		

RELATED COMMANDS:

SB	Set Bit
CB	Clear Bit

EXAMPLES:

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 commanded motor position crosses the specified absolute position.
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. The motion profiler must be on or the trippoint will automatically be satisfied.

ARGUMENTS: APn where

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

USAGE:

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

RELATED COMMANDS:

AD	Trippoint for relative distances

EXAMPLES:

#TEST	Program B
DP0	Define zero
JG 1000	Jog mode (speed of 1000 counts/sec)
BG	Begin move
AP 2000	After passing the position 2000
V1=_TP	Assign V1 X 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 samples (2 msec for TM 1000). Multiply the speed by the time period of 2 samples 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 commanded motor position crosses the specified relative distance from either the start of the move or the last AR or AD command.
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. The motion profiler must be on or the trippoint will automatically be satisfied.

ARGUMENTS: ARn where

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

USAGE:

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

RELATED COMMANDS:

AP	Trippoint for after absolute position

EXAMPLES:

```
#A;DP 0             Begin Program
JG 50000            Specify speed
BG                  Begin motion
#B                  Label
AR 25000            After passing  25000 counts of relative distance on X-axis
MG "Passed_X";TP    Send message
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 speed is 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

USAGE:

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

EXAMPLES:

```
#SPEED          Program A
PR 100000       Specify position
SP 10000        Specify speed
BG              Begin
AS              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 profile 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:		**DEFAULTS:**	
While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	-
Command Line	Yes		
Controller Usage	**ALL**		

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

positive n waits n msec from reference

negative n waits n msec from reference and sets new reference after elapsed time period

(AT -n is equivalent to AT n; AT 0)

USAGE:

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

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

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	**DMC-14x5 / 6 ONLY**

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,t where

s and t are unsigned integers in the range 0 to 2147483647 decimal. 's' represents the vector distance to be executed in the S coordinate system and 't' represents the vector distance to be executed in the T coordinate system.

USAGE: **DEFAULTS:**

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

OPERAND USAGE:

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

EXAMPLES:

```
#MOVE;DP 0,0        Label
CAT                 Specify the T coordinate system
LMXY                Linear move for X,Y
LI 1000,2000        Specify distance
LI 2000,3000        Specify distance
LE
BGT                 Begin motion in the T coordinate system
AV ,500             After path distance = 500,
MG "Path>500";TPXY  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.

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	**DMC-1412/1415/1410/1417**		

OPERAND USAGE:

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

RELATED COMMANDS:

BB	Brushless Phase Begins
BC	Brushless Commutation
BD	Brushless Degrees

BB

FUNCTION: Brushless Phase Begins

DESCRIPTION:

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

ARGUMENTS: BB x where

x 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	**DMC-1412/1415/1410/1417**		

EXAMPLES:

BB30	The offsets sinusoidal motor is 30°

RELATED COMMANDS:

BA	Brushless Axis
BC	Brushless Commutation
BD	Brushless Degrees

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: BC

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	**DMC-1412/1415/1410/1417**		

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

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: BDx where

x sets the commutation phase in degrees within 0-360°.

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	**DMC-1412/1415/1410/1417**		

OPERAND USAGE:

_BD contains the commutation phase of the brushless motor.

RELATED COMMANDS:

BA	Brushless Axis
BB	Brushless Phase Begins
BC	Brushless Commutation

BG

FUNCTION: Begin

DESCRIPTION:

The BG command starts a motion. When Used as an Operand, the BG command will return a 1 if there is a commanded motion in progress, a 0 otherwise.

ARGUMENTS: BG

USAGE:

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

OPERAND USAGE:

_BGx contains a ‘0’ if motion complete on the axis, otherwise contains a ‘1’.

RELATED COMMANDS:

AM	After motion complete
ST	Stop motion

EXAMPLES:

PR 2000	Set up for a relative move
BG	Start motion
HM	Set up for the homing
BG	Start motion
JG 1000	Set up for jog
BG	Start motion
STATE=_BG	Assign a 1 to STATE if the axis is performing a move

HINT: *You cannot give another BG command until current BG motion has been completed. Use the AM trippoint to wait for motion complete between moves. Another method for checking motion complete is to test for _BG being equal to 0.*

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.

ARGUMENTS: BIx where

x indicates the starting input line.

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	**DMC-1412/1415/1410/1417**		

EXAMPLE:

BI5	The Hall sensors of the brushless motor are connected to inputs 5, 6 and 7.

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: **DEFAULTS:**

While Moving	Yes	Default Value of m	0
In a Program	No		
Command Line	Yes		
Controller Usage	**ALL CONTROLLERS**		

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 a commanded motion, the motion will decelerate to a stop. Reverse motion beyond this limit is not permitted. The reverse limit is activated at position n-1. To disable the reverse limit, set n to -2147483648. The units are in quadrature counts.

When the reverse software limit is activated, the automatic subroutine #LIMSWI will be executed if it is included in the program and the program is executing. See the section on Automatic Subroutines in the user manual.

ARGUMENTS: BL n where

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

-214783648 turns off the reverse limit.

USAGE:

While Moving	Yes	Default Value	-214783648
In a Program	Yes	Default Format	Position format
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_BLx contains the value of the reverse software limit..

RELATED COMMANDS:

FL	Forward Limit
_LR	Reverse Limit Operand
PF	Position Formatting

EXAMPLES:

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

HINT: *The DMC-141X also provides hardware limits.*

BM

FUNCTION: Brushless Modulo

DESCRIPTION:

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

ARGUMENTS: BMx where

x 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	**DMC-1412/1415/1410/1411**		

OPERAND USAGE:

_BM 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

BN

FUNCTION: Burn

DESCRIPTION:

The BN command saves certain controller parameters in non-volatile 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	CW	IA	PF
AF	DC	IL	
BA	DV	IT	SB
BB	EI	KD	SP
BI	EO	KI	TL
BL	ER	KP	TM
BM	FA	KS	TR
BO	FL	LZ	VA
CB	FV	MO	VD
CE	GA	MT	VF
CN	CW	OE	VS
CO	GR	OF	VT

ARGUMENTS: None

USAGE:

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

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: BOx,x where

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

x = ? 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	**DMC-1412/1415/1410/1417**		

OPERAND USAGE:

_BO 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

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	Default Value	---
In a Program	No		
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	No		
Controller Usage	**DMC-1412/1414/1415/1416/1425**		

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 5 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. The configuration is saved by using the burn command, BN.

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 or BS V, n where

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	**DMC-1412/1415/1410/1417**		

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

BV

FUNCTION: Burn Variables and Array

DESCRIPTION:

The BV command saves the defined variables and arrays 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 variables is complete. Operand returns size of installed EEPROM.

ARGUMENTS: None

USAGE:

While Moving	No	Default Value	---
In a Program	No		
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	Yes		
Controller Usage	**DMC-1412/1414/1415/1416/1425**		

***Note1**: This command will store the ECAM table values in non-volatile EEPROM memory.*

***Note2**: 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 5 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: BZ x where

x is a real number between -4.998 and 4.998.

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	**DMC-1412/1415/1410/1417**		

OPERAND USAGE:

_BZ 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

CB

FUNCTION: Clear Bit

DESCRIPTION:

The CB command clears one of three bits on the output port sets the output to logic 0. The CB and SB (Set Bit) instructions can be used to control the state of output lines.

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. A standard DMC-1400 controller has 3 TTL digital outputs.

Note: When using Modbus devices (DMC-1415/1416/1425 only), the I/O points of the Modbus devices are calculated using the following formula:

n = (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 F.

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.

USAGE:

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

RELATED COMMANDS:

SB	Set Bit
OP	Define all outputs

EXAMPLES:

CB 1	Clear output bit 1
CB 2	Clear output bit 2
CB 3	Clear output bit 3

CC

FUNCTION: Configure Communications Port 2

DESCRIPTION::

The CC command configures baud rate, handshake, echo, and daisy chain features for the AUX SERIAL PORT, referred to as Port 2. This command must be given before using the MG, IN or CI commands with Port 2.

ARGUMENTS: CC m,n,r,p

m - Baud rate	300,1200,4800,9600,19200,38400
n - Handshake	0 for handshake off, 1 for handshake on
r - Mode	0 for daisy chain off, 1 for daisy chain on
p - Echo	0 for echo off, 1 for echo on

Note: echo only active when daisy chain feature is off

USAGE:

While Moving	Yes	Default Value	0,0,0
In a Program	Yes	Default Format	---
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	No		
Controller Usage	**DMC-1412/1414**		

RELATED COMMANDS:

CI	Communications Interrupt

EXAMPLES:

CC 9600,0,0,1	9600 baud, no handshake, daisy chain off, echo on. Typical setting with TERM-1500.
CC 19200,1,1,0	19,200 baud, handshake on, daisy chain on, echo off. Typical setting in daisy chain mode.

CD

FUNCTION: Contour Data

DESCRIPTION:

The CD command specifies the incremental position. The units of the command are in quadrature counts for servo mode, steps for stepper mode. This command is used only in the Contour Mode (CM).

ARGUMENTS: CD n where

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

USAGE:

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

RELATED COMMANDS:

CM	Contour Mode
WC	Wait for Contour
DT	Time Increment

EXAMPLES:

CM	Specify Contour Mode
DT 4	Specify time increment for contour
CD 200	Specify incremental positions of 200 counts
WC	Wait for complete
CD 100	New position data
WC	Wait for complete
DT0	Stop Contour
CD 0	Exit Mode

CE

FUNCTION: Configure Encoder

DESCRIPTION:

The CE command configures the encoder inputs to the quadrature type or the pulse and direction type. It also allows inverting the polarity of the encoders. The configuration applies independently to the main axis encoder and the auxiliary encoder inputs.

ARGUMENTS: CE n where

n is an integer in the range of 0 to 15. Each integer is the sum of two integers r and s which configure the main and the auxiliary encoders. The values of r and s are

R =	MAIN ENCODER TYPE	S =	AUXILIARY ENCODER TYPE
0	Normal quadrature	0	Normal quadrature
1	Normal pulse and direction	4	Normal pulse and direction
2	Reversed quadrature	8	Reversed quadrature
3	Reversed pulse and direction	12	Reversed pulse and direction

For example: n = 6 implies r = 2 and s = 4, both encoders are reversed quadrature.

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	2.0
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_CE contains the value of encoder type for the main and auxiliary encoder.

RELATED COMMMANDS:

MT	Specify motor type

EXAMPLES:

CE 0	Configure encoders
CE ?	Interrogate configuration
V = _CE	Assign configuration to a variable

Note 1: *When using pulse and direction encoders, the pulse signal is connected to CHA and the direction signal is connected to CHB.*

Note 2: *when the controller is a DMC-1425, the y-axis encoder is still considered the x-axis auxiliary encoder for the sake of the CE command.*

CF

FUNCTION: Configure

DESCRIPTION:

The CF command is used to specify the communication port to which unsolicited responses are sent. By default, the DMC-1415/1416/1425 will send unsolicited responses to the serial port.

ARGUMENTS: CF n where n is A thru F for Ethernet handles 1 thru 6, S for Main serial port.

USAGE:

While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	-
Command Line	No		
Controller Usage	**DMC-1415/1416/1425**		

OPERAND USAGE:

_CF will return the current port selected for unsolicited responses from the controller. The _CF will return a decimal value.

EXAMPLES:

CFA	Select Ethernet handle A to return unsolicited responses.
MG_CF	Interrogate configuration
:65.000	Response from CF showing handle A as default port. The number 65 is the decimal representation for the ASCII character "A"

CI

FUNCTION: Communication Interrupt

DESCRIPTION:

The CI command configures a program interrupt based on characters received on either Port 1, the MAIN serial port, or Port 2, the AUX serial port. An interrupt causes program flow to jump to the #COMINT subroutine label. If multiple program threads are used, the #COMINT subroutine runs in thread 0 and thread 1 continues to run in the background without interruption. The characters received on the serial port are stored in internal variables such as P2CH. See chapter 7 for more detailed information on the communications interrupt.

ARGUMENTS: CI m,n,o

PARAMETER	EXPLANATION
m = 0	Do not interrupt Port 1
m = 1	Interrupt on carriage return character on Port 1
m = 2	Interrupt on any character Port 1
m = -1	Clear interrupt data buffer
n = 0	Do not interrupt Port 2
n = 1	Interrupt on carriage return character on Port 2
n = 2	Interrupt on any character Port 2
n = -1	Clear interrupt data buffer
o = 0	Disable live data mode for Port 1
o = 1	Enable live data mode for Port 1

USAGE:

While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	-
Command Line	No		
Can be Interrogated	No		
Used as an Operand	No		
Controller Usage	**DMC-1412/1414**		

EXAMPLES:

CI 0,1,0	Interrupt on <enter> received on Port 2
CI 0,2,0	Interrupt on a single character received on Port 2
CI 1,2,1	Interrupt on <enter> received on Port 1, interrupt on any character received on Port 2

NOTE: *The third field of the CI command enables or disables live data mode on Port 1. If live data mode is enabled, then the controller will not respond to commands sent to the main serial port. This setting is necessary to use the communications interrupt on the main serial port.*

CM

FUNCTION: Contouring Mode

DESCRIPTION:

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

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

ARGUMENTS: CM

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

USAGE:

While Moving	No	Default Value	---
In a Program	No	Default Format	1.0
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	Yes		
Controller Usage	**ALL**		

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
1	Contour Buffer is full
CM	Specify 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, the home switch and the latch input.

ARGUMENTS: CN m,n,o where

m,n,o are integers with values 1 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	Default Value	-1.-1.-1
In a Program	Yes	Default Format	2.0
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	No		
Controller Usage	**ALL**		

RELATED COMMANDS:

MT	Motor Type

EXAMPLES:

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

CO

FUNCTION: Configure Outputs

DESCRIPTION:

The CO command configures the extended I/O in blocks of 8.

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 block 2 and the most significant bit represents block 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 block. To configure a block as an output block, substitute a one into that n_x in the formula. If the n_x value is a zero, then the block of 8 I/O points will be configured as an input. For example, if block 3 and 4 is to be configured as an output, CO 6 is issued.

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	-
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	Yes		
Controller Usage	**DMC-1415/1416/1425**		

OPERAND USAGE:

_CO returns output configuration value

RELATED COMMANDS:

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

EXAMPLES:

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

Hint: *See 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: **DEFAULTS:**

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

RELATED COMMANDS:

& |	Logical operators AND and OR

EXAMPLES:

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

#COMINT

FUNCTION: Communication interrupt automatic subroutine

DESCRIPTION:

#COMINT can be configured by the CI command to run either when any character or a carriage return is received on the auxiliary serial port. CI,1 must be issued to use #COMINT.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	No
Controller Usage	**DMC-1412 / 4 ONLY**

RELATED COMMANDS:

P1CD P2CD	Serial port 1/2 code
P1CH P2CH	Serial port 1/2 character
P1NM P2NM	Serial port 1/2 number
P1ST P2ST	Serial port 1/2 string
CI	Configure #COMINT (and set operator data entry mode)
CC	Configure serial port 2
EN	End subroutine

EXAMPLES:

