B140

COMMAND REFERENCE

Manual Rev. 1.0

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 10/2008

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
CW unsolicited
DR data record
CF config
EO echo
IN user input
LZ leading
MG message
PF position format
QR data record
QZ record info
VF variable
ZA DR
DH DHCP

CONTOUR
CD data
CM axes
_CM buffer full
DT delta time

ETHERNET
IA IP address

IH Ethernet
IK Port block
MB modbus
MW modbus wait
SA send command
SM subnet mask
TH handle status
WH which handle

CONTROL
MO motor off
_MO motor off?
SH servo here
SH servo here
TM sample time

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

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

ERRORS
AB abort
_AX abort input
BL reverse soft
ED program line
ED1 thread
-
FL forward soft
LD limit
LF forward limit
_LR reverse limit

OE off on error
SC stop code
SD switch decel
TC tell code
#CMDERR; EN1
#LIMSWI; RE1
#POSERR; RE1

FEEDBACK

AL arm latch
_AL latch
CE configure

RL read latch
_RL latch
TD tell dual
TP tell position
TV tell

GEAR
GA axes
GD distance
GM gantry mode
_GP phase
GR ratio

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

INFO
_BN serial number
_BV axes
^R^V firmware

I/O
-

@IN[x] digital
@OUT[x] digital
AI wait for
AO analog out
CB clear digital
CN configure
CI communication
II input interrupt
OB output bit
OP output port
SB set digital out
TI tell input
TS tell switches
#ININT; RI1

#COMINT;EN

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

MOTION
AC acceleration
BG begin
_BG in motion?
DC deceleration
IP increment
IT s curve

JG jog
PA position
_PA last target
PR position
PR relative target
PT position
RP desired
SP speed

ST stop
~a axis variable

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

TIME
AT wait
TIME clock
WT wait

MOTION WAIT
AD distance
AM complete
AP position (TP)
AR distance
AS at speed (SP)
MC complete
MF forward (TP)
MR reverse (TP)

STEPPER
KS smoothing
LC low
MT motor
QS query error
YA drive
YB motor
YC encoder
YR correction
YS maintenance

VECTOR
AV wait for arc
_AVS arc length
-
CR circle
CS clear sequence
_CS segment
ES elliptical scale
LE linear end
_LE total arc length
LI linear point
LM linear axes
_LM buffer space
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
VS speed

VV Vector Variable

Table of Contents

This Page is Left Blank Intentionally

Overview

Controller Notation

This command reference describes commands for Galil Motion Controller: B140. Commands are listed in alphabetical order.

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

Trippoints

The controller provides several commands that can be used to make logical decisions, or "trippoints," based on events during a running program. Such events include: the completion of a specific motion, waiting for a certain position to be reached, or simply waiting for a certain amount of time to elapse.

When a program is executing on the controller, each program line is executed sequentially. However, when a trippoint command is executed, the program halts execution of the next line of code until the status of the trippoint is cleared. Note that the trippoint only halts execution of the thread from which it is commanded while all other independent threads are unaffected. Additionally, if the trippoint is commanded from a subroutine, execution of the subroutine, as well as the main thread, is halted.

Since trippoint commands are used as program flow instructions during a running program, they should not be implemented directly from the command line of the terminal. Sending a trippoint command directly from the command line might cause an interruption in communications between the host PC and the controller until the trippoint is cleared.

As a brief introduction, the following table lists the available commands and their basic usages:

AD	after distance
AI	after input
AM	after move
AP	after absolute position
AR	after relative position
AS	at speed
AT	at time relative to a reference time
AV	after vector distance
MC	motion complete and "in position"

MF	after motion forward
MR	after motion reverse
WT	wait for time

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.

Axes Arguments

Some commands require the user to identify the specific axes to be affected. These commands are followed by uppercase X,Y,Z, W or A,B,C,D,E,F,G and H. No commas are needed and the order of axes is not important. Do not insert any spaces prior to any command. For example, STX; AMX is invalid because there is a space after the semicolon. The proper syntax for commands requires that the command argument be separated from the command by a single space. When an argument is not required and is not given, the command is executed for all axes.

Valid syntax

SH A	Servo Here, A only
SH ABD	Servo Here, A,B and D axes
SH ACD	Servo Here, A,C and D axes
SH ABCD	Servo Here, A,B, C and D axes
SH BCAD	Servo Here, A,B,C and D axes
SH ADEG	Servo Here, A,D,E and G axes
SH H	Servo Here, H axis only
SH	Servo Here, all axes

Parameter Arguments

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

Valid syntax

AC n	Specify argument for A axis only
AC n,n	Specify argument for A and B only
AC n,,n	Specify argument for A and C only
AC n,n,n,n	Specify arguments for A,B,C,D axes

Where n is replaced by actual values.

Direct Command Arguments

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

PRB=1000	Sets B axis data at 1000
PR*=1000	Sets all axes to 1000

Interrogation

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

PR ?	The controller will return the PR value for the A axis
PR ,,,?	The controller will return the PR value for the D axis
PR ?,?,?,?	The controller will return the PR value for the A,B,C and D axes
PR*=?	The controller will return the PR value for all axes

Operand Usage

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

MG ‘operand’

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

V=_TPA

Usage Description

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

"While Moving":

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

"In a program":

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

"Command Line":

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

“Controller Usage”:

Identifies the controller models that can accept the command.

Default Description

In the command description, the DEFAULT section provides the default values for controller setup parameters. These parameters can be changed and the new values can be saved in the controller's non-

volatile memory by using the command, BN. If the setup parameters are not saved in non-volatile memory, the default values will automatically reset when the system is reset. A reset occurs when the power is turned off and on, when the reset button is pushed, or the command, RS, is given.

Resetting the Controller to Factory Default

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

For example, the command SP is used to set the Speed for each axis. If this parameter is not set by using the command, SP, the controller will automatically set this value for each axis. If the Speed is changed but not saved in non-volatile memory, the default value of 25,000 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.

# #

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,. A label can only be defined at the beginning of a new line.

ARGUMENTS: #nnnnnnn where

nnnnnnn is a label name up to seven characters

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

LL	List labels
DL	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:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

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

EXAMPLES:

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

& |

FUNCTION: Bitwise Logical Operators AND and OR

DESCRIPTION:

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

ARGUMENTS: n & m or n | m where

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

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

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

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

EXAMPLES:

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


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

( )

FUNCTION: Parentheses (order of operations)

DESCRIPTION:

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

ARGUMENTS: (n) where

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

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

+ - * / %	Math Operators
&l	Logical Operators

EXAMPLES:

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

;

FUNCTION: Semicolon (Command Delimiter)

DESCRIPTION:

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

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

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

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

ARGUMENTS: n; n; n; … where

n is a Galil command

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

NO (‘ apostrophe also accepted) comment

EXAMPLES:

```
BGX; 'comment

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

#High                     ;'#High priority thread executes twice as fast as
  a = a + 1; b = b + 1
JP#High

#Low                      ;'#Low when run in parallel
  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. (They are also used to designate the argument to a function, such as @ABS[n].)

ARGUMENTS: mmmmmmmm[n] where

mmmmmmmm is the array name

n is the array index and is an integer between 0 and 799

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

DM	Dimension Array
QU	Print/Upload Array

EXAMPLES:

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

+ - * / %

FUNCTION: Math Operators

DESCRIPTION:

The addition, subtraction, multiplication, division, and modulus 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) or (n % m) where

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

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

()	Parenthesis

EXAMPLES:

```
:x =((1+(2*3))/7)-2      ;'assign -1 to x
:MG 40 % 6               ;'integer remainder of 40 divided by 6
  4.0000
:
```

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

FUNCTION: Comparison Operators

DESCRIPTION:

The comparison operators are as follows:

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

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

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

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

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

()	Parentheses
IF	If statement
JP	Jump
JS	Jump subroutine

EXAMPLES:

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

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

DEFAULTS:

Default Value	-
Default Format	-

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
:
```

~

FUNCTION: Variable Axis Designator

DESCRIPTION:

The ~ signifies a variable axis designator

ARGUMENTS: ~n=m

n is a **lowercase** letter a through h

m is a positive integer 0 through 10, where
0 or “A” (quotes required) = X axis
1 or “B” = Y axis
2 or “C” = Z axis
3 or “D” = W axis

8 or “S” = S coordinate system

10 or “N” = Virtual N axis

USAGE:	**DEFAULTS:**		
While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	1.0
Command Line	Yes		
Controller Usage	**ALL CONTROLLERS**		

OPERAND USAGE:

~n contains the axis number 0-11

EXAMPLES:

```
~a=2;~b=3    ;'Sets ~s to 2 (Z axis).  Sets ~b to 6 (W axis)
PR~a=1000    ;'Relative position move 1000 counts on ~a axis (set as Z axis)
JG~b=9000    ;'Set jog speed of ~b axis (set as W axis) to 9000 cts/sec
BG~a~b       ;'Begin Motion on ~a and ~b axis
```

` (Ascii 96)

FUNCTION: Line Continuation Character

DESCRIPTION:
Allows a command in an application program to extend beyond the confines of the maximum line length of 40 characters. This is especially useful for code compression, long MG statements, or multiple conditions in an IF,JP or JS statement.

ARGUMENTS: none

USAGE:
Default Value -
In a Program Yes Default Format -
Command Line No

OPERAND USAGE:

RELATED COMMANDS:
MG Print Message

EXAMPLES:

```
IF((var100=1000)&(var101=50));MG"GO";EL`
SE;MG"STOP";ENDIF;
```

AB

FUNCTION: Abort

DESCRIPTION:

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

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

ARGUMENTS: AB n where

n = 0 The controller aborts motion and program

n = 1 The controller aborts motion only

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

USAGE:

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

DEFAULTS:

Default Value	---
Default Format	---

RELATED COMMANDS:

SH	Re-enables motor
OE	Specifies Off On Error

EXAMPLES:

```
AB                  ;'Stops motion
OE 1,1,1,1          ;'Enable off on error
AB                  ;'Shuts off motor command and stops motion

#A                  ;'Label - Start of program
 JG 20000           ;'Specify jog speed on 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:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

@SQR[n]	Square Root

EXAMPLES:

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

AC

FUNCTION: Acceleration

DESCRIPTION:

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

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

n is an unsigned number in the range 1024 to 1073740800. The parameters input will be rounded down to the nearest factor of 1024. The units of the parameters are counts per second squared.

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

USAGE:

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

DEFAULTS:

Default Value	256000
Default Format	10.0

OPERAND USAGE:

_ACx contains the value of acceleration for the specified axis.

RELATED COMMANDS:

DC	Specifies deceleration rate.
	Feedforward Acceleration
IT	Smoothing constant - S-curve

EXAMPLES:

`AC 150000,200000,300000,400000`	`Set A-axis acceleration to 150000, B-axis to 200000 counts/sec2, the C axis to 300000 counts/sec2, and the D-axis to 400000 count/sec2.`
`AC ?,?,?,?`	`Request the Acceleration`
`149504, 199680, 299008, 399360`	`Return Acceleration (resolution, 1024)`
`V=_ACB`	`Assigns the B acceleration to the variable V`

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

@ACOS[n]

FUNCTION: Inverse cosine

DESCRIPTION:

Returns in degrees the arc cosine of the given number.

ARGUMENTS: @ACOS[n] where

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

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

@ASIN[n]	Arc sine
@SIN[n]	sine
@ATAN[n]	Arc tangent
@COS[n]	Cosine
@TAN[n]	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. If in jog (JG) mode, the commanded motion is in the direction which moves away from the specified position.

The units of the command are quadrature counts. Only one axis may be specified at a time. AD can only be used when there's command motion on the axis.

If the direction of motion is reversed when in PT mode, the starting position for AD is reinitialized to the position at which the motor is reversed.

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

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

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

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

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

RELATED COMMANDS:

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

EXAMPLES:

```
#A;DP0,0                ;'Begin Program
PR 10000,20000          ;'Specify positions
BG                      ;'Begin motion
AD 5000                 ;'After A reaches 5000
MG "Halfway to A";TPA   ;'Send message
AD ,10000               ;'After B reaches 10000
MG "Halfway to B";TPB   ;'Send message
EN                      ;'End Program
```

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

AI

FUNCTION: After Input

DESCRIPTION:

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

ARGUMENTS: AI +/-n where

n is an integer between 1 and8 and represents the input number. If n is positive, the controller will wait for the input to go high. If n is negative, it waits for n to go low.

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

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

EXAMPLES:

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

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

AL

FUNCTION: Arm Latch

DESCRIPTION:

The AL command enables the latching function (high speed main or auxiliary position capture) of the controller. When the position latch is armed, the main or auxiliary encoder position will be captured upon a low going signal. Each axis has a position latch and can be activated through the general inputs:

A axis latch	Input 1
B axis latch	Input 2
C axis latch	Input 3
D axis latch	Input 4

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

ARGUMENTS: AL nnnn or AL n,n,n,n, where

n can be A,B,C,D, specifying the main encoder for the axis to be latched

n can be SA,SB,SC,SD, specifying the auxiliary encoder.

n can be TA,TB,TC,TD, specifying the main encoder is latched from the index pulse instead of a digital input.

USAGE:

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

DEFAULTS:

Default Value	0
Default Format	1.0

OPERAND USAGE:

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

RELATED COMMANDS:

RL	Report Latch

EXAMPLES:

```
#A                  ;'Program Label
ALB                 ;'Arm B-axis latch
JG,50000            ;'Set up jog at 50000 counts/sec
BGB                 ;'Begin the move
#LOOP               ;'Loop until latch has occurred
JP #LOOP,_ALB=1
RLB                 ;'Transmit the latched position
EN                  ;'End of program
```

AM

FUNCTION: After Move

DESCRIPTION:

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

ARGUMENTS: AM nnnnnn where

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

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

USAGE:

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

DEFAULTS:

Default Value	0
Default Format	1.0

RELATED COMMANDS:

BG	_BGn contains a 0 if motion complete
MC	Motion Complete

EXAMPLES:

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

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

@AN[n]

FUNCTION: Read analog input

DESCRIPTION:

Returns the value of the given analog input of Modbus Devices in volts

ARGUMENTS: @AN[n] where

use 1/0 number calculation from AO command

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

	Analog Range
@IN[n]	Read digital input
@OUT[n]	Read digital output
SB	Set digital output bit
CB	Clear digital output bit

EXAMPLES:

```
:MG @AN[1001] ;'print analog input 1
 1.7883
:x = @AN[1001] ;'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 = (HandleNum*1000) + ((Module-1)*4) + (Bitnum-1)

HandleNum is the handle specifier from A toD.