```
CC9600,0,0,0
CI2 ;'interrupt on any character

#Loop
  MG "Loop"  ;'print a message every second
  WT 1000
JP#Loop

#COMINT
  MG "COMINT" ;'print a message when a character is
          received
EN1,1
```

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*

@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 -2147483648 to 2147483647.

USAGE: **DEFAULTS:**

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

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 $\Delta\theta$. A positive $\Delta\theta$ denotes counterclockwise traverse, negative $\Delta\theta$ 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, $\Delta\theta$, must be specified. Radius units are in quadrature counts. θ and $\Delta\theta$ 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, $\Delta\theta$ < 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)

$\Delta\theta$ 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 * $\Delta\theta$ 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**		

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:

VMXY	Specify vector motion in the X and Y plane
VS 10000	Specify vector speed
CR 1000,0,360	Generate circle with radius of 1000 counts, start at 0 degrees and complete
CR 1000,0,360 < 40000	Generate circle with radius of 1000 counts, start at 0 degrees and complete
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 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.

USAGE:

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

OPERAND USAGE:

_CSx contains the segment number in the sequence specified by x, S or T. This operand is valid in the Linear mode, LM, Vector mode, VM

RELATED COMMANDS:

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

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 which 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 is a number, either 0,1 or 2:

0	Causes the controller to return the copyright information
1	Causes the controller to set the MSB of unsolicited returned characters to 1
2	Causes the controller to not set the MSB of unsolicited characters.
“?”	returns the copyright information for the controller

m is 0 or 1 (optional)

0	Causes the controller to pause program execution when output FIFO is full until FIFO no longer full.
1	Causes the controller to continue program execution when output FIFO is full - output characters after FIFO is full will be lost.

USAGE:

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

OPERAND USAGE:

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

***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 which 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.*

***Note 2:** The CW, 1 Command should not be used with the DMC-1415 or DMC-1425. These controllers have only a 1 byte UART and are subsequently always seen as full.*

***Note 3:** Specifying both fields of the CW command is not valid. If CW, 2,1 is issued, for instance, the controller will not recognize the second field. Instead, the user must issue CW2 & CW,1 individually.*

DA

FUNCTION: Deallocate 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 commas when specified in one command. The * argument deallocates all the variables, and *[0] deallocates all the arrays.

ARGUMENTS: DA c[0],d,etc. where

c[0] - Defined array name

d - Defined variable name

* - Deallocates all the variables

*[0] - Deallocates all the arrays

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

USAGE:

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

OPERAND USAGE:

_DA contains the total number of arrays available. For example, on a DMC-1417, before any arrays have been defined, the operand _DA is 7. If one array is defined, the operand _DA will return 6.

RELATED COMMANDS:

DM	Dimension Array

EXAMPLES:

‘Cars’ and ‘Salesmen’ are arrays and ‘Total’ is a variable.

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

***NOTE:** Since this command deallocates 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 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 where

n is an unsigned numbers in the range 1024 to 67107840

"?" returns the value

USAGE:

While Moving	Yes*	Default Value	256000
In a Program	Yes	Default Format	8.0
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_DCx contains the deceleration rate

RELATED COMMANDS:

AC	Acceleration
PR	Position Relative
SP	Speed
JG	Jog
BG	Begin
IT	Smoothing constant - S-curve

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 n command defines the position of the auxiliary encoder.

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

Note: The auxiliary encoder is not available for the stepper axis.

ARGUMENTS: DE n where

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

"?" returns the position of the auxiliary encoder

"?" returns the commanded reference position of the motor (in step pulses) when used with a stepper motor. Example: DE0. 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 use the DP command.)

USAGE:

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

OPERAND USAGE:

_DE contains the current position of the specified auxiliary encoder.

EXAMPLES:

:DE 0	Set the current auxiliary encoder position to 0
:DE?	Return auxiliary encoder positions
:DUALX=_DE	Assign auxiliary encoder position of X-axis to the variable DUALX

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 DMC-14XX. Instructions in the file will be accepted as a datastream 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 any programs in the DMC-14XX RAM. The data is entered beginning at line 0. If there are too many lines or too many characters per line, the DMC-14XX 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 DMC-14XX program in RAM.

ARGUMENTS: DL n

n = no argument	Downloads program beginning at line 0. Erases programs in RAM.
n = #Label valid program	Begins download at line following #Label where label may be any label.
n = #	Begins download at end of program in RAM.

USAGE:

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

OPERAND USAGE:

When used as an operand, _DL gives the number of available labels. The total number of labels is 126.

RELATED COMMANDS:

UL	Upload

EXAMPLES:

DL;	Begin download
#A;PR 4000;BG	Data
AM;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 n total elements. 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 alphanumeric characters, starting with an uppercase alphabetic character. n is the number of entries from 1 to 1000 (1 to 2000 for the DMC-1415/1416/1425).

DM? Returns the number of array elements available.

USAGE:

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

OPERAND USAGE:

_DM contains the available array space. For example, on a DMC-1417, before any arrays have been defined, the operand _DM will return 1000. If an array of 100 elements is defined, the operand _DM will return 900.

RELATED COMMANDS:

DA	Deallocate 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[1000]	Define dimension of array Tests with 1000 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 commanded reference position in stepper mode. The units are in steps. Example: DP0. This will set the DE value to zero, but will not effect the TP value.

ARGUMENTS: DP n where

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

"?" returns the current position of the motor

USAGE:

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

OPERAND USAGE:

_DP contains the current position.

EXAMPLES:

:DP 0	Sets the current position of the X-axis to 0
:DP –50000	Sets the current position to -50000.
:DP ?	
-0050000	Returns the 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 Contouring Mode. Sending the DT command once will set the time interval for all following contour data until a new DT command is sent. 2^n samples is the time interval. Sending DT0 followed by CD0 command terminates the Contour Mode.

ARGUMENTS: DT n where

n is an integer in the range 0 to 8. 0 terminates the Contour Mode. n=1 thru 8 specifies the time interval of 2^n samples. By default the sample period is 1 msec (set by TM command); with n=1, the time interval would be 2 msec.

DT? Returns the value for the time interval for contour mode.

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	1.0
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**ALL**		

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
TM	Time

EXAMPLES:

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

DV

FUNCTION: Dual Velocity (Dual Loop)

DESCRIPTION:

The DV function changes the operation of the PID servo filter. It causes the KD (derivative) term to operate on the dual encoder instead of the main encoder. To take advantage of this mode, mount the main encoder to the load and the dual encoder to the motor.

ARGUMENTS: DVn where

n may be 0 or 1. 0 disables the function. 1 enables the dual loop.

"?" returns a 0 if dual velocity mode is disabled and 1 if enabled for the specified axis.

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	1.0
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_DV contains the state of dual velocity mode. 0 = disabled, 1 = enabled

RELATED COMMANDS:

KD	Damping constant
FV	Velocity feedforward

EXAMPLES:

DV 1	Enables dual loop
DV 0	Disables DV

***HINT:** The DV command is useful in backlash and resonance compensation.*

EA

FUNCTION: Choose ECAM master

DESCRIPTION:

The EA command selects the master axis for the electronic cam mode. Any axis may be chosen. Note: This command is only used with the DMC-1425. All other Econo series controllers use the auxiliary encoder as the master automatically.

ARGUMENTS: EA x where

x is one of the axis specified as X or Y.

USAGE:

While Moving	Yes	Default Value
In a Program	Yes	Default Format
Command Line	Yes	
Controller Usage	**DMC-1425**	

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:

EAY	Select Y 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 cam mode and n = 0 stops cam mode.

EB? Returns a 0 if ECAM is disabled and 1 if enable.

USAGE:

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

OPERAND USAGE:

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

RELATED COMMANDS:

EM	Specify Ecam Cycle
EP	CAM table intervals & starting point

EXAMPLES:

EB1	Starts ECAM mode
EB0	Stops ECAM mode
B = _EB	Assigns status of cam mode to variable “B”

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**		

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.

ARGUMENTS: ED n where

n specifies the line number to begin editing. The default line number is the last line of program space with commands.

USAGE:

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

OPERAND USAGE:

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

EXAMPLES:

```
000 #START
001 PR 2000
002 BG
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 operation at a specified position of the master encoder. 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 where

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

"?" returns 1 if specified axis is engaged and 0 if disengaged

USAGE:

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

OPERAND USAGE:

_EG contains ECAM status . 0 = axis is not engaged, 1 = axis is engaged.

RELATED COMMANDS:

EB	Enable Ecam
EQ	Ecam quit

EXAMPLES:

EG 700	Engages axes at master position 700.
MG_EG	Return the status of the 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.*

EI

FUNCTION: Enable Interrupts

DESCRIPTION:

The EI command enables an ISA, PC/104 or PCI bus interrupt on conditions such as motion complete or excess positional error. The conditions are selected by the parameter m where m is the bit mask for the selected conditions as shown below. Prior to using the EI command, one IRQ line must be jumpered on the DMC-141X. An interrupt service routine must also be incorporated in your host program. See Chapter 4 in the product manual for details.

ARGUMENTS: EI n where

n is interrupt condition for bits set. n = ΣBit number (2^m) Condition

Bit Number (m)	Condition
15	Reserved
14	Reserved
13	Application Program stopped
12	Reserved
11	Watchdog Timer
10	Limit switch
9	Excess Position error*
8	Motion complete
7	reserved
6	Input 7*
5	Input 6*
4	Input 5*
3	Input 4*
2	Input 3*
1	Input 2*
0	Input 1*

The * conditions must be re-enabled after each occurrence.

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	---
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	No		
Controller Usage	**DMC-1410/1411/1417**		

RELATED COMMANDS:

UI	User interrupt

EXAMPLES:

1. Specify interrupts for motion complete and limit switch.

   Enable bits 8 and 10. n = $2^{10} + 2^8$ = 1024 + 256 = 1280 EI 1280

2. Specify interrupts on Inputs 2 and 4.

   Enable bits 1 and 3. n = $2^1 + 2^3$ = 2 + 8 = 10 EI 10

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	Default Value
In a Program	Yes	Default Format
Command Line	No	
Controller Usage	**DMC-1415/1416/1425**	

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 slave, 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,m where

n - change in slave axis, between 1 and 2147483647

m - change in master encoder, between 1 and 8388607.

USAGE:

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

OPERAND USAGE:

_EM contains the cam cycle modulus of the motor..

RELATED COMMANDS:

EP	CAM table intervals & starting point
ET	Electronic CAM table
EB	Enable Ecam

EXAMPLES:

EM 0,2000	Define the changes in the motor to be 0. Define master cycle as 2000.
V = _EM	Assigns motor cam cycle distance to variable “V”

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 also 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

m=0	Return from subroutines without restoring trippoint
m=1	Return from #COMINT and restore trippoint
n=0	Return from #COMINT without restoring interrupt
n=1	Return from #COMINT and restore interrupt

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

Note 2: 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 by continuing to wait for the trippoint as if nothing happened, or clear the trippoint and continue executing the program at the command just after the trippoint. The EN arguments will specify how the #COMINT routine handles trippoints.

Note 3: 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	Default Value	n=0, m=0
In a Program	Yes	Default Format	
Command Line	No		
Can be Interrogated	No		
Used as an Operand	No		
Controller Usage	**ALL**		

RELATED COMMANDS:

RE	Return from error subroutine
RI	Return from interrupt subroutine

EXAMPLES:

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

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 (ININT) subroutine.*

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 conditional statements 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	Default Value
In a Program	Yes	Default Format
Command Line	No	
Controller Usage	**DMC-1415/1416/1425**	

RELATED COMMANDS:

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

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 or 1. 0 turns echo off, 1 turns echo on.

USAGE:

While Moving	Yes	Default Value (DMC-1410/1411/1417)	0
In a Program	Yes	Default Value (All others)	1.0
Command Line	Yes	Default Format	1.0
Can be Interrogated	No		
Used as an Operand	No		
Controller Usage	**ALL**		

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. m can be changed while ECAM is running.

USAGE:

While Moving	Yes	Default Value	256,0
In a Program	Yes	Default Format	
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes (m only)		
Controller Usage	**ALL**		

OPERAND USAGE:

_EP contains the value of the interval m.

RELATED COMMANDS:

EB	Enable Ecam
EG	Engage Ecam
EM	Specify Ecam Cycle
EQ	Ecam quit
ET	Electronic CAM 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. If a value is specified outside of the master's range, the slave will disengage immediately.

ARGUMENTS: EQ n where

n is the master position at which the axis is to be disengaged.

"?" contains a 1 if engage command issued and slave is waiting to engage, 2 if disengage command issued and slave 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		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

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

RELATED COMMANDS:

EB	Enable Ecam
EG	Engage Ecam
EM	Specify Ecam Cycle
EP	CAM table intervals & starting point
ET	Electronic CAM table

EXAMPLES:

EQ 300	Disengages the motor at master position 300.

***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 error 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. The units of ER are quadrature counts.

ARGUMENTS: ER nwhere

n is an unsigned number in the range 1 to 32767.

"?" returns the value of the Error limit.

USAGE:

While Moving	Yes	Default Value	16384
In a Program	Yes	Default Format	Position Format
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_ER contains the value of the Error limit.

RELATED COMMANDS:

#POSERR	Automatic Error Subroutine

EXAMPLES:

ER 200	Set the error limit to 200
ER ?	Return value
00200	
V1=_ER	Assigns V1 value of ER
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, x, will be multiplied by the ratio m/n. When m<n, the resolution of the second axis, y, will be multiplied 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	Default Value	1,1
In a Program	Yes	Default Format	
Command Line	Yes		
Controller Usage	**ALL**		

RELATED COMMANDS:

VM	Vector Mode
CR	Circle move
VP	Vector position

EXAMPLES:

VMXY;ES3,4	Divide Y resolution by 4/3

***NOTE**: ES must be issued after VM command*

ET

FUNCTION: Electronic cam table

DESCRIPTION:

The ET command sets the ECAM table entries for the slave axis. The values of the master are not required. The slave entry (n) is the position of the slave 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] = m where

n is an integer between 0 and 256

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

USAGE:

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

RELATED COMMANDS:

EB	Enable Ecam
EG	Engage Ecam
EM	Specify Ecam Cycle
EP	Specify Ecam intervals and starting point
EQ	Ecam quit

EXAMPLES:

ET [7] = 1000	Specifies the position of the slave that must be synchronized with the eighth increment of the master.

FA

FUNCTION: Acceleration Feedforward

DESCRIPTION:

The FA command sets the acceleration feedforward coefficient, or returns the previously set value. 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 Feedforward Bias = FA · AC · 1.5 · 10^{-7}

Deceleration Feedforward Bias = FA · DC · 1.5 · 10^{-7}

The Feedforward Bias product is limited to 10 Volts. FA will only be operational during independent moves.

ARGUMENTS: FA n where

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

"?" returns the value of the feedforward acceleration coefficient.

FA has a resolution of .25

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	4.0
Command Line	Yes		
Can be Interrogated	Yes	FA ?	
Used as an Operand	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_FA contains the value of the feedforward acceleration coefficient.

RELATED COMMANDS:

FV	Velocity feedforward

EXAMPLES:

AC 500000	Set feedforward coefficient to 10.
FA 10	The effective bias will be 0.75V
FA ?	Return value
010	

***NOTE**: If the feedforward 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

USAGE:

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

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

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 direction of motion is specified by the sign of the JG command.

ARGUMENTS: FI

USAGE:

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

RELATED COMMANDS:

FE	Find Edge
HM	Home
BG	Begin
AC	Acceleration Rate
DC	Deceleration Rate
SP	Speed for search

EXAMPLES:

```
#HOME               Home Routine
JG 500              Set speed and forward direction
FI                  Find index
BG                  Begin motion
AM                  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, the motor will decelerate to a stop. Forward motion beyond this limit is not permitted. The forward limit is activated at position n + 1. The forward limit is disabled at position 2147483647. The units are in counts.

When the forward software limit is activated, the automatic subroutine #LIMSWI will be executed if it is included in the program and the program is executing. See section on Automatic Subroutines in the user manual.

ARGUMENTS: FL n where

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

2147483647 turns off the forward limit

“?” returns the value of the forward limit switch

USAGE:

While Moving	Yes	Default Value	2147483647
In a Program	Yes	Default Format	Position Format
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_FL contains the value of the forward software limit.

RELATED COMMANDS:

BL	Reverse Limit
_LF	Forward Limit Operand
PF	Position Formatting

EXAMPLES:

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

***HINT:** The DMC-141X also provides 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:		**DEFAULTS:**	
While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	-
Command Line	Yes		
Controller Usage	**ALL**		

RELATED COMMANDS:

@INT	Integer part

EXAMPLES:

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

FV

FUNCTION: Velocity Feedforward

DESCRIPTION:

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

Velocity feedforward bias = $1.22 \cdot 10^{-6} \cdot$ FV · Velocity [in ct/s].

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

ARGUMENTS: FVn where

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

"?" returns the feedforward velocity for the specified axis.

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	3.0
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_FV contains the feedforward velocity.

RELATED COMMANDS:

FA	Acceleration feedforward

EXAMPLES:

FV 10	Set feedforward coefficients to 10
JG 30000	This produces 0.366 volts.
FV ?	Return the value
010	

GA

FUNCTION: Master Axis for Gearing

DESCRIPTION:

The GA command specifies the master axes for electronic gearing. Note: This command is only valid for the DMC-1425. All other Econo series controllers must use the auxiliary encoder as the master.

The master axis of the DMC-1425 must be either the X or Y axis. These may be specified as either the commanded position or the actual position. When the master is a simple axis, it may move in any direction and the slave follows. The slave axes and ratios are specified with the GR command and gearing is turned off by the command GR0.

NOTE: To gear to the dual encoder using a 141X controller, simply use the GR command.

ARGUMENTS: GA x,x or GAX=x where

x can be X or Y. The value of x is used to set the specified main 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 'X' axis, the second position corresponds to the 'Y' axis, etc. A comma must be used in place of an argument if the corresponding axes will not be a slave.

x can be CX or CY. The value of x is used to set the commanded position of the specified axis as the gearing master.

“?” returns the GA setting

USAGE:

While Moving	Yes	Default Value
In a Program	Yes	Default Format
Command Line	Yes	
Controller Usage	**DMC-1425**	

RELATED COMMANDS:

GR	Gear Ratio
GM	Gantry Mode

EXAMPLES:

#GEAR	Gear program
GA ,X	Specify X axis as master for Y
GR ,.5	Specify Y ratio
JG 5000	Specify master jog speed
BGX	Begin motion
WT 10000	Wait 10000 msec
STX	Stop

Hint: *Using the commanded position as the master axis is useful for gantry applications.*

GN

FUNCTION: Gain

DESCRIPTION:

The GN command sets the gain of the control loop or returns the previously set value. It fits in the z-transform control equation as follows:

D(z) = GN(z-ZR)/z

ARGUMENTS: GN n where

n is an unsigned integer in the range 0 to 2047 decimal.

"?" returns the value of the gain.

USAGE:

While Moving	Yes	Default Value	70
In a Program	Yes	Default Format	4
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**DMC-1410/1411/1412/1414/1417**		

OPERAND USAGE:

_GN contains the value of the gain.

RELATED COMMANDS:

ZR	Zero
KI	Integrator
KP	Proportional
KD	Derivative

EXAMPLES:

GN 12	Set gain to 12
GN ?	Returns gain
0006	

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 or GMX=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	Default Value	0
In a Program	Yes	Default Format	1.0
Command Line	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_GMx 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	Enable GM on all axes
GM 0	Disable GM on X-axis other axes remain unchanged

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 slave axis in the electronic gearing mode. The master axis for the DMC-1425 is specified with the GA command. For all 141X controllers, the master axis need not be set. Simply assign a ratio GRN to gear the x axis to the dual encoder. The master axis for all other Econo series controllers is the auxiliary encoder in servo mode, the main encoder input in stepper mode. Gear ratio may range between +/-127.9999. The slave axis will be geared to the actual position of the master. The master can go in both directions. GR 0 disables gearing. A limit switch sets the GR to 0 gearing.

ARGUMENTS: GR n where

n is a signed number in the range +/-127, with a fractional resolution of .0001

0 disables gearing

"?" returns the value of the gear ratio

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	3.4
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_GR contains the value of the gear ratio.

EXAMPLES:

#GEAR	
GR .25	Specify gear ratio
EN	End program

HM

FUNCTION: Home

DESCRIPTION:

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

For servo motor operation, the first stage consists of the motor moving 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.

The second stage consists of the motor changing directions and slowly approaching the transition again. When the transition is detected, the motor is stopped instantaneously..

The third stage consists of the motor slowly moving 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.

ARGUMENTS: None

USAGE:

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

OPERAND USAGE:

_HM contains the state of the home switch.

RELATED COMMANDS:

CN	Configure Home
FI	Find Index Only
FE	Find Home Only

EXAMPLES:

HM	Set Homing Mode
BG	Begin Homing

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

HX

FUNCTION: Halt Execution

DESCRIPTION:

The HX command halts the execution of any of the two programs that may be running independently in multitasking. The parameter n specifies either program to be halted.

ARGUMENTS: HX n where

n is either 0 or 1 to indicate the task number

USAGE:

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

OPERAND USAGE:

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

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

RELATED COMMANDS:

XQ	Execute program

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 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. (TCP/IP connection only)

>u specifies the multicast IP address where u is an integer between 0 and 63. (UDP/IP connection only)

IA? will return the IP address of the controller

USAGE:

While Moving	No	Default Value	n = 0, t=250
In a Program	No	Default Format	
Command Line	Yes		
Controller Usage	**DMC-1415/1416/1425**		

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 5. 0 represents handle A, 1 handle B, etc.
_IA4	Contains the handle number of the connection that caused the execution of #TCPERR. Contains "-1" if no error.

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	Default Value	-
In a Program	Yes	Default Format	-
Command Line	No		
Controller Usage	**DMC-1415/1416/1425**		

RELATED COMMANDS:

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

EXAMPLES:

IF (_TEX<1000)	IF conditional statement based on X motor position
MG "Motor is within 1000 counts of zero"	Message to be executed if “IF” conditional statement
ENDIF	End of IF conditional statement
IF(_TPX>5000)&(_TPY>5000)	IF conditional statement based on X and Y motor position
MG “Motors too Far”	Message to be executed if “IF” statement is true
ENDIF	End of IF statement

IH

FUNCTION: Open Internet Handle

DESCRIPTION:

The IH command is used when the DMC-1415, DMC-1416 or DMC-1425 is operated as a master (also known as a client). This command opens a handle and connects to a slave.

Each controller may have 6 handles open at any given time. They are designated by the letters A through F. 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.

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 or F

ip0,ip1,ip2,ip3 are integers between 0 and 255 and represent the individual fields of the IP address. These values must be separated by commas.

n is a signed integer between - 2147483648 and 2147483648. 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, or C=-2 for TCP/IP.

IHN => C closes all handles except for the one sending the command; where C=-1 UDP, or C=-2 for TCP.

<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, -2 for TCP/IP, and –3 for TCP/IP reset

"?" 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 -5 while establishing UDP handle
	contains -6 while establishing TCP handle
_IHh3	contains a 0 if the ARP was successful
	contains a 1 if it has failed or is still in progress.

_Ihh4	contains a 1 if waiting for a slave response
	contains 2 if transmission is successful
	contains 3 if transmission error occurs
	contains 4 if transmission timeout

USAGE:

While Moving	No	Default Value	-
In a Program	Yes	Default Format	
Command Line	Yes		
Controller Usage	**DMC-1415/1416/1425**		

RELATED COMMANDS:

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. This function triggers when the controller sees a logic change from high to low on a specified input.

If any of the specified inputs go low 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

To avoid returning to the main program on an interrupt, use the command ZS to zero the subroutine stack and use the II command to reset the interrupt..

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 0001111 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 0000010), input 2 will be activated by a logic ‘1’ and inputs 1,3, and 4 will be activated with a logic “0”.

Note: The ‘p’ data field is only supported by the DMC-1415/1416/1425.

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	3.0 (mask only)
Command Line	No		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**ALL (See Note for ‘p’ data field)**		

RELATED COMMANDS:

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

EXAMPLES:

```
#A                              Program A
II 1                            Specify interrupt on input 1
JG 5000                         Specify jog speed
BG                              Begin motion
#LOOP;JP #LOOP                  Loop
EN                              End Program
#ININT                          Interrupt subroutine
ST;MG "INTERRUPT"               Stop X, print message
AM                              After stopped
#CLEAR;JP#CLEAR,@IN[1]=0        Check for interrupt clear
BG                              Begin motion
RI                              Return to main program
```

NOTE: *An application program must be running on the controller for the Input Interrupt function to work.*

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 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 where

n is a number in the range 0 to 9.9988 Volts with a resolution of .0003.

“?” returns the value of the integrator limit

USAGE:

While Moving	Yes	Default Value	9.9982
In a Program	Yes	Default Format	1.4
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_IL contains the value of the integrator limit.

RELATED COMMANDS:

KI	Integrator

EXAMPLES:

KI 2	Integrator constants
IL 3	Integrator limits
IL ?	Returns the limit
3.0000	

IN

FUNCTION: Input Variable

DESCRIPTION:

The IN command allows a variable to be input from a keyboard. When the IN command is executed in a program, the prompt message is displayed. The operator then enters the variable value followed by a carriage return. The entered value is assigned to the specified variable name.

The IN command holds up execution of following commands in a program until a carriage return or semicolon is detected. If no value is given prior to a semicolon or carriage return, the previous variable value is kept. Input Interrupts, Error Interrupts and Limit Switch Interrupts will still be active.

ARGUMENTS: IN{P1 or P2} ”m” , n {So} where

"m" is the prompt message. May be letters, numbers, or symbols up to maximum line length and must be placed in quotations. Make sure that maximum line length is not exceeded (40 characters DMC-1410/1411/1412/1414, 80 characters DMC-1415/1416/1425).

n is the name of variable to hold value returned from input

{P1} specifies Port1, the MAIN serial port (optional-Main port by default) DMC-1412/1414 only.

{P2} specifies Port2, the AUX serial port (optional-Main port by default) DMC-1412/1414 only.

{So} specifies string data where o is the number of characters from 1 to 6

Note 1: The IN command can not be used over Ethernet. The IN command defaults to {P1}.

Note 2: Configure Port 2 communications w/ the CC command before using IN command w/ Port 2.

Note 3: The IN command can only be used in thread 0.

Note 4: Do not include a space between the comma at the end of the input message and the variable name.

USAGE:

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

EXAMPLES: Operator specifies length of material to be cut in inches and speed in inches/sec (2 pitch lead screw, 2000 counts/rev encoder).

```
#A                              Program A
CI -1                           Clear Input Buffer*
IN "Enter Speed(in/sec)",V1     Prompt operator for speed
IN "Enter Length(in)",V2        Prompt for length
V3=V1*4000                      Convert units to counts/sec
V4=V2*4000                      Convert units to counts
SP V3                           Speed command
PR V4                           Position command
BGX;AMX                         Begin motion; Wait for motion complete
MG "MOVE DONE"                  Print Message
EN                              End Program
```

*NOTE: It is a good practice to clear the input buffer before executing the IN command

@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: **DEFAULTS:**

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

RELATED COMMANDS:

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

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: **DEFAULTS:**

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

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 n command is equivalent to a PR n and BG command. The motor will move to the specified position at the requested slew speed and acceleration.

Case 2: Motor is moving towards specified position

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

Case 3: Motor is in the Jog Mode

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

WARNING: When the motor 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 where

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

"?" returns the current position

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	7.0
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	No		
Controller Usage	**ALL**		

EXAMPLES:

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

IT

FUNCTION: Independent Time Constant - Smoothing Function

DESCRIPTION:

The IT command filters the acceleration and deceleration functions in independent moves of JG, PR, PA type to produce a smooth velocity profile. The resulting profile, known as S-curve, 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 effect the trippoints AR and AD. The trippoints AR and 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 where

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

"?" returns the value of the independent time constant.

USAGE:

While Moving	Yes	Default Value	1.0
In a Program	Yes	Default Format	1.4
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_IT will return the value of the independent time constant.

EXAMPLES:

IT 0.8	Set independent time constants
IT ?	Return independent time constant
0.8	

IV

FUNCTION: Interrogate Interrupt

DESCRIPTION:

The IV command interrogates and clears the interrupt. When an interrupt occurs, the IV command is sent from the host PC. The meaning is read and the interrupt condition is cleared. The responses to the IV command are as follows.

Bit Number	**Condition**
7	General purpose input
6	Reserved
5	Applications programs stopped
4	User Interrupt
3	Watch Dog
2	Limit Switch
1	Position Error
0	Motion Complete

ARGUMENTS: IV n

USAGE:

While Moving	Yes	Default Value
In a Program	Yes	Default Format
Command Line	Yes	
Can be Interrogated	Yes	
Used as an Operand	No	
Controller Usage	**DMC-1410/1411/1417**	

EXAMPLES:

IV	Limit switch occurred
:4	2^2 Binary equivalent

JG

FUNCTION: Jog

DESCRIPTION:

The JG command sets the jog mode. The parameters following the JG set the slew speed and direction of motion. Use of the question mark returns the previously entered value or default value. The units of this are counts/second.

ARGUMENTS: JG n where

n is a signed number in the range 0 to +/-8,000,000 decimal (+/-12,000,000 for the DMC-1415/1416/1425) (Use JGN = n for virtual axis)

For stepper motor operation, the maximum value is +/-2,000,000 (+/-3,000,000 for the DMC-1415/1416/1425)

"?" returns the absolute value of the jog speed

USAGE:

While Moving	Yes	Default Value	25000
In a Program	Yes	Default Format	Position Format
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_JG will return the absolute value of the jog speed.

RELATED COMMANDS:

BG	Begin
ST	Stop
AC	Acceleration
DC	Deceleration
IP	Increment Position
TV	Tell Velocity

EXAMPLES:

JG 100	Set for jog mode with a slew speed of 100 counts/sec
BG	Begin Motion
JG -2000	Change speed and direction.

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

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 subroutine 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 can be nested 8 deep in the standard controller.

ARGUMENTS: JS destination,condition where

destination is a line number or label

condition is a conditional statement using a logical operator

The logical operators are:

< 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	Default Value	---
In a Program	Yes	Default Format	---
Command Line	No		
Can be Interrogated	No		
Used as an Operand	No		
Controller Usage	**ALL**		

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 controller filter. The filter transfer function is

D(z) = 4 • KP + 4 • KD(z-1)/z + KIz/2 (z-1)

For further details on the filter see the section Theory of Operation in the product manual.

ARGUMENTS: KD n where

n is an unsigned number in the range 0 to 4095.875 with a resolution of .125

"?" returns the value of the derivative constant

USAGE:

While Moving	Yes	Default Value	64
In a Program	Yes	Default Format	4.2
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_KD contains the value of the derivative constant.

RELATED COMMANDS:

KP	Proportional Constant
KI	Integral

EXAMPLES:

```
KD 100          Specify KD
KD ?            Return KD
0100.00
```

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 where

n in an unsigned number in the range 0 to 2047.875 with a resolution of 1/128

"?" returns the value of the integrator

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	4.0
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_KI contains the value of the integrator**.**

RELATED COMMANDS:

KP	Proportional Gain
KD	Derivative
ZR	Zero
IL	Integrator Limit

EXAMPLES:

KI 12	Specify integral gain
KI ?	Return value
0012	

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 in the product manual.

ARGUMENTS: KP n where

n is an unsigned number in the range 0 to 1023.875 with a resolution of .125.

“?” returns the value of the proportional constant

USAGE:

While Moving	Yes	Default Value	6
In a Program	Yes	Default Format	4.2
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_KP contains the value of the proportional constant.

RELATED COMMANDS:

KP	Proportional Gain
KI	Integral constant

KS

FUNCTION: Step Motor Smoothing

DESCRIPTION:

The KS parameter smoothes the frequency of the step motor pulses. Larger values of KS provide greater smoothness. This parameter will also increase the motion time by 3KS sampling periods. KS adds a single pole low pass filter onto the output of the motion profiler. This function smoothes out the generation of step pulses and is most useful when operating in full or half step mode.

Note: The KS will cause the step output to be delayed.

ARGUMENTS: KS n where

n is a positive integer in the range between 0.5 and 16 with a resolution of 1.

USAGE:

While Moving	Yes	Default Value	2
In a Program	Yes	Default Format	4.0
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_KS contains the value of the smoothing constant.

RELATED COMMANDS:

MT	Motor Type
IT	Independent Time Constant-Smoothing Function
MC	Motion Complete

EXAMPLES:

KS 5	Set smoothing parameter to 5.

Hint: *KS is valid for step motor only.*

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	Default Value	-
In a Program	Yes	Default Format	-
Command Line	Yes		
Controller Usage	**ALL**		

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 for the coordinate system.

USAGE:

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

OPERAND USAGE:

_LEx 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
PF	Position Formatting

EXAMPLES:

LM XY	Specify linear interpolation mode for X and Y 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: _LFn where n is the specified axis.

Note: This operand is affected by the configuration of the limit switches set by the command CN:

For CN -1:

_LFn = 1 when the limit swithch input is inactive*

_LFn = 0 when the limit switch input is active*

For CN 1:

_LFn – 0 when the limit switch input is negative*

_LFn = 1 when the limit switch input is active*

EXAMPLES:

MG _LF	Display the status of the forward limit switch
JP#A,_LF=0	Jump to label, #A, forward limit switch is activated

This is an Operand - Not a command.

LI

FUNCTION: Linear Interpolation Distance

DESCRIPTION:

The LI x,y 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. 255 returned means the buffer is empty and 255 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 XY designates linear interpolation for the X and Y axes. The speed of these axes will be computed from $VS^2=XS^2+YS^2$ where XS and YS are the speed of the X and Y axes. The controller always uses the axis specifications from LM, not LI, to compute the speed. The parameters o and p are optional and can be used to define the vector speed that is attached to the motion segment.

ARGUMENTS: LI n,n <o >p or LIX=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. o 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. p is an unsigned even integer between 0 and 12,000,000.

USAGE:

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

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 XY	Specify linear interpolation mode
LI 1000,2000	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	Default Value	-
In a Program	Yes	Default Format	-
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**ALL**		

RELATED COMMANDS:

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. Only the DMC-1425 supports the linear interpolation mode. 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 XY designates linear interpolation for the X and Y axes. The speed of these axes will be computed from $VS^2=XS^2+YS^2$, where XS and YS are the speed of the X and Y axes. The controller always uses the axis specifications from LM, not LI, to compute the speed.

ARGUMENTS: LMxx where

x is always X and Y for the DMC-1425

x = ? Returns the number of spaces available in the sequence buffer for additional LI commands.

USAGE:

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

OPERAND USAGE:

_LMx 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
CS	_CS - Sequence counter

EXAMPLES:

LM XY	Specify linear interpolation mode
VS 10000; VA 100000;VD 1000000	Specify vector speed, acceleration and deceleration
LI 100,200	Specify linear distance
LI 200,300	Specify linear distance
LE; BGS	Last vector, then begin motion

_LR*

FUNCTION: Reverse Limit Switch Operand (Keyword)

DESCRIPTION:

*The _LR operand contains the state of the reverse limit switch.

Note: This is not a command.

NOTE: This operand is affected by the configuration of the limit switches set by the command CN:

For CN-1:

_LRx = 1 when the limit switch input is inactive*

_LRx = 0 when the limit switch input is active*

For CN 1:

_LRx = 0 when the limit switch input is inactive*

_LRx = 1 when the limit switch input is active*

* The term “active” refers to the condition when at least 1 ma 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 of product manual for further details.

EXAMPLES:

MG _LR	Display the status of the reverse limit switch
JP#A,_LR=0	Jump to label, #A, when reverse limit switch is activated

This is an Operand – Not a command.

LS

FUNCTION: List

DESCRIPTION:

The LS command sends a listing of the program memory. The listing will start with the line pointed to by the first parameter, which can be either a line number or a label. If no parameter is specified, it will start with line 0. The listing will end with the line pointed to by the second parameter--again either a line number or label. If no parameter is specified, the listing will go to the last line of the program.

ARGUMENTS: LS n,m where

n,m are valid numbers from 0 to 250 (0 to 500 for the DMC-1415/1416/1425), or labels. n is the first line to be listed, m is the last.

USAGE:

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

EXAMPLES:

```
:LS #A,6                  List program starting at #A through line 6
002 #A
003 PR 500
004 BG
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 labels in memory. The listing will be in alphabetical order.

ARGUMENTS: None

USAGE:

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

RELATED COMMANDS:

LL	List Labels

EXAMPLES:

```
: LV
APPLE  = 60.0000
BOY    = 25.0000
ZEBRA = 37.0000
```

LZ

FUNCTION: Inhibit 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 is

1 to remove leading zeros

0 to disabled the leading zero removal

LZ? Returns the state of the LZ function.

USAGE:

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

EXAMPLES:

```
TE          Tell error
0004
LZ 1        Inhibit leading zeros
TE          Tell error
4
```

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-1415/1416/1425 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 F).

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	Default Value	-
In a Program	Yes	Default Format	-
Command Line	Yes		
Controller Usage	**DMC-1415/1416/1425**		

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 is completed and the encoder reaches or passes the specified position. 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 #MCTIME.

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. The actual number of steps that have been generated can be monitored by using the interrogation command TD. Note: The MC command is useful when operating with stepper motors since the step pulses can be delayed from the commanded position due to the stepper motor smoothing function, KS.

ARGUMENTS: MC

USAGE:

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

RELATED COMMANDS:

BG	Begin
AM	After Move
TW	Timeout

EXAMPLES:

#MOVE	Program MOVE
PR 5000	Position relative moves
BG	Start the axis
MC	After the move is complete
SB1	Set output 1 to logic 1
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. The MF command can also be used when the encoder is the master and not under servo control.

ARGUMENTS: MF n where

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

USAGE:

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

RELATED COMMANDS:

AD	Trippoint for after Relative Distance
MR	Reverse motion to position
AP	After Absolute Position

EXAMPLES:

```
#TEST                        Program B
DP0                          Define zero
JG 1000                      Jog mode (speed of 1000 counts/sec)
BG                           Begin move
MF 2000                      After passing the position 2000
V1=_TP                       Assign V1 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 {Pn},{Ex},{U} "m", {^n}, V {Fm.n or $m,n} {N} {Sn}

"m" is a text message including letters, numbers, symbols or <ctrl>G. Make sure that maximum line length is not exceeded (40 characters DMC-1410/1411/1412/1414, 80 characters DMC-1415/1416/1425).

{^n} is an ASCII character specified by the value n

V is a variable name or array element where the following specifiers can be used for formatting:

{Fm.n} Display variable in decimal format with m digits to left of decimal, and n to the right.

{$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 thru 6

{N} Suppress carriage return line feed.

{Pn} (DMC-1412/1414 only) Specifies which serial port to send the message. 1 = main port, 2 = auxiliary port. Defaults to 1 if not specified.

{U} for USB port

{Ex}for Ethernet and ‘x’ specifies the Ethernet handle (A,B,C,D,E,F or H).

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	Default Value	-
In a Program	Yes	Default Format	Variable Format
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	No		
Controller Usage	**ALL**		

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}, {^30}, {^37}, {N} Sends carriage return, characters 0 and 7 followed by no carriage return line feed command to the port.

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

USAGE:

While Moving	No	Default Value	0
In a Program	Yes	Default Format	1.0
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_MO will return the state of the motor, 0 = servo loop on and 1 = servo loop off.

RELATED COMMANDS:

SH	Servo Here

EXAMPLES:

MO	Turn off motor
SH	Turn motor on
Bob=_MO	Sets Bob equal to the 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. The MR command can also be used when the encoder is the master and not under servo control.

ARGUMENTS: MR n where

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

USAGE:

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

RELATED COMMANDS:

AD	Trippoint for after Relative Distance
MF	Forward motion to position
AP	Trippoint After absolute position

EXAMPLES:

#TEST	Program B
DP0	Define zero
JG 1000	Jog mode (speed of 1000 counts/sec)
BG	Begin move
MR -3000	After passing the position -3000
V1=_TP	Assign current position to variable V1.
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 the specified motor is driven independently by an external device.*

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 servo motors 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 servo motors, and inverts logic level of the pulse train, for step motors.

ARGUMENTS: MT n where n is:

1	Servo motor
-1	Servo motor reversed polarity
2	Step motor with active low step pulses
-2	Step motor with active high step pulses
2.5	Step motor with reversed direction and active low step pulses
-2.5	Step motor with reversed direction and active high step pulses

"?" returns the value of the motor type

USAGE:

While Moving	Yes	Default Value	1
In a Program	Yes	Default Format	1.0
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_MT contains the value of the motor type.

RELATED COMMANDS:

EXAMPLES:

```
MT 1            Configure x as servo
MT ?            Interrogate motor type
V=_MT           Assign motor type to variable
```

NB

FUNCTION: Notch Bandwidth

DESCRIPTION:

The NB command sets real part of the notch poles

ARGUMENTS: NB n,n or NBX=n where

n is ranges from 0 Hz to $\frac{1}{(16 \cdot TM)}$

USAGE:

While Moving	Yes	Default Value	0.5
In a Program	Yes	Default Format	
Command Line	Yes		
Controller Usage	**DMC-1415/1416/1425**		

RELATED COMMANDS:

NOTE: **This command is only valid for the DMC – 1415, 1416, and 1425 controllers.**
NF

NF	Notch Filter
NZ	Notch Zeros

EXAMPLES:

_NBX = 10	Sets the real part of the notch pole to 10/2 Hz
NOTCH = _NBX	Sets the variable "NOTCH" equal to the notch bandwidth value for the X axis

NOTE: This command is only valid for the DMC – 1415, 1416, and 1425 controllers.

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 or NFX=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	Default Value	0
In a Program	Yes	Default Format	
Command Line	Yes		
Controller Usage	**DMC-1415/1416/1425**		

OPERAND USAGE:

_NFx 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 Y axis to 20 Hz

NOTE: This command is only valid for the DMC – 1415, 1416, and 1425 controllers.

NO

FUNCTION: No Operation

DESCRIPTION:

The NO command performs no action in a sequence, but can be used as a comment in a program. After the NO, characters can be given to form a program comment up to the maximum line length of the controller. This helps to document a program.

An apostrophe ' may also be used instead of the NO to document a program. This feature is only supported on the DMC-1415/1416/1425.

ARGUMENTS: NO m where

m is any group of letter, number, symbol or <cntrl>G

USAGE:

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

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 or NZX=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	Default Value	0.5
In a Program	Yes	Default Format	
Command Line	Yes		
Controller Usage	**DMC-1415/1416/1425**		

OPERAND USAGE:

_NZx contains the value of the Notch filter zero for the specified axis.

RELATED COMMANDS:

NB	Notch Bandwidth

"NOTE: *This command is only valid for the DMC – 1415, 1416, and 1425 controllers.* NF"

NF	Notch Filter

EXAMPLES:

NZX = 10	Sets the real part of the notch zero to 10/2 Hz

NOTE: This command is only valid for the DMC – 1415, 1416, and 1425 controllers.

OB

FUNCTION: Output Bit

DESCRIPTION:

The OB n, logical expression command defines output bit n 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.

ARGUMENTS: OB n, expression where

n is output bit

expression is any valid logical expression, variable or array element.

USAGE:

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

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, otherwise set 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 a duration of approximately 600 nanoseconds and is available at the output compare signal (labeled CMP/ICOM on the ICM-1460). For a one shot compare, 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.

Note: The OC function requires that the main encoder and auxiliary encoders be configured exactly the same (see the command, CE). For example: CE 0, CE 5, CE 10, CE 15.

Note: If the ICM-1460 has the –Opto Input option, the output compare is not brought out to the CMP/ICOM terminal. If the output compare is to be used, the pin will need to be accessed directly from the 37 Pin-D cable.

ARGUMENTS: OCX = m, n   where

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 for one shot

o= one shot when moving in the forward direction

-65536 one shot when moving in the reverse direction

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.

Note: 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	Default Value	-
In a Program	Yes	Default Format	-
Command Line	Yes		
Controller Usage	**DMC-1415/1416**		

OPERAND USAGE:

_OCx contains the state of the OC function

_OCx = 0 : OC function has been enabled but not generated any pulses.

_OCx = 1: OC function not enables or has generated the first output pulse.

EXAMPLES:

OCX=300,100	Select X 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 the position error exceeds the limit specified by the ER command or an abort occurs from either the abort input or an AB command.

The OE command, in addition to shutting off the motor command, will toggle the amplifier enable signal.

ARGUMENTS: OE n where

n may be 0 or 1. 0 disables function. 1 enables off-on-error.

"?" returns the state of the off-on-error function

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	1.0
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_OE contains the status of the off-on-error function.

RELATED COMMANDS:

ER	Error limit
SH	Servo Here
#POSERR	Error Subroutine

EXAMPLES:

OE 1	Enable OE
OE 0	Disable OE

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. If the PID values are zero, then the output voltage will be the OF value. The OF command works in servo mode only.

ARGUMENTS: OF n where

n is a signed number in the range -9.998 to 9.998 volts with resolution of .0003.

"?" returns the offset

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	1.4
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_OF contains the offset.

EXAMPLES:

```
OF 1              Set offset to 1
OF ?              Return offset
1.0000
```

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 (DMC-1410/1411/1412/1414/1417): OP m,n where

m is an integer in the range 0 to 7 and is the decimal representation of the 3 output bits.

n is an integer in the range 1 to 3 decimal and is used to specify the number of bits effected starting with the LSB. For example, if n=2, only outputs 1 and 2 will be changed by OP m. If the n parameter is not specified, all bits will be changed.

ARGUMENTS (DMC-1415/1416/1425): OP m,a,b,c,d where

m is an integer in the range 0 to 7 and is the decimal representation of the general output bits.

a,b,c,d represent the extended I/O in consecutive groups of 16 bits, (values from 0 to 65535). Arguments that are given for I/O points which 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, where n is any of the above arguments.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes
Used as an Operand	Yes
Controller Usage	**ALL**

DEFAULTS:

Default Value	0
Default Format	3.0

OPERAND USAGE (DMC-1410/1411/1412/1414/1417):

_OP contains the status of the outputs.

OPERAND USAGE (DMC-1415/1416/1425):

_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 Byte

EXAMPLES:

OP 0	Clear Output Port – all bits
OP 7	Set outputs 1,2 and 3
MG _OP0	Returns the first parameter "m" (DMC-1415/1416/1425 only)
MG _OP1	Returns the second parameter "a" (DMC-1415/1416/1425 only)

@OUT[n]

FUNCTION: Read digital output

DESCRIPTION:

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

ARGUMENTS: @IN[n] where

n is an unsigned integer in the range 1 to 80

USAGE: **DEFAULTS:**

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

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
```

P1CD P2CD

FUNCTION: Serial port 1 or serial port 2 code

DESCRIPTION:

DMC-21x2/3: P1CD returns the status of the serial port when in the operator data entry mode (CI,1)

DMC-2xx0: P2CD returns the status of the auxiliary serial port (port 2)

USAGE: **DEFAULTS:**

While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	-
Command Line	Yes		
Controller Usage	**DMC-1412 / 4 ONLY**		

RELATED COMMANDS:

P1CH P2CH	Serial port 1/2 character
P1NM P2NM	Serial port 1/2 number
P1ST P2ST	Serial port 1/2 string
CI	Configure #COMINT (and set operator data entry mode)
CC	Configure serial port 2
#COMINT	Communication interrupt automatic subroutine

EXAMPLES:

```
:^R^V
DMC2240 Rev 1.0o
:^R^S

:CC 9600,0,0,0
:MG "TEST" {P2} ;'send a message to the hand
terminal
:MG P2CD ;'no characters entered on hand terminal
 0.0000
:MG P2CD ;'the number 6 was pushed on the hand
terminal
 1.0000
:MG P2CD ;'enter key pushed on hand terminal
 3.0000
:MG P2CD ;'the character B was pushed (shift f2)
      then enter
2.0000
```

P1CH P2CH

FUNCTION: Serial port 1 or serial port 2 character

DESCRIPTION:

P1CH returns the last character sent to the serial port when in the operator data entry mode (CI,1)

P2CH returns the last character sent to the auxiliary serial port (port 2)

USAGE: **DEFAULTS:**

While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	-
Command Line	Yes		
Controller Usage	**DMC-1412 / 4 ONLY**		

RELATED COMMANDS:

P1CD P2CD	Serial port 1/2 code
P1NM P2NM	Serial port 1/2 number
P1ST P2ST	Serial port 1/2 string
CI	Configure #COMINT (and set operator data entry mode)
CC	Configure serial port 2
#COMINT	Communication interrupt automatic subroutine

EXAMPLES:

```
:^R^V
DMC2240 Rev 1.0o
:^R^S

:CC 9600,0,0,0
:MG "TEST" {P2} ;'send a message to the hand
      terminal
:MG P2CH {S1} ;'the 6 button was pushed on the hand
      terminal
6
:
```

P1NM P2NM

FUNCTION: Serial port 1 or serial port 2 number

DESCRIPTION:

P1NM and P2NM convert from ASCII (e.g. “1234”) to binary so that a number can be stored into a variable and math can be performed on it. Numbers from -2147483648 to 2147483647 can be processed.

P1NM returns the last number (followed by carriage return) sent to the serial port when in the operator data entry mode (CI,1)

P2NM returns the last number (followed by carriage return) sent to auxiliary serial port (port 2)

USAGE: **DEFAULTS:**

While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	-
Command Line	Yes		
Controller Usage	**DMC-1412 / 4 ONLY**		

RELATED COMMANDS:

P1CD P2CD	Serial port 1/2 code
P1CH P2CH	Serial port 1/2 character
P1ST P2ST	Serial port 1/2 string
CI	Configure #COMINT (and set operator data entry mode)
CC	Configure serial port 2
#COMINT	Communication interrupt automatic subroutine

EXAMPLES:

```
:^R^V
DMC2240 Rev 1.0o
:^R^S

:CC 9600,0,0,0
:MG "TEST" {P2} ;'send a message to the hand
      terminal
:x = P2NM ;'the 1, 2, 3, <enter> buttons were pushed
:MG x
 123.0000
:
```

P1ST P2ST

FUNCTION: Serial port 1 or serial port 2 string

DESCRIPTION:

P1ST returns the last string (followed by carriage return) sent to the serial port when in the operator data entry mode (CI,1)

P2ST returns the last string (followed by carriage return) sent to auxiliary serial port (port 2)

NO MORE THAN SIX CHARACTERS CAN BE ACCESSED.

USAGE: **DEFAULTS:**

While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	-
Command Line	Yes		
Controller Usage	**DMC-1412 / 4 ONLY**		

RELATED COMMANDS:

P1CD P2CD	Serial port 1/2 code
P1CH P2CH	Serial port 1/2 character
P1NM P2NM	Serial port 1/2 number
CI	Configure #COMINT (and set operator data entry mode)
CC	Configure serial port 2
#COMINT	Communication interrupt automatic subroutine

EXAMPLES:

```
:CC 9600,0,0,0
:MG "TEST" {P2} ;'send a message to the hand terminal
:MG P2ST {S3} ;'the characters ABC were entered
ABC
:
```

#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*

PA

FUNCTION: Position Absolute

DESCRIPTION:

The PA command will set the final destination of the next move. The position is referenced to the absolute zero. If a ? is used, then the current destination (current command position if not moving, destination if in a move) is returned. For each single move, the largest position move possible is +/- 2147483647. Units are in quadrature counts.

ARGUMENTS: PA n where

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

n = ? Returns the commanded position

USAGE:

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

OPERAND USAGE:

_PA contains current command position if not moving, start position if given during motion.

RELATED COMMANDS:

PR	Position relative
SP	Speed
AC	Acceleration
DC	Deceleration
BG	Begin

EXAMPLES:

:PA 400	X-axis will go to 400 counts
:PA ?	Returns the current commanded position
0000000	
:BG	Start the move
:PA 700	X-axis will go to 700 on the next move
:BG	

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 ?	PA ?
DE ?	PR ?
DP ?	TE
FL ?	
IP ?	
TP	

ARGUMENTS: PF m.n where

m is an integer between -8 and 10. The negative sign for m specifies hexadecimal representation.

n is an integer between 0 and 4

PF? Returns the value of m

USAGE:

While Moving	Yes	Default Value	10.0
In a Program	Yes	Default Format	10.0
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_PF contains the value of position format parameter.

EXAMPLES:

:TP	Tell position
0000000000	Default format
:PF 5.2	Change format to 5 digits of integers and 2 of fractions
:TP	Tell Position
00021.00	
PF-5.2	New format Change format to hexadecimal*
:TP	Tell Position
$00015.00	Report in hex

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. If a ? is used, then the current incremental distance is returned (even if it was set by a PA command). Units are in quadrature counts.

ARGUMENTS: PR n where

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

“?” returns the current incremental distance

USAGE:

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

OPERAND USAGE:

_PR will return the current incremental distance.

RELATED COMMANDS:

BG	Begin
AC	Acceleration
DC	Deceleration
SP	Speed
IP	Increment Position

EXAMPLES:

:PR 100	On the next move the X-axis will go 100 counts,
:BG	
:PR ?	Return relative distances
0000000100	

QD

FUNCTION: Download Array

DESCRIPTION:

The QD command transfers array data from the host computer to the DMC-1400. 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 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)

USAGE:

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

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 details of the status information are described in Chapter 4 of the user's manual.

ARGUMENTS: QR xx where

x is X,Y,Z,W,A,B,C,D,E,F,G,H or I or any combination to specify the axis, axes, sequence, or I/O status

I represents the status of the I/O

Chapter 4 of the user manual provides the definition of the data record information.

USAGE:

While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	-
Command Line	Yes		
Controller Usage	**DMC-1415/1416/1425**		

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 DMC-1400 to a host computer. 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	Default Value	0
In a Program	Yes	Default Format	Position Format
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	No		
Controller Usage	**ALL**		

RELATED COMMANDS:

QD	Download array

QZ

FUNCTION: Return Data Record information

DESCRIPTION:

The QZ command is an interrogation command that returns information regarding the Data Record (DMC-1415/1416/1425). 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	Default Value	---
In a Program	Yes	Default Format	
Command Line	Yes		
Controller Usage	**DMC-1415/1416/1425**		

RELATED COMMANDS:

QR	Data Record

RA

FUNCTION: Record Array

DESCRIPTION:

The RA command selects one or two arrays for automatic data capture. The selected arrays must have been 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 [] where

n,m are dimensioned arrays as defined by DM command. The [] contain nothing.

USAGE:

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

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 _TP	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.

RC? returns status of recording. '1' if recording, '0' if not recording.

USAGE:

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

OPERAND USAGE:

_RC contains status of recording '1' if recording, '0' if not recording.