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

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

n = the voltage which ranges from 9.99 to –9.99

USAGE:

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

DEFAULTS:

Default Value	---
Default Format	---

RELATED COMMANDS:

SB	Set Bit
CB	Clear Bit
MB	Modbus

AP

FUNCTION: After Absolute Position

DESCRIPTION:

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

1. The actual motor position crosses the specified absolute position. When using a stepper motor, this condition is satisfied when the stepper position (as determined by the output buffer) has crossed the specified position. The motion profiling on the axis is complete.
2. The commanded motion is in the direction which moves away from the specified position.

The units of the command are quadrature counts. Only one axis may be specified at a time. AP can only be used when there's commanded motion on the axis.

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

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

USAGE:

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

DEFAULTS:

Default Value	---
Default Format	---

RELATED COMMANDS:

AR	Trippoint for relative distances
MF	Trippoint for forward motion

EXAMPLES:

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

Hint: *The accuracy of the AP command is the number of counts that occur in 2*TM µsec. Multiply the speed by 2*TM µsec 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. When using a stepper motor, this condition is satisfied when the stepper position (as determined by the output buffer) has crossed the specified Relative Position. For further information see Chapter 6 of the - *"Stepper Motor Operation"*.
2. The motion profiling on the axis is complete.
3. If in jog (JG) mode, the commanded motion is in the direction which moves away from the specified position.

If the direction of the motion is reversed when in position tracking mode (see PT command), the starting point for the trippoint is reinitialized to the point at which the motion reversed.

The units of the command are quadrature counts. Only one axis may be specified at a time. AR can only be used when there's commanded motion on the axis.

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

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

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

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

AV	Trippoint for after vector position for coordinated moves
AP	Trippoint for after absolute position

EXAMPLES:

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

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

AS

FUNCTION: At Speed

DESCRIPTION:

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

ARGUMENTS: AS nnnnnn where

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

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

EXAMPLES:

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

WARNING:

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

@ASIN[n]

FUNCTION: Inverse sine

DESCRIPTION:

Returns in degrees the arc sine of the given number.

ARGUMENTS: @ASIN[n] where

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

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

@ACOS[n]	Arc cosine
@SIN[n]	sine
@ATAN[n]	Arc tangent
@COS[n]	Cosine
@TAN[n]	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, even integer in the range 0 to 2 Billion

n = 0 defines a reference time at current time

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

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

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

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	-
Command Line	No		
Controller Usage	**ALL CONTROLLERS**		

EXAMPLES:

The following commands are sent sequentially

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

@ATAN[n]

FUNCTION: Inverse tangent

DESCRIPTION:

Returns in degrees the arc tangent of the given number.

ARGUMENTS: @ATAN[n]

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

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

@ASIN[n]	Arc sine
@SIN[n]	sine
@ACOS[n]	Arc cosine
@COS[n]	Cosine
@TAN[n]	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
EN	End 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                 ;' End program
```

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

#AUTOERR

FUNCTION: Automatic subroutine for notification of EEPROM checksum errors

DESCRIPTION:

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

USAGE:

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

RELATED COMMANDS:

_RS	Checksum error code
EN	End program

EXAMPLES:

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

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

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

AV

FUNCTION: After Vector Distance

DESCRIPTION:

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

ARGUMENTS: AV s where

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

USAGE:

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

DEFAULTS:

Default Value	0
Default Format	-

OPERAND USAGE:

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

EXAMPLES:

```
#MOVE;DP 0,0          ;'Label
CAT                   ;'Specify the T coordinate system
LMAB                  ;'Linear move for A,B
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";TPAB    ;'Print Message
EN                    ;'End Program
```

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

AX

FUNCTION: Abort Input Activation

DESCRIPTION: Input 8 can be used as an undedicated digital input or as the controller abort input. AZ sets the behavior of input 8.

ARGUMENTS: AXn where

n=0 input 8 configured as undedicated input.

n=1 input 8 configured abort input.

n=? returns status of AX

USAGE:

While Moving:	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**B140 only**

DEFAULTS:

Default Value:	1

BG

FUNCTION: Begin

DESCRIPTION:

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

ARGUMENTS: BG nnnnnnnnnn where

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

USAGE:

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

DEFAULTS:

Default Value	0
Default Format	-

OPERAND USAGE:

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

RELATED COMMANDS:

AM	After motion complete
ST	Stop motion

EXAMPLES:

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

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

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 to450 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 to3. The thread.

USAGE:

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

DEFAULTS:

Default Value of m	0

OPERAND USAGE:

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

= -LineNumber: breakpoint armed

= LineNumber: breakpoint encountered

= -2147483648: breakpoint not armed

RELATED COMMANDS:

SL	Single Step
TR	Trace

EXAMPLES:

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

BL

FUNCTION: Reverse Software Limit

DESCRIPTION:

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

When the reverse software limit is activated, the automatic subroutine #LIMSWI will be executed if it is included in the program.

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

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

n = -2147483648 Turns off the reverse limit.

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

USAGE:

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

DEFAULTS:

Default Value	-214783648
Default Format	Position format

OPERAND USAGE:

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

RELATED COMMANDS:

FL	Forward Limit
PF	Position Formatting

EXAMPLES:

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

Hint: *Galil Controllers also provide hardware limits. Both hardware or software limits will trigger the #LIMSWI automatic subroutine.*

BN

FUNCTION: Burn

DESCRIPTION:

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

PARAMETERS SAVED DURING BURN:

AC	CN	GR	MT	SM
				SP
	CW	IA		
	DC	IK	OE	
	DH	IL		TM
	EI	IT	OP	TR
	EO			VA
BL	ER			VD
CB		KS	PF	VF
	FL	LC		VS
		LD	PW	YA
	GA	LZ	SB	YB
CE	GM	MO		YC

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

OPERAND USAGE:

_BN contains the serial number of the controller.

RELATED COMMANDS:

BP	Burn Program
BV	Burn Variables

EXAMPLES:

```
AC 200000          Set acceleration
DC 150000          Set deceleration rate
SP 10000           Set speed
MT –1              Set motor type for A axis to be type ‘-1’, reversed
                   polarity servo motor
MO                 Turn motor off
BN                 Burn parameters; may take up to 5 seconds
```

BP

FUNCTION: Burn Program

DESCRIPTION::

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

ARGUMENTS: None

USAGE:

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

DEFAULTS:

Default Value	---

RELATED COMMANDS:

BN	Burn Parameters
BV	Burn Variable

BV

FUNCTION: Burn Variables & Arrays

DESCRIPTION::

The BV command saves the controller 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 is complete.

ARGUMENTS: None

USAGE:

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

DEFAULTS:

Default Value	---

OPERAND USAGE:

_BV returns the number of controller axes.

RELATED COMMANDS:

BN	Burn Parameters
BP	Burn Program

***Note 1**: This command will store the ECAM table values in non-volatile EEPROM memory.*

***Note 2**: 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.*

CB

FUNCTION: Clear Bit

DESCRIPTION:

The CB command sets the specified output bit low. CB can be used to clear the outputs of extended I/O which have been configured as outputs.

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.

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

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

EXAMPLES:

```
CB 3            Clear output bit 3
CB 1            Clear output bit 1
```

CD

FUNCTION: Contour Data

DESCRIPTION:

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

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

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

m (optional) is an integer in the range 0 to 8.

n = m = 0 terminates the Contour Mode.

m = 1 through 8 specifies the time interval (DT) of 2^m samples.

n = 0 and m = -1 pauses the contour buffer.

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

Note1: The command CD 0,0 . . .=0 would follow the last CD command in a sequence. CD 0,0 . . .=0 is similar to VE and LE. Once executed by the controller, CD 0,0 . . .=0 will terminate the contour mode.

Note2: The command CD0=0 will assign a variable CD0 the value of 0. In this case the user must have a space after CD in order to terminate the Contour Mode correctly. Example: CD 0=0 will terminate the contour mode for the X axis.

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

CM	Contour Mode
DT	Time Increment

EXAMPLES:

```
#Cont0                  ;'Define label #Cont0
CM ABCD                 ;'Specify Contour Mode
DT 4                    ;'Specify time increment for contour
CD 200,350,-150,500     ;'Specify incremental positions on A,B,C and C axes
                         'A-axis moves 200 counts  B-axis moves 350 counts C-
                         'axis moves -150 counts C-axis moves 500 counts
CD 100,200,300,400      ;'New position data
CD 0,0,0,0=0            ;'End of Contour Buffer/Sequence
#Wait;JP#Wait,_CM<>531  ;'Wait until path is done
EN                      ;'End program

#Cont1                  ;'Define label #Cont1
CM ABC                  ;'Specify Contour Mode
DT 8                    ;'Specify time increment for contour
CD 100,100,100          ;'New position data
```

```
CD 100,100,100          ;'New position data
CD 0,0,0 =-1            ;'Pause countour buffer set DT to resume
CD 100,100,100          ;'New position data
CD 100,100,100          ;'New position data
CD 0,0,0,0=0            ;'End of Contour Buffer/Sequence
#Wait;JP#Wait,_CM<>511  ;'Wait until path is done
EN
```

CE

FUNCTION: Configure Encoder

DESCRIPTION:

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

The MT command is configured for a stepper motor, the auxiliary encoder (used to count stepper pulses) will be forced to pulse and direction.

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

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

M	Main encoder type	N	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 = 10 implies M = 2 and N = 8, thus both encoders are reversed quadrature.

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

USAGE:

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

DEFAULTS:

Default Value	0
Default Format	2.0

OPERAND USAGE:

_CEn contains the value of encoder type for the axis specified by 'n'.

RELATED COMMANDS:

MT	Specify motor type

EXAMPLES:

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

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

CF

FUNCTION: Configure

DESCRIPTION:

Sets the default port for unsolicited messages. By default, the B140 will send unsolicited responses to the main RS-232 serial port. The CF command allows the user to send unsolicited responses to the Main or Aux Serial Port, or Handles A-D.

ARGUMENTS: CF n where

n is A thru D for Ethernet handles 1 thru 4, S for Main serial port, D port or I is to set to the port that issues the CF command.

USAGE:

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

OPERAND USAGE:

_CF contains the decimal value of the ASCII letter.

RELATED COMMANDS:

CW	Configures MSB of unsolicited messages
WH	What Handle
TH	Tell Handles

CI

FUNCTION: Configure Communication Interrupt

DESCRIPTION:

The CI command configures a program interrupt based on characters received on communications port 1. An interrupt causes program flow to jump to the #COMINT subroutine. If multiple program threads are used, the #COMINT subroutine runs in thread 0 and the remaining threads continue to run without interruption. The characters received can be accessed via the internal variables P1CH, P1ST, P1NM, P1CD.

ARGUMENTS: CI n, m

m=1 to enable the mode

PARAMETER	EXPLANATION
n = 0	Do not interrupt
n = 1	Interrupt on carriage return
n = 2	Interrupt on any character
n = -1	Clear interrupt data buffer

USAGE:

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

DEFAULTS:

Default Value	n = 0, m = 0
Default Format	-

RELATED COMMANDS:

	Configure communications
IN	Communication input
MG	Message output

EXAMPLES:

```
CI 1,1      Interrupt when the <enter> key is received on port 1
CI 2,1      Interrupt on a single character received on Port 1
```

CM

FUNCTION: Contour Mode

DESCRIPTION:

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

The command, CM?, can be used to check the number of available contour segments. A value of 0 returned from the command CM? indicates that the Contour Buffer is full. A value of 31 indicates that the Contour Buffer is empty.

ARGUMENTS: CM nnnnnnnnnn where

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

n = ? Returns a 0 if the contour buffer is full and 31 if the contour buffer is empty.

USAGE:

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

DEFAULTS:

Default Value	0
Default Format	3.0

OPERAND USAGE:

_CM contains a '0' if the contour buffer is full; otherwise it contains the number of available contour segments.

RELATED COMMANDS:

CD	Contour Data
DT	Time Increment

EXAMPLES:

```
V=_CM;V=          Return contour buffer status
CM?               Return contour buffer status
CM AC             Specify A,C axes for Contour Mode
```

#CMDERR

FUNCTION: Command error automatic subroutine

DESCRIPTION:

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

USAGE:

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

RELATED COMMANDS:

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

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 program
```

***NOTE**: An application program must be executing for #CMDERR to execute, which runs in thread 0.*

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

CN

FUNCTION: Configure

DESCRIPTION:

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

ARGUMENTS: CN m,n,o,p,q where

m,n,o are integers with values 1 or -1. p is an integer, 0 or 1.

m =	1	Limit switches active high
	-1	Limit switches active low
n =	1	Home switch configured to drive motor in forward direction when input is high. See HM and FE commands.
	-1	Home switch configured to drive motor in reverse direction when input is high. See HM and FE commands
o =	1	Latch input is active high
	-1	Latch input is active low
p =	1	Configures inputs 5,6,7,8, as selective abort inputs for axes A,B,C,D,E respectively. Will also trigger #POSERR automatic subroutine if program is running.
	0	Inputs 5,6,7,8, are configured as general use inputs
q=	1	Abort input will not terminate program execution
	0	Abort input will terminate program execution

USAGE:

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

DEFAULTS:

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

OPERAND USAGE:

_CN0 Contains the limit switch configuration
_CN1 Contains the home switch configuration
_CN2 Contains the latch input configuration
_CN3 Contains the state of the selective abort function (1 enabled, 0 disabled)
_CN4 Contains whether the abort input will terminate the program

RELATED COMMANDS:

AL Arm latch

EXAMPLES:

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

@COM[n]

FUNCTION: Bitwise complement

DESCRIPTION:

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

ARGUMENTS: @COM[n] where

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

The integer is interpreted as a 32-bit field.