RELATED COMMANDS:

DM	Dimension Array
RD	Record Data
RA	Record Array Mode

EXAMPLES:

```
#RECORD                 Record
DM Torque[1000]         Define Array
RA Torque[]             Specify Record Mode
RD _TT                  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:

DATA TYPE	MEANING
_DE	2nd encoder
_TP	Position
_TE	Position error
_SH	Commanded position
_RL	Latched position
_TI	Inputs
_OP	Outputs
_TS	Switches, only 0-4 bits valid
_SC	Stop code
_TT	Tell torque
_TVn	Filtered velocity. (Note: will be 65 times greater than TV command)

ARGUMENTS: RD m1, m2 where

the arguments are the data type to be captured using the record array feature. The order is important. Each of the two data types corresponds with the array specified in the RA command.

USAGE:

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

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 ERRORX[50]	Define array
RA ERRORX[]	Specify record mode
RD _TE	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. No RE needed after ZS.

ARGUMENTS: RE n where

n = 0 or 1

0 clears the interrupted trippoint

1 restores state of trippoint

USAGE:

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

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.*

A motion trippoint like MF or MR requires the axis to be actively profiling in order to be restored with the RE1 command.

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: **DEFAULTS:**

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

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 or 1

0 clears interrupt trippoint

1 restores trippoint

USAGE:

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

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.*

A motion trippoint like MF or MR requires the axis to be actively profiling in order to be restored with the RI1 command.

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 Input 1. The armed state of the latch can be configured using the CN command.

ARGUMENTS: RL

USAGE:

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

OPERAND USAGE:

_RL contains the latched position.

RELATED COMMAND:

AL	Arm Latch

EXAMPLES:

JG 5000	Set up to jog
BG	Begin jog
AL	Arm the latch; assume that after about 2 seconds, input goes low
RL	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:		**DEFAULTS:**	
While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	-
Command Line	Yes		
Controller Usage	**ALL**		

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.

ARGUMENTS: RP

USAGE:

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

OPERAND USAGE:

_RP contains the commanded reference position.

RELATED COMMAND:

TP	Tell Position
TE	Tell Error

Note: The relationship between RP, TP and TE is that the position error, _TE, equals the difference between the reference position, _RP and the actual position, _TP.

EXAMPLES:

:PF 7	Position format of 7
0:RP	
0000200	Return reference position
PF-6.0	Change to hex format
RP	
$0000C8	Return in hex
Position=_RP	Assign the variable, Position, the value of RP

HINT: *RP command is useful when operating step motors since it provides the commanded position in steps when operating in stepper mode.*

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 EEROM.

USAGE:

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

OPERAND USAGE:

_RS contains the power-up error:	bit 0 for variable checksum error
	bit 1 for parameter checksum error
	bit 2 for program checksum error
	bit 3 for master reset error (no program to execute)

EXAMPLES:

RS	Reset the controller

<control>R<control>S

FUNCTION: Master Reset

DESCRIPTION:

The Master Reset command resets the DMC-1400 to factory default settings and erases EEPROM.

A master reset can also be performed by installing a jumper on the DMC-14XX 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	Default Value	-
In a Program	No	Default Format	-
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	No		
Controller Usage	**ALL**		

<control>R<control>V

FUNCTION: Revision Information

DESCRIPTION:

The Revision Information command causes the controller to return the firmware revision information.

USAGE:

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

SA

FUNCTION: Send Command

DESCRIPTION:

SA sends a command from one Galil controller to another controller or IOC-7007 board over Ethernet. Any command can be sent to another Galil card and will be interpreted by the card as a "local" command.

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 other Galil Ethernet controller.

arg is a command, number, Galil controller or IOC-7007 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 "KI 1,2"

USAGE:

While Moving	Yes	Default Value	--
In a Program	Yes	Default Format	--
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**DMC-1415/1416/1425**		

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:

IH	Set Internet Handles

EXAMPLES:

SAA="KI",1,2;WT5	Sends the command to handle A (slave controller): KI 1,2
SAA="KI?,?"	Sends the command to handle A (slave controller): KI?,?
MG _SAA1	Display the content of the operand _SAA (first response to KI?,? command)
: 1	
MG _SAA2	Display the content of the operand _SAA (2nd response to KI?,? command)
: 2	

NOTE: A wait statement (eg. WT5) should be inserted between successive calls to SA.

SA n

FUNCTION: Serial Address

DESCRIPTION:

SA assigns the address of a serial controller in a daisy-chain network. See Chapter 4 in the DMC-1412/1414 user manual for more information on daisy-chaining.

ARGUMENTS: SA n

where n is a number between 0 and 7 representing the address of the controller. This command may be saved by issuing the BN command.

USAGE:

While Moving	Yes	Default Value	--
In a Program	Yes	Default Format	--
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	No		
Controller Usage	**DMC-1412/1414**		

OPERAND USAGE:

_SA gives the serial address of the controller

EXAMPLES:

SA7	Assigns the address 7 to the current controller
%7	Talk only to controller 7
PR500	Specify X distance on controller 7
%0	Talk only to controller 0
PR500	Specify X distance on controller 0
!BG	Begin motion on all controllers

SB

FUNCTION: Set Bit

DESCRIPTION:

The SB command sets one of three bits on the output port.

Note: When using Modbus devices (DMC-1415/1416/1425 ONLY), the I/O points of the Modbus devices are calculated using the following formula:

n = (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 F.

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.

ARGUMENTS: SB n where

n is an integer in the range 1 to 3 decimal.

USAGE:

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

RELATED COMMAND:

OP	Configure output port
CB	Clear Bit

EXAMPLES:

SB 3	Set output line 3
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 software limit, FL	11	Stopped by selective Abort Input
3	Decelerating or stopped by REV limit switch or software 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

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	3.0
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_SC contains the value of the stop code.

EXAMPLES:

Tom=_SC	Assign the Stop Code to variable Tom

SH

FUNCTION: Servo Here

DESCRIPTION:

The SH command 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

USAGE:

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

RELATED COMMANDS:

MO	Motor-off

EXAMPLES:

```
SH                 Servo motor
```

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 -2147483648 to 2147483647.

USAGE: **DEFAULTS:**

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

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:		**DEFAULTS:**	
While Moving	Yes	Default Value	1
In a Program	No		
Command Line	Yes		
Controller Usage	**ALL CONTROLLERS**		

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 for independent moves. The parameters input will be rounded down to the nearest factor of 2 and the units of the parameter are in counts per second. Note: Negative values will be interpreted as the absolute value.

ARGUMENTS: SP n where

n is an unsigned even integer in the range 0 to 8,000,000 for servo motors on the DMC-1410/1411/1412/1414/1417 (0 to 12,000,000 for servo motors on the DMC-1415/1416/1425).

OR

0 to 2,000,000 for stepper motor operation on the DMC-1410/1411/1412/1414/1417 (0 to 3,000,000 for stepper motor operation on the DMC-1415/1416/1425).

“?” returns the speed

USAGE:

While Moving	Yes	Default Value	25000
In a Program	Yes	Default Format	Position Format
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_SP contains the current speed setting.

RELATED COMMANDS:

AC	Acceleration
DC	Deceleration
PR	Position Relation
BG	Begin

EXAMPLES:

PR 2000	Specify position relative move
SP 5000	Specify speeds
BG	Begin motion of all axes
AM	After motion is complete

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

RELATED COMMANDS:

@ABS	Absolute value

EXAMPLES:

```
:MG @SQR[2]
 1.4142
:MG @SQR[-2]
 1.4142
  :
```

ST

FUNCTION: Stop

DESCRIPTION:

The ST command stops commanded motion. The motor will come to a decelerated stop. ST sent from the host with no arguments will stop motion and any programs that are running on the controller.

ARGUMENTS: ST

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	No		

RELATED COMMANDS:

BG	Begin Motion
AM	Wait for motion end
DC	Deceleration rate

EXAMPLES:

ST	Stops 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 -2147483648 to 2147483647.

USAGE: **DEFAULTS:**

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

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 program
Bit 6	DMA Active (when available)
Bit 5	Contouring
Bit 4	Executing error or limit switch routine
Bit 3	Input interrupt enabled
Bit 2	Executing input interrupt routine
Bit 1	0 (Reserved)
Bit 0	Echo on

ARGUMENTS: None

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	1.0
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_TB contains the status byte.

EXAMPLES:

TB	Tell status information from the controller
65	Executing program and echo 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. Entering the TC command will provide the user with a code as to the reason. After TC has been read, it is set to zero. TC 1 returns the text message as well as the numeric code.

ARGUMENTS: TC n

n=0 returns code only

n=1 returns code and message

CODE	EXPLANATION	CODE	EXPLANATION
1	Unrecognized command	60	Download error - line too long or too many lines
2	Command only valid from program	61	Duplicate or bad label
3	Command not valid in program	62	Too many labels
4	Operand error	63	IF statement without ENDIF
5	Input buffer full	65	IN command must have a comma
6	Number out of range	66	Array space full
7	Command not valid while running	67	Too many arrays or variables
8	Command not valid when not running	68	Not valid from USB Port
9	Variable error	71	IN only valid in task #0
10	Empty program line or undefined label	80	Record mode already running
11	Invalid label or line number	81	No array or source specified
12	Subroutine more than 16 deep	82	Undefined Array
13	JG only valid when running in jog mode	83	Not a valid number
14	EEPROM check sum error	84	Too many elements
15	EEPROM write error	90	Only X Y Z W valid operand
16	IP incorrect sign during position move or IP given during forced deceleration	96	SM jumper needs to be installed for stepper motor operation
17	ED, BN and DL not valid while program running	97	Bad Binary Command Format
18	Command not valid when contouring	98	Binary Commands not valid in application program
19	Application strand already executing	99	Bad binary command number
20	Begin not valid with motor off	100	Not valid when running ECAM
21	Begin not valid while running	101	Improper index into ET (must be 0-256)
22	Begin not possible due to Limit Switch	102	No master axis defined for ECAM
24	Begin not valid because no sequence defined	103	Master axis modulus greater than 256∗EP value
25	Variable not given in IN command	104	Not valid when axis performing

			ECAM
28	S operand not valid	105	EB1 command must be given first
29	Not valid during coordinated move	110	No hall effect sensors detected
30	Sequence segment too short	111	Must be made brushless by BA command
31	Total move distance in a sequence > 2 billion	112	BZ command timeout
32	More than 511 segments in a sequence	113	No movement in BZ command
33	VP or CR commands cannot be mixed with LI commands	114	BZ command runaway
41	Contouring record range error	118	Controller has GL1600 not GL1800
42	Contour data being sent too slowly	120	Bad Ethernet transmit
46	Gear axis both master and follower	121	Bad Ethernet packet received
50	Not enough fields	122	Ethernet input buffer overrun
51	Question mark not valid	123	TCP lost sync
52	Missing " or string too long	124	Ethernet handle already in use
53	Error in {}	125	No ARP response from IP address
54	Question mark part of string	126	Closed Ethernet handle
55	Missing [or []	127	Illegal Modbus function code
56	Array index invalid or out of range	128	IP Address not valid
57	Bad function or array	130	Illegal IOC Command
58	Not a valid Command Operand (i.e._GNX)	131	Serial Port Handshake timeout
59	Mismatched parentheses		

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	3.0
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_TC contains the value of the error code.

EXAMPLES:

:GF32	Bad command
?TC	Tell error code
1	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-14x5 / 6 ONLY**

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. The auxiliary encoder is not available if the controller is set up for stepper.

When operating with stepper motors, the TD command returns the number of counts that have been output by the controller.

ARGUMENTS: TD

USAGE:

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

OPERAND USAGE:

_TD contains the dual encoder position.

RELATED COMMANDS:

DE	Dual Encoder

EXAMPLES:

:PF 7	Position format of 7
:TD	Return Dual encoder
0000200	
DUAL=_TD	Assign the variable, DUAL, the value of TD

TE

FUNCTION: Tell Error

DESCRIPTION:

This command returns the current position error of the motor. The range of possible error is +/-2147483647. The Tell Error command is not valid for step motors since they operate open-loop.

ARGUMENTS: TE

USAGE:

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

RELATED COMMANDS:

ER	Error Limit
#POSERR	Error Subroutine

EXAMPLES:

TE	Return position error
00005	
Error=_TE	Sets the variable, Error, with the 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 and IP address.

ARGUMENTS: None

USAGE:

While Moving	Yes	Default Value	--
In a Program	Yes	Default Format	--
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	No		
Controller Usage	**DMC-1415/1416/1425**		

RELATED COMMANDS:

IH	Internet Handle
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
IHB TCP PORT 1061 TO IP ADDRESS 10,51,0,89 PORT 1001
IHC TCP PORT 1012 TO IP ADDRESS 10,51,0,93 PORT 1002
IHD TCP PORT 1023 TO IP ADDRESS 10,51,0,93 PORT 1003
IHE TCP PORT 1034 TO IP ADDRESS 10,51,0,101 PORT 1004
IHF TCP PORT 1045 TO IP ADDRESS 10,51,0,101 PORT 1005
```

TI

FUNCTION: Tell Inputs

DESCRIPTION:

This command returns the state of all 7 general digital inputs or 3 inputs for the DMC-1425. Response is a decimal number which when converted to binary represents the status of all 7 digital inputs.