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

& |	Logical operators AND and OR

EXAMPLES:

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

#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 serial port.

USAGE:

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

RELATED COMMANDS:

P1CD	Serial port 1 code
P1CH	Serial port 1 character
P1NM	Serial port 1 number
P1ST	Serial port 1 string
CI	Configure #COMINT
EN	End subroutine

EXAMPLES:

```
#A              ;'Program Label

CI2,1             ;'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 of -32768 to 32767, with a fractional resolution of 16-bit..

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

@ASIN[n]	Arc sine
@SIN[n]	sine
@ATAN[n]	Arc tangent
@ACOS[n]	Arc cosine
@TAN[n]	Tangent

EXAMPLES:

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

CR

FUNCTION: Circle

DESCRIPTION:

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

ARGUMENTS: CR r,θ,Δθ < n > o where

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

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

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

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

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

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

USAGE:

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

DEFAULTS:

RELATED COMMANDS:

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

EXAMPLES:

```
VMAB                Specify vector motion in the A and B plane
VS 10000            Specify vector speed
CR 1000,0,360       Generate circle with radius of 1000 counts, start at 0
                    degrees and complete one circle in counterclockwise
                    direction.
CR 1000,0,360<40000 Generate circle with radius of 1000 counts, start at 0
                    degrees and complete one circle in counterclockwise
VE                  End Sequence
BGS                 Start motion
```

CS

FUNCTION: Clear Sequence

DESCRIPTION:

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

ARGUMENTS: CSS where

S can be used to clear the sequence buffer for the "S" coordinate system.

USAGE:

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

DEFAULTS:

Default Value	---
Default Format	---

OPERAND USAGE:

_CSn contains the segment number in the sequence specified by n, S. 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

EXAMPLES:

```
#CLEAR              ;'Label

VP 1000,2000        ;'Vector position
VP 4000,8000        ;'Vector position
CSS                 ;'Clear vectors specified in S coordinate system

VP 1000,5000        ;'New vector
VP 8000,9000        ;'New vector
CSS                 ;'Clear vectors specified in S coordinate system
EN                  ;'End program
```

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 Galil PC 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. Because of the dual function, only one field can be set at a time. Instead of "CW2,1," use "CW2;CW,1".

ARGUMENTS: CW n,m where

n = 0 Causes the controller to return the copyright information

n = 1 Causes the controller to set the MSB of unsolicited returned characters to 1

n = 2 Causes the controller to not set the MSB of unsolicited characters.

n = ? Returns the copyright information for the controller.

m is optional

m = 0 Causes the controller to pause program execution when hardware handshake disables transmissions.

m = 1 Causes the controller to continue program execution when hardware handshake disables transmission. Characters output will be lost.

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	2, 0
In a Program	Yes	Default Format	-----
Command Line	Yes		
Controller Usage	**ALL CONTROLLERS**		

OPERAND USAGE:

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

Note: *The CW command can cause unrecognized 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*

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 comma when specified in one command. The argument * deallocates all the variables, and *[0] deallocates all the arrays.

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

c[0] = Defined array name

variable-name = 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
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	------
Default Format	------

OPERAND USAGE:

_DA contains the total number of arrays available. For example, before any arrays have been defined, the operand _DA is 6. If one array is defined, the operand _DA will return 5.

RELATED COMMANDS:

DM	Dimension Array

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

```
DM Cars[400],Sales[50]          Dimension 2 arrays
Total=70                        Assign 70 to the variable Total
DA Cars[0],Sales[0],Total       Deallocate the 2 arrays & variables
DA*[]                           Deallocate all arrays
DA *,*[]                        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 of the motors for independent moves such as PR, PA and JG moves. The parameters will be rounded down to the nearest factor of 1024 and have units of counts per second squared.

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

n is an unsigned numbers in the range 1024 to 1073740800

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

USAGE:

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

DEFAULTS:

Default Value	256000
Default Format	10.0

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

OPERAND USAGE:

_DCn contains the deceleration rate for the specified axis.

RELATED COMMANDS:

AC	Acceleration
PR	Position Relative
PA	Position Absolute
SP	Speed
JG	Jog
SD	Limit Switch Deceleration

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: Define Encoder Position

DESCRIPTION:

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

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

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

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

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

USAGE:

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

DEFAULTS:

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

OPERAND USAGE:

_DEn contains the current position of the specified encoder.

RELATED COMMANDS:

DP	Define position
TD	Tell Dual Encoder position

EXAMPLES:

`DE 0,100,200,400`	`Set the current encoder position to 0,100,200,400 on A,B,C and D axes`
`DE?,?,?,?`	`Return auxiliary encoder positions`
`DualA=_DEA`	`Assign auxiliary encoder position of A-axis to the variable DualA`

DH

FUNCTION: DHCP Enable

DESCRIPTION:

The DH command configures the DHCP or BOOT-P functionality on the controller for Server IP addressing.

ARGUMENTS: DH n where

n = 0 disables DHCP and enables BOOT-P

n = 1 disables BOOT-P and enables DHCP

n = ? returns the current state of the setting

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes

DEFAULTS:

Default Value	1.0
Default Format	-

RELATED COMMANDS:

IA	IP Address

EXAMPLES:

`DH 1`	`Sets the DHCP function on. IA assignment will no longer work. IP address cannot be burned. Controller will receive its IP address from the DHCP server on the network.`
`DH 0`	`Sets the DHCP function off, and the Boot-P function on.`

DL

FUNCTION: Download

DESCRIPTION:

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

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

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

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

ARGUMENTS: DL n where

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

USAGE:

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

DEFAULTS:

Default Value	---
Default Format	---

OPERAND USAGE:

When used as an operand, _DL gives the number of available labels (62 maximum)

RELATED COMMANDS:

UL	Upload

EXAMPLES:

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

DM

FUNCTION: Dimension

DESCRIPTION:

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

ARGUMENTS: DM c[n] where

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

n = ? Returns the number of array elements available.

USAGE:

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

DEFAULTS:

Default Value	---
Default Format	---

OPERAND USAGE:

_DM contains the available array space. For example, before any arrays have been defined, the operand _DM will return 800. If an array of 100 elements is defined, the operand _DM will return 700.

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[600]               Define dimension of array Tests with 600 elements
```

DP

FUNCTION: Define Position

DESCRIPTION:

.

The DP command sets the commanded reference position. The units are in steps. Example: DP 0 this will set the registers for TD and RP to zero, but will not effect the TP register value.

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

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

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

USAGE:

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

DEFAULTS:

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

OPERAND USAGE:

_DPn contains the current position of the specified axis.

RELATED COMMANDS:

PF	Position Formatting

EXAMPLES:

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

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

DR

FUNCTION: Configures Axes and I/O Data Record Update Rate

DESCRIPTION:

The controller creates a QR record and sends it periodically to a UDP Ethernet Handle

ARGUMENTS: DR n, m

n specifies the data update rate in samples between updates. When TM is set to the default of 1000, n specifies the data update rate in milliseconds. n=0 to turn it off, or n must be an integer in the range of 2 to 30,000.

m specifies the Ethernet handle on which to periodically send the Data Record. 0 is handle A, 1 is B… 3 is D. The handle must be UDP (not TCP).

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes

DEFAULTS:

Default Value	DR0 (off)
Default Format	--

OPERAND USAGE:

_DR contains the data record update rate.

RELATED COMMANDS:

QZ	Sets format of data
QR	Query a single data record

EXAMPLES:

```
:DR1000,0
:G x      ~   P
_ `      @~   P
_ H      `~   P
_ 0      ~   P
DR0
```

Note: *The data record is in a binary, non-printable format (the output above is normal)*

DT

FUNCTION: Delta Time

DESCRIPTION:

The DT command sets the time interval for Contour Mode. Sending the DT command once will set the time interval for all contour data until a new DT command (or CDm=n) is sent.

ARGUMENTS: DT n where

n is an integer in the range 0 to 8.

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

n = -1 allows a pre-load of the contour buffer or to asynchrounsly pause the contour buffer. DT-1 during countor mode will pause the contour buffer (and commanded movement). A positive DT will resume contour mode from paused position of buffer.

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

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

USAGE:

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

DEFAULTS:

Default Value	1
Default Format	1.0

OPERAND USAGE:

_DT contains the value for the time interval for Contour Mode

RELATED COMMANDS:

CM	Contour Mode
CD	Contour Data

EXAMPLES:

```
DT 4                    Specifies time interval to be 16 msec
DT 7                    Specifies time interval to be 128 msec

#Cont0                  ;'Define label #Cont0
CM ABCD                 ;'Specify Contour Mode
DT 4                    ;'Specify time increment for contour
CD 200,350,-150,500     ;'Specify incremental positions on A,B,C and C axes
                         'A-axis moves 200 counts  B-axis moves 350 counts C-
                         'axis moves -150 counts C-axis moves 500 counts
CD 100,200,300,400      ;'New position data
CD 0,0,0,0=0            ;'End of Contour Buffer/Sequence
#Wait;JP#Wait,_CM<>31   ;'Wait until path is done
EN                      ;'End program

#Cont1                  ;'Define label #Cont1
CM AB                   ;'Specify Contour Mode
DT -1                   ;'Pause Contour Mode to allow pre-load of buffer
CD 100,200              ;'Countour Data pre-loaded in buffer
```

```
CD 400,200              ;'Countour Data pre-loaded in buffer
CD 200,100              ;'Countour Data pre-loaded in buffer
CD 300,50               ;'Countour Data pre-loaded in buffer
AI -1                   ;'Wait for Analog input 1 to go low
DT 8                    ;'Set positive DT to start contour mode
CD 0,0,0,0=0            ;'End of Contour Buffer/Sequence
#Wait;JP#Wait,_CM<>31   ;'Wait until path is done
EN                      ;'End program
```

EA

FUNCTION: Choose ECAM master

DESCRIPTION:

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

ARGUMENTS: EA n where

n is one of the axis specified as A,B,C,D, or N

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

RELATED COMMANDS:

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

EXAMPLES:

```
EAB                Select B as a master for ECAM
```

EB

FUNCTION: Enable ECAM

DESCRIPTION:

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

ARGUMENTS: EB n where

n = 1 Starts ECAM mode

n = 0 Stops ECAM mode.

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

USAGE:

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

DEFAULTS:

Default Value	0
Default Format	1.0

OPERAND USAGE:

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

RELATED COMMANDS:

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

EXAMPLES:

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

EC

FUNCTION: ECAM Counter

DESCRIPTION:

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

ARGUMENTS: EC n where

n is an integer between 0 and 256.

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

USAGE:

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

DEFAULTS:

Default Value	0
Default Format	1.0

OPERAND USAGE:

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

RELATED COMMANDS:

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

EXAMPLES:

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

ED

FUNCTION: Edit

DESCRIPTION:

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

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

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

OPERAND USAGE:

_ED	contains the line number of the last line to have an error.
_ED1	contains the number of the thread where the error occurred (for multitasking).

EXAMPLES:

```
ED
0 #START
1 PR 2000
2 BGA
3 SLKJ                          Bad line
4 EN
5 #CMDERR                       Routine which occurs upon a command error
6 V=_ED
7 MG "An error has occurred" {n}
8 MG "In line", V{F3.0}
9 ST
10 ZS0
11 EN
```

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

EG

FUNCTION: ECAM go (engage)

DESCRIPTION:

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

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

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

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

USAGE:

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

DEFAULTS:

Default Value	0
Default Format	1.0

OPERAND USAGE:

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

RELATED COMMANDS:

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

EXAMPLES:

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

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

ELSE

FUNCTION: Else function for use with IF conditional statement

DESCRIPTION:

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

ARGUMENTS: ELSE

USAGE:

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

DEFAULTS:

Default Value	
Default Format	

RELATED COMMANDS:

ENDIF	End of IF conditional Statement

EXAMPLES:

```
#A
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 is false
ENDIF                                  ;'End of 1st conditional statement
EN
```

EM

FUNCTION: Cam cycles (modulus)

DESCRIPTION:

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

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

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

USAGE:

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

DEFAULTS:

Default Value	
Default Format	

OPERAND USAGE:

_EMn contains the cycle of the specified axis.

RELATED COMMANDS:

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

EXAMPLES:

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

EN

FUNCTION: End

DESCRIPTION:

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

The EN command is used to end the automatic subroutines #COMINT and #CMDERR.

When the EN command is used to terminate the #COMINT communications interrupt subroutine, there are 2 argurments. The first determines whether trippoints will be restored upon completion of the subroutine, and the second determines whether the communication will be re-enabled.

ARGUMENTS: EN m, n where

m = 0: Return from subroutine without restoring trippoint

m = 1: Return from subroutine and restore trippoint

n = 0 : Return from #COMINT without restoring CI interrupt trigger

n = 1 : Return from #COMINT and restore CI interrupt trigger

Note 1: The default value for the argument is 0.

Note 2: Use the RE command to return from the interrupt handling subroutines #LIMSWI and #POSERR. Use the RI command to return from the #ININT subroutine.

USAGE:

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

DEFAULTS:

Default Value	m=0
Default Format	

RELATED COMMANDS:

RE	Return from error subroutine
RI	Return from interrupt subroutine

EXAMPLES:

```
#A              ;'Program A
PR 500          ;'Move A axis forward 500 counts
BGA             ;'Begin motion
AMA             ;'Pause the program until the A axis completes the 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 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
In a Program	Yes
Command Line	No
Controller Usage	**ALL CONTROLLERS**

RELATED COMMANDS:

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

EXAMPLES:

```
#A
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 is false
ENDIF                                  ;'End of 1st conditional statement
EN
```

EO

FUNCTION: Echo

DESCRIPTION:

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

ARGUMENTS: EO n where

n = 0 0 turns echo off

n = 1 1 turns echo on.

USAGE:

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

DEFAULTS:

Default Value	0
Default Format	1.0

EXAMPLES:

```
EO 0          Turns echo off
EO 1          Turns echo on
```

EP

FUNCTION: Cam table intervals and starting point

DESCRIPTION:

The EP command defines the ECAM table intervals and offset. The offset is the master position of the first ECAM table entry. The interval is the difference of the master position between 2 consecutive table entries. This command effectively defines the size of the ECAM table. The parameter m is the interval and n is the starting point. Up to 257 points may be specified.

ARGUMENTS: EP m,n where

m is a positive integer in the range between 1 and 32,767

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

n is an integer between -2,147,483,648 and 2,147,483,647. n is the offset.

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

OPERAND USAGE:

_EP contains the value of the interval m.

RELATED COMMANDS:

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

EXAMPLES:

```
EP 20,100          Sets the cam master points to 100,120,140 . . .
D =  _EP           Set the variable D equal to the ECAM internal valve
```

EQ

FUNCTION: ECAM quit (disengage)

DESCRIPTION:

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

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

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

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

USAGE:

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

OPERAND USAGE:

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

RELATED COMMANDS:

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

EXAMPLES:

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

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

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

DEFAULTS:

Default Value	1,1
Default Format	

RELATED COMMANDS:

VM	Vector Mode
CR	Circle move
VP	Vector position

EXAMPLES:

```
VMAB;ES3,4          Divide B resolution by 4/3
VMCA;ES2,3          Divide A resolution by 3/2
VMAC; ES3,2         Divide A Resolution by 3/2
```

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

ET

FUNCTION: Electronic cam table

DESCRIPTION:

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

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

m is an integer between 0 and 256

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

n=? Returns the slave position for the specified point.

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

USAGE:

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

RELATED COMMANDS:

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

EXAMPLES:

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

EW

FUNCTION: ECAM Widen Segment

DESCRIPTION:

The EW command allows widening the length of one or two ECAM segments beyond the width specified by EP. For ECAM tables with one or two long linear sections, this allows placing more points in the curved sections of the table.

There are only two widened segments, and if used they are common for all ECAM axes. Remember that the widened segment lengths must be taken into account when determining the modulus (EM) for the master. The segments chosen should not be the first or last segments, or consecutive segments.

ARGUMENTS: EW m1=n1,m2=n2 where

m1 is the index of the first widened segment. m1 is a positive integer between 1 and 255.

n1 is the length of the first widened segment in master counts. n1 is an integer between 1 and 2,147,483,647.

m2 is the index of the second widened segment. m2 is a positive integer between 3 and 255.

n2 is the length of the second widened segment in master counts. n2 is an integer between 1 and 2,147,483,647.

If m1 or m2 is set to -1, there is no widened segment. The segment number m2 must be greater than m1, and m2 may not be used unless m1 is used.

USAGE:

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

DEFAULTS:

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

OPERAND USAGE:

_EW0 contains m1, the index of the first widened segment.

_EW1 contains n1, the length of the first widened segment.

_EW2 contains m2, the index of the second widened segment

_EW3 contains n2, the length of the second widened segment.

RELATED COMMANDS:

EP	ECAM master positions
EA	Choose ECAM master
EB	Enable ECAM
EC	Set ECAM table index
EG	Engage ECAM Slave
EM	Specify ECAM cycle
EQ	Disengage ECAM Slave
ET	ECAM table

EXAMPLES:

```
EW 41=688           :'Widen segment 41 to 688 master counts
EW 41=688, 124=688  :'Widen segments 41 and 124 to 688 master counts
```

EY

FUNCTION: ECAM Cycle Count

DESCRIPTION:

Sets or gets the ECAM cycle count. This is the number of times that the ECAM axes have exceeded their modulus as defined by the EM command. EY will increment by one each time the master exceeds its modulus in the positive direction, and EY will decrement by one each time the master exceeds its modulus in the negative direction. EY can be used to calculate the absolute position of an axis with the following equation:

Absolute position = EY * EM + TP

ARGUMENTS: EY n where

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

n = ? returns the current cycle count.

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

OPERAND USAGE:

_EY returns the current cycle count

RELATED COMMANDS:

EM	ECAM modulus

EXAMPLES:

```
MG _EY * _EMY + _TPY    print absolute position of master (Y)
```

FE

FUNCTION: Find Edge

DESCRIPTION:

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

This command is useful for creating your own homing sequences.

ARGUMENTS: FE nnnnnnnn where

n is A,B,C,D, or any combination to specify the axis or axes

No argument specifies all axes.

USAGE:

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

DEFAULTS:

Default Value	
Default Format	

RELATED COMMANDS:

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

EXAMPLES:

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

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

FI

FUNCTION: Find Index

DESCRIPTION:

The FI and BG commands move the motor until an encoder index pulse is detected. The controller looks for a transition from low to high. The first stage jogs the motor at the speed and direction of the JG command until a transition is detected on the index line. When the transition is detected, the motor will decelerate to a stop.

ARGUMENTS: FI nnnnnn where

n is A,B,C,D, or any combination to specify the axis or sequence

No argument specifies all axes.

USAGE:

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

DEFAULTS:

Default Value	
Default Format	

RELATED COMMANDS:

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

EXAMPLES:

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

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

FL

FUNCTION: Forward Software Limit

DESCRIPTION:

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

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

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

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

n = 2147483647 turns off the forward limit

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

USAGE:

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

DEFAULTS:

Default Value	2147483647
Default Format	Position Format

OPERAND USAGE:

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

RELATED COMMANDS:

BL	Reverse Limit
PF	Position Formatting

EXAMPLES:

```
FL 150000           Set forward limit to 150000 counts on the A-axis

#TEST               ;'Test Program
AC 1000000          ;'Acceleration Rate
DC 1000000          ;'Deceleration Rate
FL 15000            ;'Forward Limit
JG  5000            ;'Jog Forward
BGA                 ;'Begin
AMA                 ;'After Limit
TPA                 ;'Tell Position
EN                  ;'End
```

Hint: *Galil controllers also provide hardware limits. Both hardware or software limits will trigger the #LIMSWI automatic subroutine if a program is running.*

@FRAC[n]

FUNCTION: Fractional part

DESCRIPTION:

Returns the fractional part of the given number

ARGUMENTS: @FRAC[n]

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

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

@INT[n]	Integer part

EXAMPLES:

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

GA

FUNCTION: Master Axis for Gearing

DESCRIPTION:

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

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

n can be A,B,C,D, or N. The value of n is used to set the specified main encoder axis as the gearing master and M and N represents the virtual axis. The slave axis is specified by the position of the argument. The first position of the argument corresponds to the 'A' axis, the second position corresponds to the 'B' axis, etc. A comma must be used in place of an argument if the corresponding axes will not be a slave.

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

n can be S. S is used to specify the vector motion of the coordinated system, S, as the gearing master.

n can be DA,DB,DC,DD. The value of n is used to set the specified auxiliary encoder axis as the gearing master.

USAGE:

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

DEFAULTS:

Default Value	
Default Format	

RELATED COMMANDS:

GR	Gear Ratio
GM	Gantry Mode

EXAMPLES:

```
#GEAR              ;'Gear program
GA ,A,PS           ;'Specify A axis as master for B and vector motion
                   ;'on PS as master for C
GR ,.5,-2.5        ;'Specify B and C ratios
JG 5000            ;'Specify master jog speed
BGA                ;'Begin motion
WT 10000           ;'Wait 10000 msec
STA                ;'Stop
AMA                ;'Wait for motion to complete
EN                 ;'End Program
```

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

GD

FUNCTION: Gear Distance

DESCRIPTION:

The GD command sets the distance of the master axis over which the specified slave will be engaged, disengaged or changed to a new gear setting. The distance is entered as an absolute value, the motion of the master may be in either direction. If the distance is set to 0, then the gearing will engage instantly.

ARGUMENTS: GD n,n,n,n,n,n,n,n where

n is an integer in the range 0 to 32767, the units are in encoder counts

n = 0 will result in the conventional method of instant gear change

n = ? will return the value that is set for the appropriate axis

USAGE:

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

DEFAULTS:

Default Value	0
Default Format	5.0

OPERAND USAGE:

_GDn contains the distance the master axis will travel for the specified slave axis to fully engage, disengage, or change ratios.

RELATED COMMANDS:

_GP	Gearing Phase Differential
GR	Gear Ratio
GA	Gear Axis

EXAMPLES:

```
#A
GA,X            ;'Sets the X axis as the gearing master for the Y axis
GD,5000         ;'Set distance over which gearing is engaged to 5000 counts
                ;'of the master axis.
JG5000          ;'Set the X axis jog speed to 5000 cts/sec
BGX             ;'Begin motion on the X axis
ASX             ;'Wait until X axis reaches the set speed of 5000 counts/sec
GR,1            ;'Engage gearing on the Y axis with a ration of 1:1, the
                ;'distance to fully engage gearing will be 5000 counts of the
                ;'master axis
WT1000          ;'Wait 1 second
GR,3            ;'Set the gear ratio to three.  The ratio will be changed
                ;'over the distance set by the GD command
WT1000          ;'Wait 1 second
GR,0            ;'Disengage the gearing between the Y axis slave and the
                ;'master.  The gearing will be disengaged over the number of
                ;'counts of the master specified with the GD command above
EN              ;'End program
```

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 be stopped by the ST command or by limit switches. Only GR0 will stop the gearing in this mode.

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

n = 0 Disables gantry mode function

n = 1 Enables the gantry mode

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

USAGE:

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

DEFAULTS:

Default Value	0
Default Format	1.0

OPERAND USAGE:

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

RELATED COMMANDS:

GR	Gear Ratio
GA	Gear Axes

EXAMPLES:

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

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

_GP

FUNCTION: Gearing Phase Differential Operand

DESCRIPTION:

The _GP operand contains the value of the "phase differential"[1] accumulated on the most current change in the gearing ratio between the master and the slave axes. The value does not update if the distance over which the slave will engage is set to 0 with the GD command.

The operand is specified as: _GPn where n is the specified slave axis

[1]Phase Differential is a term that is used to describe the lead or lag between the master axis and the slave axis due to gradual gear shift. Pd=GR*Cm-Cs where Pd is the phase differential, GR is the gear ratio, Cm is the number of encoder counts the master axis moved, and Cs is the number of encoder counts the slave moved.

RELATED COMMANDS:

GR	Gear Ratio
GA	Gear Axis

EXAMPLES:

```
#A
GAY                 ;'Sets the Y axis as the gearing master for the X axis.
                    ;'This axis does not have to be under servo control.  In
                    ;'this example, the axis is connected to a conveyor
                    ;'operating open loop.
GD1000              ;'Set the distance that the master will travel to 1000
                    ;'counts before the gearing is fully engaged for the X
                    ;'axis slave.
AI-1                ;'Wait for input 1 to go low.  In this example, this
                    ;'input is representing a sensor that senses an object
                    ;'on a conveyor.  This will trigger the controller to
                    ;'begin gearing and synchronize the master and slave
                    ;'axes together.
GR1                 ;'Engage gearing between the master and slave
P1=_TPY             ;'Sets the current Y axis position to variable P1.  This
                    ;'variable is used in the next command, because MF
                    ;'requires an absolute position..
MF,(P1+1000)        ;'Wait for the Y axis (master) to move forward 1000
                    ;'encoder counts so the gearing engagement period is
                    ;'complete.  Then the phase difference can be adjusted
                    ;'for.  Note this example assumes forward motion.
IP_GPX              ;'Increment the difference to bring the master/slave in
                    ;'position sync from the point that the GR1 command was
                    ;'issued.
EN                  ;'End Program
```

GR

FUNCTION: Gear Ratio

DESCRIPTION:

GR specifies the Gear Ratios for the geared axes in the electronic gearing mode. The master axis is defined by the GA command. The gear ratio may be different for each geared axis. The master can go in both directions. A gear ratio of 0 disables gearing for each axis. A limit switch also disables the gearing unless gantry mode has been enabled (see GM command).

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

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

n = 0 Disables gearing

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

USAGE:

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

DEFAULTS:

Default Value	0
Default Format	3.4

OPERAND USAGE:

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

RELATED COMMANDS:

GA	Master Axis
GM	Gantry Mode

EXAMPLES:

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

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

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

HM

FUNCTION: Home

DESCRIPTION:

The HM command performs a two-stage homing sequence for stepper motor operation.