BIT	TI
Bit 6	Input 7
Bit 5	Input 6
Bit 4	Input 5
Bit 3	Input 4
Bit 2	Input 3
Bit 1	Input 2
Bit 0	Input 1

ARGUMENTS: TI

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	3.0
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_TI contains the status byte of the input block. Note that the operand can be masked to return only specified bit information - see section on Bitwise operations

EXAMPLES:

TI	
08	Input 4 is high, others low
TI	
00	All inputs low
Input =_TI	Sets the variable, Input, with the TI value
TI	
127	All inputs high

TIME*

FUNCTION: Time Operand (Keyword)

DESCRIPTION:

*The TIME operand contains 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 microseconds. 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 with the other operands.

USAGE:

Used as an Operand	Yes	Format	TIME
Controller Usage	**ALL**		

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 where

n is an unsigned number in the range 0 to 9.998 volts with resolution of 0.0003.

USAGE:

While Moving	Yes	Default Value	9.9988
In a Program	Yes	Default Format	1.4
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_TL contains the value of the torque limit.

EXAMPLES:

```
TL 1                    Limit X-axis to 1volt
TL ?                    Return limit
1.0000
```

TM

FUNCTION: Update Time

DESCRIPTION:

The TM command sets the sampling period of the control loop. Changing the sampling period will uncalibrate the speed and acceleration parameters. A negative number turns off the servo loop. The units of this command are μsec.

ARGUMENTS: TM n where

n is an integer in the range 375 to 20000 decimal with resolution of 125 microseconds. For the DMC-1415/1416/1425 the range is from 250 to 20000.

"?" returns the value of the sample clock

USAGE:

While Moving	Yes	Default Value	1000
In a Program	Yes	Default Format	5.0
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_TM contains the value of the sample time.

EXAMPLES:

TM -1000	Turn off internal clock
TM 2000	Set sample rate to 2000 μsec (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

ARGUMENTS: TP

USAGE:

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

OPERAND USAGE:

_TP contains the current position value.

EXAMPLES:

:PF 7	Position format of 7
:TP	Return position
0000200	
PF-6.0	Change to hex format
TP	Return in hex
$0000C8	
Position=_TP	Assign the variable, Position, the value of TP

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 or 1

0 disables function

1 enables function

USAGE:

While Moving	Yes	Default Value	TR0
In a Program	Yes	Default Format	---
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	Yes		
Controller Usage	**ALL**		

TS

FUNCTION: Tell Switches

DESCRIPTION:

TS returns the state of the Home switch, Forward and 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	Error limit exceeded if high
Bit 5	Motor off if high
Bit 4	Undefined
Bit 3	Forward Limit inactive if high
Bit 2	Reverse Limit inactive if high
Bit 1	State of home switch
Bit 0	Latch not armed if high

Note: The value for bits 1, 2 and 3 depend on the limit switch and home switch configuration (see CN command). For active low configuration (default), these bits are '1' when the switch is inactive and '0' when active. For active high configuration, these bits are '0' when the switch is inactive and '1' when active.

ARGUMENTS: TS xx where

x is X or Y or both, an argument is only required by the DMC-1425

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	3.0
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_TS contains the current status of the switches.

EXAMPLES:

V1=_TSX	Assigns value of TSX to the variable V1
V1=	Interrogate value of variable V1
015 (returned value)	Decimal value corresponding to bit pattern 00001111
	X axis not in motion (bit 7 has value of 0)
	X axis error limit not exceeded (bit 6 has value of 0)
	X axis motor is on (bit 5 has value of 0)
	X axis forward limit is inactive (bit 3 has value of 1)
	X axis reverse limit is inactive (bit 2 has value of 1)
	X axis home switch is high (bit 1 has value of 1)
	X axis latch is not armed (bit 0 has value of 1)

TT

FUNCTION: Tell Torque

DESCRIPTION:

The TT command reports the value of the analog servo command output signal, which is a number between -9.998 and 9.998 volts.

ARGUMENTS: TT

USAGE:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	1.4
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_TT contains the value of the torque.

RELATED COMMANDS:

TL	Torque Limit

EXAMPLES:

V1=_TT	Assigns value of TT to variable, V1
TT	Report torque
-0.2843	Torque is -.2843 volts

TV

FUNCTION: Tell Velocity

DESCRIPTION:

The TV command returns the actual velocity in units of quadrature count/s. The value returned includes the sign.

ARGUMENTS: TV

No argument will provide the velocity for all axes.

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	7.0
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_TV contains the value for the velocity.

EXAMPLES:

```
VELX=_TV             Assigns value of velocity to the variable VELX
TV                   Returns the 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 n 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 where

n specifies timeout in msec range 0 to 32767 msec, -1 disables the timeout

“?” returns the timeout in msec for the MC command

USAGE:

While Moving	Yes	Default Value	32766
In a Program	Yes	Default Format	
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_TW contains the timeout in msec for the MC command .

RELATED COMMANDS:

MC	Motion Complete - "In Position"

UI

FUNCTION: User Interrupt

DESCRIPTION:

The UI command causes an interrupt on the selected IRQ line. There are 16 user interrupts where UI n, n = 0 through 15. Prior to using the UI command, one IRQ line must be jumpered on the DMC-141X. An interrupt service routine must also be incorporated in your host program. The IV command should be sent from this routine for interrupt status and to re-enable the interrupt. Refer to Chapter 4 in the product manual for details.

ARGUMENTS: UI n where

n is an integer between 0 and 15.

USAGE:

While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	---
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	No		
Controller Usage	**DMC-1410/1411/1417**		

EXAMPLES:

#I	Label
PR 10000	Position relative
SP 5000	Speed
BG	Begin motion
AS	Wait for at speed
UI 1	Send interrupt 1
EN	End program

This program sends an interrupt to the selected IRQ line. The host should have an interrupt service routine written to handle the interrupts. The IV command may be used to return interrupt information and re-enable the interrupts.

UL

FUNCTION: Upload

DESCRIPTION:

The UL command transfers data from the DMC-141X to a host computer. 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:

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

OPERAND USAGE:

When used as an operand, _UL gives the number of available variables. The total number of variables is 126.

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 n where

n 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.

n = ? Returns the value of the vector acceleration for the coordinate plane.

USAGE:

While Moving	Yes	Default Value	262144
In a Program	Yes	Default Format	Position Format
Command Line	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_VAx 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

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 n where

n 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.

n = ? Returns the value of the vector deceleration for the coordinate plane.

USAGE:

While Moving	No	Default Value	262144
In a Program	Yes	Default Format	Position Format
Command Line	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_VDx contains the value of the vector deceleration for the coordinate system.

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

EXAMPLES:

```
#VECTOR           Vector Program Label
VMXY              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	Default Value	-
In a Program	Yes	Default Format	-
Command Line	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_VEx contains the length of the vector in counts for the coordinate system.

RELATED COMMANDS:

VM	Vector Mode
VS	Vector Speed
VA	Vector Acceleration
VD	Vector Deceleration
CR	Circle
VP	Vector PGosition
BG	Begin Sequence
CS	Clear Sequence

EXAMPLES:

VM XY	Vector move in XY
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 allows the variables and arrays to be formatted for number of digits before and after the decimal point. When displayed, the value m represents the number of digits before the decimal point, and the value n represents the number of digits after the decimal point. When in hexadecimal, the string will be preceded by a $. 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).

ARGUMENTS: VF m.n where

m and n are unsigned numbers in the range 0<m<10 and 0<n<4. A negative m specifies hexadecimal format.

VF? returns the value of the format for variables and arrays.

USAGE:

While Moving	Yes	Default Value	10.4
In a Program	Yes	Default Format	2.1
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_VF contains the value of the format for variables and arrays.

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 command is only used for the DMC-1425 two axis controller.

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. Vector Motion can be specified for one axis by specifying 2nd parameter, m, as N (the designation for the virtual axis). Specifying one axis is useful for obtaining sinusoidal motion on 1 axis.

USAGE:

While Moving	No	Default Value	X,Y
In a Program	Yes	Default Format	-
Command Line	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_VM contains instantaneous commanded vector velocity for the coordinate system.

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

EXAMPLES:

VM XY	Specify coordinated mode for X,Y
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 straight line 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 scale factor set using the command ES.

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 □ 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	Default Value	-
In a Program	Yes	Default Format	-
Command Line	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_VPx 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 X,Y	Specify motion plane
VP 1000,2000	Specify vector position X,Y
CR 1000,0,360	Specify arc
VE	Vector end
VS 2000;VA 400000	Specify vector speed/vector acceleration
BGS;EN	Begin motion sequence; 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 n where

N is between 0 and 10 with a resolution of .0001.

n = ?	Returns the value of the vector speed ratio for the coordinate plane.

USAGE:

While Moving	Yes	Default Value	1
In a Program	Yes	Default Format	-
Command Line	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_VRx contains the vector speed ratio of the coordinate system.

RELATED COMMANDS:

VS	Vector Speed

EXAMPLES:

```
#A                    Vector Program
VMXY                  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; XQ#SPEED,1    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 n where

n 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.

n = ? Returns the value of the vector speed for the coordinate plane.

USAGE:

While Moving	Yes	Default Value	8192
In a Program	Yes	Default Format	-
Command Line	Yes		
Controller Usage	**ALL**		

OPERAND USAGE:

_VSx contains the vector speed of the 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 the coordinate system
VS ?	Return vector speed of the 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

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 n where

n is an unsigned number in the range between 0.004 and 1.0, with a resolution of 1/256.

n = ?	Returns the value of the vector time constant for the S coordinate plane.

USAGE:

While Moving	Yes	Default Value	1.0
In a Program	Yes	Default Format	1.4
Command Line	Yes		
Controller Usage	**ALL**		

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 the coordinate system
VT ?	Return vector time constant for the 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	Default Value	---
In a Program	Yes	Default Format	---
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	No		
Controller Usage	**ALL CONTROLLERS**		

RELATED COMMANDS:

CM	Contour Mode
CD	Contour Data
DT	Contour Time

EXAMPLES:

CM	Specify contour mode
DT 4	Specify time increment for contour
CD 200	Specify incremental position
WC	Wait for contour data to complete
CD 100	
WC	Wait for contour data to complete
DT 0	Stop contour
CD 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 IHF to indicate on which handle the command was executed. The command returns RS232 if communicating serially.

ARGUMENTS: None

USAGE:

While Moving	Yes	Default Value	--
In a Program	Yes	Default Format	--
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**DMC-1415/1416/1425**		

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 – 5, 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:

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	---
Command Line	Yes		
Can be Interrogated	No		
Used as an Operand	No		
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
BG              Begin the move
AM              After the move is over
WT 10000        Wait 10 seconds
SB 1            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 two programs may be executed simultaneously to perform multitasking.

ARGUMENTS: XQ #A,n XQm,n where

A is a program name of up to seven characters

m is a line number

n is the thread number (0 or 1) for multitasking

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

OPERAND USAGE:

_XQ 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!*

ZR

FUNCTION: Zero

DESCRIPTION:

The ZR command sets the compensating zero by changing the KD value in the control loop or returns the previously set value. It fits in the control equation as follows:

D(z) = GN(z-ZR/z)

ARGUMENTS: ZR n where

n is an unsigned number in the range 0 to 1 decimal with a resolution of 1/256

USAGE:

While Moving	Yes	Default Value	.9143
In a Program	Yes	Default Format	3.0
Command Line	Yes		
Can be Interrogated	Yes		
Used as an Operand	Yes		
Controller Usage	**DMC-1410/1411/1412/1414/1417**		

OPERAND USAGE:

_ZR contains the value of the compensating zero.

RELATED COMMANDS:

KD	Derivative
KP	Proportional
KI	Integral Gain

EXAMPLES:

ZR .95	Set zero to 0.95
ZR ?	Zero
0.9527	

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.

ARGUMENTS: ZS n where

0 returns stack to original condition

1 eliminates one return on stack

USAGE:

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

OPERAND USAGE:

_ZSn contains the stack level for the specified thread where n = 0 or 1. The response, an integer between zero and seven, indicates zero for beginning condition and seven for the deepest value.

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=
EN
```

THIS PAGE LEFT BLANK INTENTIONALLY

Index