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

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

USAGE:

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

DEFAULTS:

OPERAND USAGE:

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

RELATED COMMANDS:

CN	Configure Home
FI	Find Index Only
FE	Find Home Only
HV	Homing velocity

EXAMPLES:

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

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

HS

FUNCTION: Handle Assignment Switch

DESCRIPTION:

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

ARGUMENTS: HSh=i where

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

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

S is used to represent the current handle executing the command

USAGE:

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

DEFAULTS:

Default Value	--
Default Format	--

RELATED COMMANDS:

IH	Internet Handle

EXAMPLES:

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

HX

FUNCTION: Halt Execution

DESCRIPTION:

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

ARGUMENTS: HXn where

n is an integer in the range of 0 to 3 and indicates the thread number.

USAGE:

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

DEFAULTS:

Default Value	n = 0
Default Format	

OPERAND USAGE:

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

0 Thread not running

1 Thread is running

2 Thread has stopped at trippoint

RELATED COMMANDS:

XQ	Execute program
HX	Stop all threads of motion

EXAMPLES:

```
XQ #A              Execute program #A, thread zero
XQ #B,3            Execute program #B, thread three
HX0                Halt thread zero
HX3                Halt thread three
```

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. 1< = t < = 2,147,483,647 up to 5 retries occur. (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
In a Program	Yes
Command Line	Yes
Controller Usage	**ALL CONTROLLERS**

DEFAULTS:

Default Value	n = 0, t=250
Default Format	

OPERAND USAGE:

_IA0	contains the IP address representing a 32 bit signed number (Two's complement)
_IA1	contains the value for t (retry time)
_IA2	contains the number of available handles
_IA3	contains the number of the handle using this operand where the number is 0 to 5. 0 represents handle A, 1 handle B, etc.
_IA4	contains the number of the handle that lost communication last, contains A-1 on reset to indicate no handles lost
_IA5	returns autonegotiation Ethernet speed. Returns 10 for 10-Base T and returns 100 for 100-Base T, it will return -1 if there is no physical link

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 consist of one or more conditional statements and each condition must be enclosed with parenthesis (). 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

Note: Bit wise operators | and & can be used to evaluate multiple conditions.

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

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

EXAMPLES:

```
#A
IF (_TEA<1000)                          ;'IF conditional statement based on
                                        ;'A motor position
MG "Motor is within 1000 counts of zero" ;'Message to be executed if "IF"
                                        ;'conditional statement is true
ENDIF                                   ;'End of IF conditional statement
EN                                      ;'End Program
```

IH

FUNCTION: Open Internet Handle

DESCRIPTION:

The IH command is used when the controller is operated as a master (also known as a client). This command opens a handle and connects to a slave.

Each controller may have 4 handles open at any given time. They are designated by the letters A through H. To open a handle, the user must specify:

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

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.

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 2147483647. This value is the 32 bit IP address and can be used instead of specifying the 4 address fields.

IHS => r closes the handle that sent the command; where r = -1 for UDP/IP, or r = -2 for TCP/IP.

IHT => r closes all handles except for the one sending the command; where r = -1 UDP, or r = -2 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, or -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 a -5 while attempting to establish a UDP handle

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

_IHh3 contains a 0 if the ARP was successful

contains a 1 if it has failed or is still in progress

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

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

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

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

USAGE:

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

DEFAULTS:

Default Value	-----
Default Format	-----

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. By default, input interrupts are configured for activation with a logic "0" but can be configured for activation with a logic "1" signal.

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

ARGUMENTS: II m,n,o,p where

m is an integer between 0 and 8 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 8. 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 255. 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 00001111 where the bottom 4 bits are 1 (bit 0 through bit 3) and the top 4 bits are 0 (bit 4 through bit 7). 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 255. 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 00000010), input 2 will be activated by a logic '1' and inputs 1,3, and 4 will be activated with a logic "0".

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	**All Controllers**

DEFAULTS:

Default Value	
Default Format	3.0 (mask only)

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

IK

FUNCTION: Block Ethernet ports

DESCRIPTION:

The IK command blocks the controller from receiving packets on Ethernet ports lower than 1000 except for ports 0, 23, 68, and 502.

ARGUMENTS: IKn where

n = 0 allows controller to receive Ethernet packets on any port

n = 1 blocks controller from receiving Ethernet packets on all ports lower than 1000 except for 0, 23, 68, and 502.

n = ? queries controller for value of IK

USAGE:		**DEFAULTS:**	
In a Program	Yes	Default Value	n = 1
Command Line	Yes		

OPERAND USAGE:

_IK can not be used as an operand.

RELATED COMMANDS:

TH	Tell Handles
IH	Open new Ethernet handle

EXAMPLES:

```
IK1             Blocks undesirable port communication
IK0             Allows all Ethernet ports to be used
```

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.

The IN command may only be used in thread 0.

ARGUMENTS: IN "m",n where

m is prompt message

n is the variable name

The total number of characters for n and m must be less than 40 characters.

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

USAGE:

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

DEFAULTS:

Default Value	-----
Default Format	Position Format

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
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
BGA                                 ;'Begin motion
AMA                                 ;'Wait for motion complete
MG "MOVE DONE"                      ;'Print Message
EN                                  ;'End Program
```

@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 8

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

@AN[n]	Read analog input
@OUT[n]	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
x = ?          query variable
:1.000
```

#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.

USAGE:

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

RELATED COMMANDS:

II	Input interrupt
@IN[n]	Read digital input
RI	Return from interrupt

EXAMPLES:

```
#A
II1                                    ;' arm digital input 1

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

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

***NOTE**: The automatic subroutine runs in thread 0.*

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

@INT[n]

FUNCTION: Integer part

DESCRIPTION:

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

ARGUMENTS: @INT[n]

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

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

@FRAC[n]	Fractional part

EXAMPLES:

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

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


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

IP

FUNCTION: Increment Position

DESCRIPTION:

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

Case 1: Motor is standing still

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

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

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

Case 3: Motor is in the Jog Mode

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

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

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

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

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

USAGE:

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

DEFAULTS:

Default Value	
Default Format	PF

RELATED COMMANDS:

PF	Position Formatting

EXAMPLES:

```
IP 50                 50 counts with set acceleration and speed

#CORRECT              ;Label
AC 100000             ;Set acceleration
JG 10000;BGA          ;Jog at 10000 counts/sec rate
WT 1000               ;Wait 1000 msec
IP 10                 ;Move the motor 10 counts instantaneously
STA                   ;Stop Motion
AMA                   ;Wait for ST to complete
EN                    ;End Program
```

IT

FUNCTION: Independent Time Constant - Smoothing Function

DESCRIPTION:

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

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

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

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

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

USAGE:

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

DEFAULTS:

Default Value	1
Default Format	1.4

OPERAND USAGE:

_ITn contains the value of the independent time constant for the specified 'n' axis.

RELATED COMMANDS:

PR	Position relative
PA	Position absolute
JG	Jog
VM	Vector mode
LM	Linear Interpolation Mode

EXAMPLES:

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

JG

FUNCTION: Jog

DESCRIPTION:

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

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

n is a signed numbers in the range 0 to +/-3,000,000 decimal. The units of this are steps/second. (Use JGN = n for the virtual axes)

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

USAGE:

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

DEFAULTS:

Default Value	25000
Default Format	8.0

OPERAND USAGE:

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

RELATED COMMANDS:

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

EXAMPLES:

`JG 100,500,2000,5000`	`Set for jog mode with a slew speed of 100 counts/sec for the A-axis, 500 counts/sec for the B-axis, 2000 counts/sec for the C-axis, and 5000 counts/sec for D-axis.`
`BG`	`Begin Motion`
`JG ,,-2000`	`Change the C-axis to slew in the negative direction at -2000 counts/sec.`

JP

FUNCTION: Jump to Program Location

DESCRIPTION:

The JP command causes a jump to a program location on a specified condition. The program location may be any program line number or label. The condition is a conditional statement which uses a logical operator such as equal to or less than. A jump is taken if the specified condition is true.

Multiple conditions can be used in a single jump statement. The conditional statements are combined in pairs using the operands "&" and "|". The "&" operand between any two conditions, requires that both statements must be true for the combined statement to be true. The "|" operand between any two conditions, requires that only one statement be true for the combined statement to be true. *Note: Each condition must be placed in parenthesis for proper evaluation by the controller.*

ARGUMENTS: JP location,condition where

location is a program line number or label

condition is a conditional statement using a logical operator

The logical operators are:

< less than

> greater than

= equal to

<= less than or equal to

>= greater than or equal to

<> not equal to

USAGE:

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

DEFAULTS:

Default Value	
Default Format	

RELATED COMMANDS:

JS	Jump to Subroutine
IF	If conditional statement
ELSE	Else function for use with IF conditional statement
ENDIF	End of IF conditional statement

EXAMPLES:

```
JP #POS1,V1<5       Jump to label #POS1 if variable V1 is less than 5
JP #A,V7*V8=0       Jump to #A if V7 times V8 equals 0
JP #B               Jump to #B (no condition)
```

Hint: *JP is similar to an IF, THEN command. Text to the right of the comma is the condition that must be met for a jump to occur. The destination is the specified label before the comma.*

JS

FUNCTION: Jump to Subroutine

DESCRIPTION:

The JS command will change the sequential order of execution of commands in a program. If the jump is taken, program execution will continue at the line specified by the destination parameter, which can be either a line number or label. The line number of the JS command is saved and after the next EN command is encountered (End of subroutine), program execution will continue with the instruction following the JS command. There can be a JS command within a subroutine.

Multiple conditions can be used in a single jump statement. The conditional statements are combined in pairs using the operands "&" and "|". The "&" operand between any two conditions, requires that both statements must be true for the combined statement to be true. The "|" operand between any two conditions, requires that only one statement be true for the combined statement to be true. *Note: Each condition must be placed in parenthesis for proper evaluation by the controller.*

Note: Subroutines may be nested 16 deep in the controller.

A jump is taken if the specified condition is true. Conditions are tested with logical operators. The logical operators are:

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

ARGUMENTS: JS destination, condition where

destination is a line number or label

condition is a conditional statement using a logical operator

USAGE:

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

DEFAULTS:

Default Value	
Default Format	

RELATED COMMANDS:

EN	End

EXAMPLES:

```
JS #SQUARE,V1<5      Jump to subroutine #SQUARE if V1 is less than 5
JS #LOOP,V1<>0       Jump to #LOOP if V1 is not equal to 0
JS #A                Jump to subroutine #A (no condition)
```

KS

FUNCTION: Step Motor Smoothing

DESCRIPTION:

The KS parameter sets the amount of smoothing of stepper motor pulses. This is most useful when operating in full or half step mode. 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.

Note: KS will cause a delay in the generation of output steps.

ARGUMENTS: KS n,n,n,n or KSA=n where

n is a positive number in the range between 0.5 and 64 with a resolution of 1/32.

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

USAGE:

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

DEFAULTS:

Default Value	2.000
Default Format	2.3

OPERAND USAGE:

_KSn contains the value of the stepper motor smoothing constant for the specified axis.

RELATED COMMANDS:

MT	Motor Type

EXAMPLES:

```
KS 2, 4 , 8        Specify a,b,c axes
KS 5               Specify a-axis only
KS ,,15            Specify c-axis only
```

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

LL	List Labels
LS	List Program
LV	List Variable

EXAMPLES:

```
: LA
CA [10]
LA [5]
NY [25]
VA [17]
```

LC

FUNCTION: Low Current Stepper Mode

DESCRIPTION:

Causes the amp enable line for the specified axes to toggle (disabling the stepper drives) a programmable amount of time after the respective axes stop (profiler holding position). Each axis is handled individually. This will reduce current consumption, but there will be no holding torque.

ARGUMENTS: LC n,n,n,n where

n = 0 Normal (stepper drive always on)

n = 1 Stepper drive on at a reduced current

n is an integer between 2 and 32767 specifying the number of samples to wait between the end of the move and when the amp enable line toggles

n = ? Returns the current value

USAGE:

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

DEFAULTS:

Default Value	0
Default Format	5.0

OPERAND USAGE:

_LCn contains the low current value.

RELATED COMMANDS:

MT	Motor Type

EXAMPLES:

```
MTZ=2          Specify stepper mode for the z axis
LCZ=1          Specify low current mode for the z axis and disable immediately
```

LD

FUNCTION: Limit Disable

DESCRIPTION:

Disables limit switches. Soft limits BL and FL are still in effect. This feature should be used to gain additional digital inputs if limit switches are not used, or if there is a noise problem which causes limit switch conditions even though no limit switches are connected.

ARGUMENTS: LD n,n,n,n, or LDA=n where

n = 0	enabled (default)
n = 1	forward limit disabled
n = 2	reverse limit disabled
n = 3	both disabled
n = ?	returns the current setting

USAGE:

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

DEFAULTS:

Default Value	0
Default Format	1.0

OPERAND USAGE:

_LDn contains the current value

RELATED COMMANDS:

_LFX	State of Forward limit
_LRX	State of Reverse limit
SC	Stop code
BL	Backward soft limit
FL	Forward soft limit

EXAMPLES:

```
LDX=1              Disable the forward limit switch on the X axis
```

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 move length in encoder counts for the selected coordinate system, S

USAGE:

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

DEFAULTS:

Default Value	0
Default Format	PF

OPERAND USAGE:

_LEn contains the total vector move length in steps.

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:

```
CAS              Specify S coordinated motion system
LM CD            Specify linear interpolation mode for C and D axes
LI ,,100,200     Specify linear distance
LE               End linear move
BGS              Begin motion
```

_LF

FUNCTION: Forward Limit Switch Operand

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 switch input is inactive*

_LFn = 0 when the limit switch input is active*

For CN 1:

_LFn = 0 when the limit switch input is inactive*

_LFn = 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.

EXAMPLES:

```
MG _LFA             Display the status of the A axis forward limit switch
```

LI

FUNCTION: Linear Interpolation Distance

DESCRIPTION:

The LI a,b,c,d command specifies the incremental distance of travel for each axis in the Linear Interpolation (LM) mode. LI parameters are relative distances given with respect to the current axis positions. Up to 7 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. 31 returned means the buffer is empty and 32 LI segments can be sent. A zero means the buffer is full and no additional segments can be sent. It should be noted that the controller computes the vector speed based on the axes specified in the LM mode. For example, LM ABC designates linear interpolation for the A,B and C axes. The speed of these axes will be computed from $VS^2=AS^2+BS^2+CS^2$ where AS, BS and CS are the speed of the A,B and C axes. If the LI command specifies only A and B, the speed of C will still be used in the vector calculations. The controller always uses the axis specifications from LM, not LI, to compute the speed. The parameter n is optional and can be used to define the vector speed that is attached to the motion segment.

ARGUMENTS: LI n,n,n,n,n,n,n,n <o >p or LIA=n where

n is a signed integer in the range -8,388,607 to 8,388,607 and represents the incremental move distance (at least one n must be non-zero).

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

p specifies a vector speed to be achieved at the end of the linear segment. Based on vector accel and decal rates, p is an unsigned even integer between 0 and 3,000,000 for steppers.

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

LE	Linear end
LM	Linear Interpolation Mode

EXAMPLES:

```
LM ABC              Specify linear interpolation mode
LI 1000,2000,3000   Specify distance
LE                  Last segment
BGS                 Begin sequence
```

#LIMSWI

FUNCTION: Limit switch automatic subroutine

DESCRIPTION:

Without #LIMSWI defined, the controller will effectively issue the STn on the axis when it's limit switch is tripped. With #LIMSWI defined, the axis is still stopped, and in addition, code is executed. #LIMSWI is most commonly used to turn the motor off when a limit switch is tripped (see example below). For #LIMSWI to run, 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

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**: The automatic subroutine runs in thread 0.*

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

<control>L<control>K

FUNCTION: Lock program

DESCRIPTION:

<control>L<control>K locks user access to the application program. When locked, the ED, UL, LS, and TR commands will give privilege error #106. The application program will still run when locked.

The locked or unlocked state can be saved with a BN command. Upon master reset, the controller is unlocked. Once the program is unlocked, it will remain accessible until a lock command or a reset (with the locked condition burned in) occurs.

ARGUMENTS: <control>L<control>Kpassword,n where

When n is 1, this command will lock the application program.

When n is 0, the program will be unlocked.

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

PW	Password
ED	Edit program
UL	Upload program
LS	List program
TR	Trace program

EXAMPLES:

```
:PW test,test                    Set password to "test"
:^L^Ktest,1                      Lock the program
:ED                              Attempt to edit program
?
:TC1
106 Privilege violation
:
```

LL

FUNCTION: List Labels

DESCRIPTION:

The LL command returns a listing of all of the program labels in memory and their associated line numbers. The listing will be in alphabetical order.

ARGUMENTS: None

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

LA	List Arrays
LS	List Program
LV	List Variables

EXAMPLES:

```
: LL
# FIVE=5
# FOUR=4
# ONE=1
# THREE=3
# TWO=2
```

LM

FUNCTION: Linear Interpolation Mode

DESCRIPTION:

The LM command specifies the linear interpolation mode and specifies the axes for linear interpolation. Any set of 1 thru 8 axes may be used for linear interpolation. LI commands are used to specify the travel distances for linear interpolation. The LE command specifies the end of the linear interpolation sequence. Several LI commands may be given as long as the controller sequence buffer has room for additional segments. Once the LM command has been given, it does not need to be given again unless the VM command has been used.

It should be noted that the controller computes the vector speed based on the axes specified in the LM mode. For example, LM ABC designates linear interpolation for the A,B and C axes.

The speed of these axes will be computed from $VS^2=AS^2+BS^2+CS^2$, where AS, BS and CS are the speed of the A,B and C axes. In this example, If the LI command specifies only A and B, the speed of C will still be used in the vector calculations. The controller always uses the axis specifications from LM, not LI, to compute the speed.

ARGUMENTS: LM nnnnnnnnn where

n is A,B,C,D or any combination to specify the axis or axes

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

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

OPERAND USAGE:

_LM contains the number of spaces available in the sequence buffer for the 'n' coordinate system.

RELATED COMMANDS:

LE	Linear end
LI	Linear Distance
VA	Vector acceleration
VS	Vector Speed
VD	Vector deceleration
AV	Vector distance
CS	_CS - Sequence counter

EXAMPLES:

```
LM ABCD                          Specify linear interpolation mode
VS 10000; VA 100000;VD 1000000   Specify vector speed, acceleration and deceleration
LI 200,300,400,500               Specify linear distance
LE; BGS                          Last vector, then begin motion
```

_LR

FUNCTION: Reverse Limit Switch Operand

DESCRIPTION:

The _LR operand contains the state of the reverse limit switch for the specified axis.

The operand is specified as: _LRn 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:

_LRn = 1 when the limit switch input is inactive*

_LRn = 0 when the limit switch input is active*

For CN 1:

_LRn = 0 when the limit switch input is inactive*

_LRn = 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 in the User’s Manual for further details.

EXAMPLES:

```
MG _LRA             Display the status of the A axis reverse limit switch
```

LS

FUNCTION: List Program

DESCRIPTION:

The LS command returns a listing of the programs in memory.

ARGUMENTS: LS n,m where

n and m are valid numbers from 0 to 449, or labels. n is the first line to be listed, m is the last.

n is an integer in the range of 0 to 449 or a label in the program memory. n is used to specify the first line to be listed.

m is an integer in the range of 1 to 449 or a label on the program memory. m is used to specify the last line to be listed.

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	0, Last Line
In a Program	No	Default Format	-
Command Line	Yes		
Controller Usage	**ALL CONTROLLERS**		

RELATED COMMANDS:

LA	List Arrays
LL	List Labels
LV	List Variables

EXAMPLES:

```
:LS #A,6          List program starting at #A through line 6
2 #A
3 PR 500
4 BGA
5 AM
6 WT 200
```

Hint: *Remember to quit the Edit Mode <cntrl> Q prior to giving the LS command. (DOS)*

LV

FUNCTION: List Variables

DESCRIPTION:

The LV command returns a listing of all of the program variables in memory. The listing will be in alphabetical order.

ARGUMENTS: None

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	VF

RELATED COMMANDS:

LA	List Arrays
LS	List Program
LL	List Labels

EXAMPLES:

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

LZ

FUNCTION: Leading Zeros

DESCRIPTION:

The LZ command is used for formatting the values returned from interrogation commands or interrogation of variables and arrays. By enabling the LZ function, all leading zeros of returned values will be removed.

ARGUMENTS: LZ n where

n = 1 Removes leading zeros

n = 0 Does not remove leading zeros.

n = ? Returns the state of the LZ function. '0' does not remove and '1' removes zeros

USAGE:

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

DEFAULTS:

Default Value	1
Default Format	-

OPERAND USAGE:

_LZ contains the state of the LZ function. '0' is disabled and '1' is enabled.

EXAMPLES:

```
LZ 0                Disable the LZ function
TPA                 Interrogate the controller for current position of A axis
:0000021645.0000    Value returned by the controller
VAR1=               Request value of variable "VAR1"  (previously set to 10)
:0000000010.0000    Value of variable returned by controller
LZ1                 Enable LZ function
TPA                 Interrogate the controller for current position of A axis
:21645.0000         Value returned by the controller
VAR1=               Request value of variable "VAR1"  (previously set to 10)
:10.0000            Value of variable returned by controller
```

MB

FUNCTION: Modbus

DESCRIPTION:

The MB command is used to communicate with I/O devices using the first two levels of the Modbus protocol.

The format of the command varies depending on each function code. The function code, -1, designates that the first level of Modbus is used (creates raw packets and receives raw data). The other codes are the 10 major function codes of the second level that the controller 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 D).

ARGUMENTS:

MBh = -1, len, array[] where

len is the number of the bytes

Array[] is the name of array containing data

MBh = addr, 1, m, n, array[] where

m is the starting bit number

n is the number of bits

array[] of which the first element will hold result

MBh = addr, 2, m, n, array[] where

m is the starting bit number

n is the number of bits

array[] of which the first element will hold result

MBh = addr, 3, m, n, array[] where

m is the starting register number

n is the number of registers

array[] will hold the response

MBh = addr, 4, m, n, array[] where

m is the starting register number

n is the number of registers

array[] will hold the response

MBh = addr, 5, m, n where

m is the starting bit number

n is 0 or 1 and represents the coil set to off or on.

MBh = addr, 6, m, n where

m is the register number

n is the 16 bit value

MBh = addr, 7, array[] where

array[] is where the returned data is stored (one byte per element)

MBh = addr, 15, m, n, array[] where

m is the starting bit number

n is the number of bits

array[] contains the data (one byte per element)

MBh = addr, 16, m, n, array[] where

m is the starting register number

n is the number of registers

array[] contains the data (one 16 bit word per element)

MBh = addr, 17, array[] where

array[] is where the returned data is stored

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

Note: Port 502 must be used as the Modbus Ethernet handle. See the IH command for more info on how to open a handle with a specific port number.

MC

FUNCTION: Motion Complete - "In Position"

DESCRIPTION:

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 recommended when operating with stepper motors since the generation of step pulses can be delayed due to the stepper motor smoothing function, KS. In this case, the MC command would only be satisfied after all steps are generated.

ARGUMENTS: MC nnnn where

n is A,B,C,D or any combination to specify the axis or axes

No argument specifies that motion on all axes is complete.

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

BG	Begin
AM	After Move

EXAMPLES:

```
#MOVE                ;'Program MOVE
 PR 2000,4000        ;'Independent Move on A and B axis
 BG AB               ;'Start the B-axis
 MC AB               ;'After the move is complete on T coordinate system,
 MG "DONE";  TP      ;'Print message
EN                   ;'End of Program
```

Hint: *MC can be used to verify that the actual motion has been completed.*

MF

FUNCTION: Forward Motion to Position

DESCRIPTION:

* When using a stepper motor, this condition is satisfied when the stepper position (as determined by the output buffer) has crossed the specified Forward Motion Position.

ARGUMENTS: MF n,n,n,n, or MFA=n where

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

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

AR	Trippoint for after Relative Distances
AP	Trippoint for after Absolute Position

EXAMPLES:

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

Hint: *The accuracy of the MF command is the number of counts that occur in 2*TM μsec. Multiply the speed by 2*TM μsec to obtain the maximum error. MF tests for absolute position. The MF command can also be used when the specified motor is driven independently by an external device.*

MG

FUNCTION: Message

DESCRIPTION:

The MG command sends data out the bus. This can be used to alert an operator, send instructions or return a variable value.

ARGUMENTS: MG "m", {^n}, V {Fm.n or $m,n} {N} {Ex} {Pn} where

"m" is a text message including letters, numbers, symbols or <ctrl>G (up to 72 characters).

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

V is a variable name or array element where the following formats can be used:

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

{Zm.n} Same as {Fm.n} but suppresses the leading zeros.

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

{Sn} Display variable as a string of length n where n is 1 through 6

{N} Suppress carriage return line feed.

{Ex} Sends the message out the Ethernet handle x, where x is A,B,C,D

{P} Sends the message out the Serial port.

{Lx} Sends the message to the LCD, where x is 1 or 2 for the top or bottom line of the LCD.

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

EXAMPLES:

Case 1: Message command displays ASCII strings

MG "Good Morning" Displays the string

Case 2: Message command displays variables or arrays

MG "The Answer is", Total {F4.2} Displays the string with the content of variable 'Total' in local format of 4 digits before and 2 digits after the decimal point.

Case 3: Message command sends any ASCII characters to the port.

MG {^13}, {^10}, {^48}, {^055} displays carriage return and the characters 0 and 7.

MO

FUNCTION: Motor Off

DESCRIPTION:

The MO command shuts off the control algorithm. The controller will continue to monitor the motor position. To turn the motor back on use the Servo Here command (SH).

ARGUMENTS: MO nnnnnn where

n is A,B,C,D or any combination to specify the axis or axes.

No argument specifies all axes.

USAGE:

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

DEFAULTS:

Default Value	0
Default Format	1.0

OPERAND USAGE:

_MOn contains the state of the motor for the specified axis.

RELATED COMMANDS:

SH	Servo Here

EXAMPLES:

`MO`	`Turn off all motors`
`MOA`	`Turn off the A motor. Leave the other motors unchanged`
`MOB`	`Turn off the B motor. Leave the other motors unchanged`
`MOCA`	`Turn off the C and A motors. Leave the other motors unchanged`
`SH`	`Turn all motors on`
`Bob=_MOA`	`Sets Bob equal to the A-axis servo status`
`Bob=`	`Return value of Bob. If 1, in motor off mode, If 0, in servo mode`

Hint: *The MO command is useful for positioning the motors by hand. Turn them back on with the SH command.*

MR

FUNCTION: Reverse Motion to Position

DESCRIPTION:

* When using a stepper motor, this condition is satisfied when the stepper position (as determined by the output buffer) has crossed the specified Reverse Motion Position.

ARGUMENTS: MR n,n,n,n or MRA=n where

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

USAGE:

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

DEFAULTS:

Default Value	
Default Format	

RELATED COMMANDS:

AR	Trippoint for Relative Distances
AP	Trippoint after Absolute Position

EXAMPLES:

```
#TEST                  ;'Program Test
 DP0                   ;'Define zero
 JG -1000              ;'Jog mode (speed of 1000 counts/sec)
 BG A                  ;'Begin move
 MF -3000              ;'After passing the position -3000
 V1=_TPA               ;'Assign V1 A position
 MG "Position is", V1  ;'Print Message
 ST                    ;'Stop
EN                     ;'End of Program
```

Hint: *The accuracy of the MR command is the number of counts that occur in 2*TM μsec. Multiply the speed by 2*TM μsec 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:

Sets the step and direction signaling behavior.

ARGUMENTS: MT n,n,n,n or MTA=n where

n = -2	Specifies Step motor with active high step pulses
n = 2	Specifies Step motor with active low step pulses
n = -2.5	Specifies Step motor with reversed direction and active high step pulses
n = 2.5	Specifies Step motor with reversed direction and active low step pulses
n = ?	Returns the value of the motor type for the specified axis.

USAGE:

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

DEFAULTS:

Default Value	2,2,2,2
Default Format	1.1

OPERAND USAGE:

_MTn contains the value of the motor type for the specified axis.

RELATED COMMANDS:

CE	Configure encoder type

EXAMPLES:

```
MT 2,2,2,2
MT ?,?              Interrogate motor type
V=_MTA              Assign motor type to variable
```

MW

FUNCTION: Modbus Wait

DESCRIPTION:

Enabling the MW command causes the controller to hold up execution of the program after sending a Modbus command until a response from the Modbus device has been received. If the response is never received, then the #TCPERR subroutine will be triggered and an error code of 123 will occur on _TC.

ARGUMENTS: MWn where

n = 0 Disables the Modbus Wait function

n = 1 Enables the Modbus Wait function

USAGE:

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

DEFAULTS:

Default Value	1
Default Format	1.0

OPERAND USAGE:

MW? contains the state of the Modbus Wait.

_MW contains returned function code

_MW1 contains returned error code

RELATED COMMANDS:

MB Modbus

EXAMPLES:

```
MW1                 Enables Modbus Wait
SB1001              Set Bit 1 on Modbus Handle A
CB1001              Clear Bit 1 on Modbus Handle A
```

Hint: *The MW command ensures that the command that was sent to the Modbus device was successfully received before continuing program execution. This prevents the controller from sending multiple commands to the same Modbus device before it has a chance to execute them.*

NO (‘ apostrophe also accepted)

FUNCTION: No Operation

DESCRIPTION:

The NO or an apostrophe (‘) command performs no action in a sequence, but can be used as a comment in a program. This helps to document a program.

ARGUMENTS: NO m where

m is any group of letters and numbers

up to 32 characters can follow the NO command

USAGE:

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

DEFAULTS:

Default Value	
Default Format	

OPERAND USAGE:

_NO returns a bit field indicating which threads are running. For example, 0 means no threads are running, 1 means only thread 0 is running, 3 means threads 0 and 1 are running, and 15 means all 4 threads are running).

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
```

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, *expressio*n where

n denotes the output bit

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

USAGE:

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

DEFAULTS:

Default Value	
Default Format	

EXAMPLES:

```
OB 1, POS=1          If POS 1 is non-zero, Bit 1 is high.
                     If POS 1 is zero, Bit 1 is low
OB 2, @IN[1]&@IN[2]  If Input 1 and Input 2 are both high, then
                     Output 2 is set high
OB 3, COUNT[1]       If the element 1 in the array is zero, clear bit 3
OB N, COUNT[1]       If element 1 in the array is zero, clear bit N
```

OE

FUNCTION: Off On Error

DESCRIPTION:

Sets axis to stop when SPM Error.

ARGUMENTS: OE n,n,n,n,n or OEA=n where

n = 1	Motor will stop on SPm error.
n=2	Motor shut off (MO)by limit switches.
n=3	Motor shut off by SPM, limit switch, or abort

USAGE:

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

DEFAULTS:

Default Value	0
Default Format	1.0

OPERAND USAGE:

_OEn contains the status of the off on error function for the specified axis.

RELATED COMMANDS:

	Abort
SH	Servo Here
#POSERR	Error Subroutine
#LIMSWI	Limit switch automatic subroutine

EXAMPLES:

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

Hint: *The OE command is useful for preventing system damage.*

OP

FUNCTION: Output Port

DESCRIPTION:

The OP command sends data to the output ports of the controller. You can use the output port to control external switches and relays.

ARGUMENTS: OP m where

m is an integer in the range 0 to 15 decimal, or $0 to $F hexadecimal.

m is the decimal representation of the general output bits. 1 through Output 4.

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	5.0
Command Line	Yes		
Controller Usage	**ALL CONTROLLERS**		

OPERAND USAGE:

_OP0 contains the value of the first argument, m

RELATED COMMANDS:

SB	Set output bit
CB	Clear output bit

EXAMPLES:

```
OP 0              Clear Output Port -- all bits
OP $05            Set outputs 1,3; clear the others
MG _OP0           Returns the first parameter "m"
```

@OUT[n]

FUNCTION: Read digital output

DESCRIPTION:

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

ARGUMENTS: @OUT[n] where

n is an unsigned integer in the range 1 to 4

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

@AN[n]	Read analog input
@IN[n]	Read digital input
SB	Set digital output bit
CB	Clear digital output bit
OF	Set analog output offset

EXAMPLES:

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

PA

FUNCTION: Position Absolute

DESCRIPTION:

The PA command will set the final destination of each axis. The position is referenced to the absolute zero.

ARGUMENTS: PA n,n,n,n, or PAA=n where

n is a signed integers in the range -2147483648 to 2147483647 decimal. Units are in encoder counts.

n = ? Returns the commanded position at which motion stopped.

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	Position Format

OPERAND USAGE:

_PAn contains the last commanded position at which motion stopped.

RELATED COMMANDS:

PR	Position relative
SP	Speed
AC	Acceleration
DC	Deceleration
BG	Begin
PF	Position Formatting

EXAMPLES:

```
PA 400,-600,500,200     A-axis will go to 400 counts B-axis will go to –600
                        counts C-axis will go to 500 counts D-axis will go to
                        200 counts
BG;AM                   Execute Motion and Wait for Motion Complete
PA ?,?,?,?              Returns the current commanded position after motion has
                        completed
:400, -600, 500, 200
BG                      Start the move
PA 700                  A-axis will go to 700 on the next move while the
BG                      B,C and D-axis will travel the previously set relative
                        distance if the preceding move was a PR move, or will
                        not move if the preceding move was a PA move.
```

PF

FUNCTION: Position Format

DESCRIPTION:

The PF command allows the user to format the position numbers such as those returned by TP. The number of digits of integers and the number of digits of fractions can be selected with this command. An extra digit for sign and a digit for decimal point will be added to the total number of digits. If PF is negative, the format will be hexadecimal and a dollar sign will precede the characters. Hex numbers are displayed as 2's complement with the first bit used to signify the sign.

If a number exceeds the format, the number will be displayed as the maximum possible positive or negative number (i.e. 999.99, -999, $8000 or $7FF).

The PF command can be used to format values returned from the following commands:

BL ?	LE ?
	PA ?
DP ?	PR ?
EM ?	
FL ?	VE ?
IP ?	TE
TP	

ARGUMENTS: PF m.n where

m is an integer between -8 and 10 which represents the number of places preceding the decimal point. A negative sign for m specifies hexadecimal representation.

n is an integer between 0 and 4 which represent the number of places after the decimal point.

n = ? Returns the value of m.

USAGE:

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

DEFAULTS:

Default Value	10.0
Default Format	2.1

OPERAND USAGE:

_PF contains the value of the position format parameter.

EXAMPLES:

```
TPX                 Tell position of X
:0                  Default format
PF 5.2              Change format to 5 digits of integers and 2 of fractions
TPX                 Tell Position
:21.00
PF-5.2              New format.  Change format to hexadecimal
TPX                 Tell Position
:$00015.00          Report in hex
```

P1CD

FUNCTION: Serial port 1 code

DESCRIPTION:

P1CD returns the status of the serial port

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

P1CH	Serial port 1 character
	Serial port1 number
P1NM	Serial port 1 string
CI	Configure #COMINT
#COMINT	Communication interrupt automatic subroutine

EXAMPLES:

```
:MG "TEST" {P1} ;'send a message to the hand terminal
:MG P1CD ;'no characters entered on hand terminal
 0.0000
:MG P1CD ;'the number 6 was pushed on the hand terminal
 1.0000
:MG P1CD ;'enter key pushed on hand terminal
 3.0000
:MG P1CD ;'the character B was pushed (shift f2) then enter
 2.0000
```

P1CH

FUNCTION: Serial port 1 character

DESCRIPTION:

P1CH returns the last character sent to the serial port.

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

P1CD	Serial port1 code
P1NM	Serial port1 number
P1ST	Serial port 1 string
CI	Configure #COMINT
#COMINT	Communication interrupt automatic subroutine

P1NM

FUNCTION: Serial port 1 number

DESCRIPTION:

P1NM converts 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 serial port

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

P1CD	Serial port 1 code
P1CH	Serial port1 character
P1ST	Serial port 1 string
CI	Configure #COMINT
#COMINT	Communication interrupt automatic subroutine

P1ST

FUNCTION: Serial port 1 string

DESCRIPTION:

P1ST returns the last string (followed by carriage return) NO MORE THAN SIX CHARACTERS CAN BE ACCESSED.

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

P1CD	Serial port 1 code
P1CH	Serial port 1 character
P1NM	Serial port 1 number
CI	Configure #COMINT
#COMINT	Communication interrupt automatic subroutine

#POSERR

FUNCTION: Position error automatic subroutine

DESCRIPTION:

Automate subroutine run when S.P.M. determines a stepper error.

USAGE:

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

RELATED COMMANDS:

OE	Off an error
YA	Step Drive Resolution
YB	Step Motor Resolution
YE	Encoder Resolution
YR	Error Correction
YS	Stepper Position Maintenance. Mode (SPM) Enable
Q5	Error Magnitude

EXAMPLES:

```
#A              ;'"Dummy" program
JP #A

#POSERR         ;'Position error routine
  MG "TE > ER" ;'Send message
RE1             ;'Return to main program
```

***NOTE**: The automatic subroutine runs in thread 0.*

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

PR

FUNCTION: Position Relative

DESCRIPTION:

The PR command sets the incremental distance and direction of the next move. The move is referenced with respect to the current position. .

ARGUMENTS: PR n,n,n,n or PRA=n where

n is a signed integer in the range -2147483648 to 2147483647 decimal. Units are in encoder counts

n = ? Returns the current incremental distance for the specified axis.

USAGE:

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

DEFAULTS:

Default Value	0
Default Format	Position Format

OPERAND USAGE:

_PRn contains the current incremental distance for the specified axis.

RELATED COMMANDS:

PA	Position Absolute
BG	Begin
AC	Acceleration
DC	Deceleration
SP	Speed
IP	Increment Position
PF	Position Formatting

EXAMPLES:

```
PR 100,200,300,400      On the next move the A-axis will go 100 counts,
BG                      the B-axis will go to 200 counts forward, C-axis will
                        go 300 counts and the D-axis will go 400 counts.
PR ?,?,?                Return relative distances
:100, 200, 300
PR 500                  Set the relative distance for the A axis to 500
BG                      The A-axis will go 500 counts on the next move while
                        the B-axis will go its previously set relative
                        distance.
```

PT

FUNCTION: Position Tracking

DESCRIPTION:

The PT command will place the controller in the position tracking mode. In this mode, the controller will allow the user to issue absolute position commands on the fly. The motion profile is trapezoidal with the parameters controlled by acceleration, deceleration, and speed (AD, DC, SP). The absolute position may be specified such that the axes will begin motion, continue in the same direction, reverse directions, or decelerate to a stop. When an axis is in the special mode, the ST command, will exit the mode. The PA command is used to give the controller an absolute position target. Motion commands other than PA are not supported in this mode.

ARGUMENTS: PT n,n,n,n

n=0 or 1 where 1 designates the controller is in the special mode.

n=? returns the current setting

USAGE:

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

DEFAULTS:

Default Value	0
Default Format	1.0

RELATED COMMANDS:

PA	Position Absolute
AC	Acceleration
DC	Deceleration
SP	Speed

EXAMPLES:

```
#A
 PT1,1,1,1          ;'Enable the position tracking mode for axes X, Y, Z, and W
#LOOP               ;'Create label #LOOP in a program.  This small program will
                    ;'update the absolute position at 100 Hz.  Note that the
                    ;'user must update the variables V1, V2, V3 and V4 from the
                    ;'host PC, or another thread operating on the controller.
 PAV1,V2,V3,V4      ;'Command XYZW axes to move to absolute positions.  Motion
                    ;'begins when the command is processed.  BG is not required
                    ;'to begin motion in this mode.  In this example, it is
                    ;'assumed that the user is updating the variables at a
                    ;'specified rate.  The controller will update the new
                    ;'target position every 10 milliseconds (WT10).
 WT10               ;'Wait 10 milliseconds
JP#LOOP             ;'Repeat by jumping back to label LOOP
```

Special Notes: *The AM, and MC trip points are not valid in this mode. It is recommended to use MF and MR as trip points with this command, as they allow the user to specify both the absolute position, and the direction. The AP trip point may also be used.*

PW

FUNCTION: Password

DESCRIPTION:

The password can be set with the command PW password,password where the password can be up to 8 alphanumeric characters. The default value after master reset is a null string. The password can only be changed when the controller is in the unlocked state (^L^K). The password is burnable but cannot be interrogated. If you forget the password you must master reset the controller to gain access.

ARGUMENTS: PW n,n where

n is a string from 0 to 8 characters in length

USAGE:

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

DEFAULTS:

Default Value	"" (null string)
Default Format	-

RELATED COMMANDS:

<control>L<control>K	Lock/Unlock
ED	Edit program
UL	Upload program
LS	List program
TR	Trace program

EXAMPLES:

```
:PWtest,test                    Set password to "test"
:^L^K test,1                    Lock the program
:ED                             Attempt to edit program
?
:TC1
106 Privilege violation
:
PWtest,test                     Set the password to "test"
```

QD

FUNCTION: Download Array

DESCRIPTION:

The QD command transfers array data from the host computer to the controller. QD array[], start, end requires that the array name be specified along with the index of the first element of the array and the index of the last element of the array. The array elements can be separated by a comma (,) or by <CR> <LF>. The downloaded array is terminated by a \.

ARGUMENTS: QD array[],start,end where

array[] is valid array name

start is index of first element of array (default=0)

end is index of last element of array (default = size-1)

USAGE:

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

DEFAULTS:

Default Value	start=0, end=size-1
Default Format	

RELATED COMMANDS:

QU	Upload array

HINT:

Using Galil terminal software, the command can be used in the following manner:

1. Set the timeout to 0
2. Send the command QD

3a. Use the send file command to send the data file.

OR

3b. Enter data manually from the terminal. End the data entry with the character '\'

QR

FUNCTION: Data Record

DESCRIPTION:

The QR command causes the controller to return a record of information regarding controller status. This status information includes 4 bytes of header information and specific blocks of information as specified by the command arguments.

ARGUMENTS: QR nnnnnn where

n is A,B,C,D, or I or any combination to specify the axis, axes, sequence, or I/O status

S represent the S coordinated motion planeI represents the status of the I/O

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

Note: The Galil windows terminal will not display the results of the QR command since the results are in binary format.

QS

FUNCTION: Error Magnitude

DESCRIPTION:

The QS command reports the magnitude of error, in step counts, for axes in Stepper Position Maintenance mode. A step count is directly proportional to the resolution of the step drive.

ARGUMENTS: QS nnnn or QSn = ? where

n is A,B,C,D, or any combination to specify the axis or axes

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	1.4
Command Line	Yes		

OPERAND USAGE:

_QSn contains the error magnitude in drive step counts for the given axis.

RELATED COMMANDS:

YA	Step Drive Resolution
YB	Step Motor Resolution
YC	Encoder Resolution
YR	Error Correction
YS	Stepper Position Maintenance Mode Enable, Status

EXAMPLES:

```
1. For Q microstepping drive, query the error of B axis:
    :QSB=?
    :253            This shows 253 step counts of error.  The drive
                    resolution is 64 microsteps per full motor step, nearly
                    four full motor steps of error.
2. Query the value of all axes:
    :QS
                    Response shows all axes error values
:0,253,0,0,0,0,0,0
```

Notes:

1. *When QS exceeds three full motor steps of error, the YS command indicates the excessive position error condition by changing to 2. This condition also executes the #POSERR automatic subroutine if included in the runtime code.*
2. *The operand use of the QS command can be used in conjunction with the YR command to correct for position error. See the YR command for more details.*

QU

FUNCTION: Upload Array

DESCRIPTION:

The QU command transfers array data from the controller to a host computer. The QU requires that the array name be specified along with the first element of the array and last element of the array. The uploaded array will be followed by a <control>Z as an end of text marker.

ARGUMENTS: QU array[],start,end,delim where

"array[]" is a valid array name

"start" is the first element of the array (default=0)

"end" is the last element of the array (default = last element)

"delim" specifies the character used to delimit the array elements. If delim is 1, then the array elements will be separated by a comma. Otherwise, the elements will be separated by a carriage return.

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

RELATED COMMANDS:

QD	Download array

QZ

FUNCTION: Return/ Data Record information

DESCRIPTION:

The QZ command is an interrogation command that returns information regarding data record transfers. 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:		**DEFAULTS:**	
While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	
Command Line	Yes		
Controller Usage	**ALL CONTROLLERS**		

RELATED COMMANDS:

DR	Ethernet data record update rate

RA

FUNCTION: Record Array

DESCRIPTION:

The RA command selects one through four arrays for automatic data capture. The selected arrays must be dimensioned by the DM command. The data to be captured is specified by the RD command and time interval by the RC command.

ARGUMENTS: RA n [],m [],o [],p [] where

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

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

DM	Dimension Array
RD	Record Data
RC	Record Interval

EXAMPLES:

```
#Record             ;'Label
DM POS[100]         ;'Define array
RA POS[]            ;'Specify Record Mode
RD _TPA             ;'Specify data type for record
RC 1                ;'Begin recording at 2 msec intervals
PR 1000;BG          ;'Start motion
EN                  ;'End
```

Hint: *The record array mode is useful for recording the real-time motor position during motion. The data is automatically captured in the background and does not interrupt the program sequencer. The record mode can also be used for a teach or learn of a motion path.*

RC

FUNCTION: Record

DESCRIPTION:

The RC command begins recording for the Automatic Record Array Mode (RA). RC 0 stops recording .

ARGUMENTS: RC n,m where

n is an integer 1 thru 8 and specifies 2^n samples between records. RC 0 stops recording.

m is optional and specifies the number of records to be recorded. If m is not specified, the DM number will be used. A negative number for m causes circular recording over array addresses 0 to m-1. The address for the array element for the next recording can be interrogated with _RD.

n = ? Returns status of recording. '1' if recording, '0' if not recording.

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

OPERAND USAGE:

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

RELATED COMMANDS:

DM	Dimension Array
RD	Record Data
QZ	Record Array Mode

EXAMPLES:

```
#RECORD                ;'Record
DM POS                 ;'Define Array
RA T800                ;'Specify Record Mode
RD POS                 ;'Specify Data Type
RC TDA                 ;'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:

TIME	Time in servo sample as read by the TIME command
_DEn	2nd encoder
_TPn	Position
_TDn	Stepper Position
_SHn	Commanded position
_RLn	Latched position
_TI	Inputs
_OP	Outputs
_TSn	Switches, only 0-3 bits valid
_SCn	Stop code

where 'n' is the axis specifier, A…D

ARGUMENTS: RD m_1, m_2, m_3, m_4, where

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

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

OPERAND USAGE:

_RD contains the address for the next array element for recording.

RELATED COMMANDS:

RA	Record Array
RC	Record Interval
DM	Dimension Array

EXAMPLES:

```
DM ERRORA[50],ERRORB[50]        Define array
RA ERRORA[],ERRORB[ ]           Specify record mode
RD _TEA,_TEB                    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. A motion trippoint such as MF or MR requires the axis to be actively profiling in order to be restored with RE1. RE0 clears the trippoint. To avoid returning to the main program on an interrupt, use the ZS command to zero the subroutine stack.

ARGUMENTS: RE n where

n = 0	Clears the interrupted trippoint
n = 1	Restores state of trippoint

no argument clears the interrupted trippoint

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

#POSERR	Error Subroutine
#LIMSWI	Limit Subroutine

EXAMPLES:

```
#A;JP #A;EN          ;'Label for main program
#POSERR              ;'Begin Error Handling Subroutine
MG "ERROR"           ;'Print message
SB1                  ;'Set output bit 1
RE                   ;'Return to main program and clear trippoint
```

REM

FUNCTION: Remark

DESCRIPTION:

REM is used for comments. The REM statement is NOT a controller command. Rather, it is recognized by Galil PC software, which strips away the REM lines before downloading the DMC file to the controller. REM differs from NO (or ‘) in the following ways:

(1) NO comments are downloaded to the controller and REM comments aren’t

(2) NO comments take up execution time and REM comments don’t; therefore, REM should be used for code that needs to run fast.

(3) REM comments cannot be recovered when uploading a program but NO comments are recovered. Thus the uploaded program is less readable with REM.

(4) NO comments take up program line space and REM lines don’t.

(5) REM comments must be the first and only thing on a line, whereas NO can be used to place comments to the right of code on the same line.

NO (or ‘) should be used instead of REM unless speed or program space is an issue.

ARGUMENTS: REM n where

n is a text string comment

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

NO (‘ apostrophe also accepted)	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. A motion trippoint such as MF or MR requires the axis to be actively profiling in order to be restored with RI1. RI0 clears the trippoint. To avoid returning to the main program on an interrupt, use the command ZS to zero the subroutine stack. This turns the jump subroutine into a jump only.

ARGUMENTS: RI n where

n = 0	Clears the interrupted trippoint
n = 1	Restores state of trippoint

no argument clears the interrupted trippoint

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

#ININT	Input interrupt subroutine
II	Enable input interrupts

EXAMPLES:

```
#A;II1;JP #A;EN          ;'Program label
#ININT                   ;'Begin interrupt subroutine
MG "INPUT INTERRUPT"     ;'Print Message
SB 1                     ;'Set output line 1
RI 1                     ;'Return to the main program and restore trippoint
```

RL

FUNCTION: Report Latched Position

DESCRIPTION:

The RL command will return the last position captured by the latch. The latch must first be armed by the AL command and then a 0 must occur on the appropriate input. Each axis uses a specific general input for the latch input:

X (A)	axis latch	Input	1
Y (B)	axis latch	Input	2
Z (C)	axis latch	Input	3
W (D)	axis latch	Input	4

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

Note: The Latch Function works with the main encoder. When working with a stepper motor without an encoder, the latch can be used to capture the stepper position. To do this, place a wire from the controller Step (PWM) output into the main encoder input, channel A+. Connect the Direction (sign) output into the channel B+ input. Configure the main encoder for Step/Direction using the CE command. The latch will now capture the stepper position based on the pulses generated by the controller.

ARGUMENTS: RL nnnnnn where

n is X,Y,Z,W,A,B,C,D, or any combination to specify the axis or axes

USAGE:

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

DEFAULTS:

Default Value	0
Default Format	Position Format

OPERAND USAGE:

_RLn contains the latched position of the specified axis.

RELATED COMMAND:

AL	Arm Latch

EXAMPLES:

```
JG ,5000    Set up to jog the B-axis
BGB         Begin jog
ALB         Arm the B latch; assume that after about 2 seconds, input goes low
RLB         Report the latch
:10000
```

@RND[n]

FUNCTION: Round

DESCRIPTION:

Rounds the given number to the nearest integer

ARGUMENTS: @RND[n]

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

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

@INT[n]	Truncates to the nearest integer

EXAMPLES:

```
:MG @RND[1.2]
 1.0000
:MG @RND[5.7]
 6.0000
:MG @RND[-1.2]
 -1.0000
:MG @RND[-5.7]
 -6.0000
:MG @RND[5.5]
 6.0000
:MG @RND[-5.5]
 -5.0000
:
```

RP

FUNCTION: Reference Position

DESCRIPTION:

This command returns the commanded reference position of the motor(s).

ARGUMENTS: RP nnnnnn where

n is A,B,C,D, or N, or any combination to specify the axis or axes`

USAGE:

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

DEFAULTS:

Default Value	0
Default Format	Position Format

OPERAND USAGE:

_RPn contains the commanded reference position for the specified axis.

RELATED COMMAND:

TP	Tell Position

EXAMPLES: Assume that ABC and D axes are commanded to be at the positions 200, -10, 0, -110 respectively. The returned units are in quadrature counts.

```
PF 7                                Position format of 7
LZ0                                 Turn leading zeroes on
RP
0000200,-0000010,0000000,-0000110   Return A,B,C,D reference positions
RPA
0000200                             Return the A motor reference position
RPB
-0000010                            Return the B motor reference position
PF-6.0                              Change to hex format
RP
$0000C8,$FFFFF6,$000000,$FFFF93     Return A,B,C,D in hex
Position =_RPA                      Assign the variable, Position, the value
                                    of RPA
```

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.

RS-1 Soft master reset. Restores factory defaults without changing EEPROM. To restore EEPROM settings use RS with no arguments.

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	0
In a Program	No	Default Format	3.0
Command Line	Yes		
Controller Usage	**ALL CONTROLLERS**		

OPERAND USAGE:

_RS returns the state of the processor on its last power-up condition. The value returned is the decimal equivalent of the 4 bit binary value shown below.

Bit 3	For master reset error (there should be no program to execute)
Bit 2	For program check sum error
Bit 1	For parameter check sum error
Bit 0	For variable check sum error

<control>R<control>S

FUNCTION: Master Reset

DESCRIPTION:

This command resets the controller to factory default settings and erases EEPROM.

A master reset can also be performed by installing a jumper on the controller at the location labeled MRST and resetting the controller (power cycle or pressing the reset button). Remove the jumper after this procedure.

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

<control>R<control>V

FUNCTION: Revision Information

DESCRIPTION:

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

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

SA

FUNCTION: Send command

DESCRIPTION:

SA sends a command form one controller to another via Ethernet.

NOTE: A wait statement (e.g. WT5) must be inserted between successive calls to SA.

ARGUMENTS: SAh=arg or SAh=arg, arg, arg, arg, arg, arg, arg, arg, where

h is the handle being used to send commands to the slave controller.

arg is a number, controller operand, variable, mathematical function, or string; The range for numeric values is 4 bytes of integer (2^{31}) followed by two bytes of fraction (+/- 2,147,483,647.9999). The maximum number of characters for a string is 38 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: KI1,2

USAGE:

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

DEFAULTS:

Default Value	-----
Default Format	-----

OPERAND USAGE:

_SAhn gives the value of the response to the command sent with an SA command. The h value represents the handle A thru D and the n value represents the specific field returned from the controller (0-3). If the specific field is not used, the operand will be –2^31.

RELATED COMMAND:

MG	Display messages
IH	Opens handle

EXAMPLES:

```
#A
IHA=10,0,0,12        ;'Configures handle A to be connected to a controller with
                     ;'the IP address 10.0.0.12
#B;JP#B,_IHA2<>-2    ;'Wait for connection
SAA="KI", 1, 2       ;'Sends the command to handle A (slave controller):  KI 1,2
WT5
SAA="TE"             ;'Sends the command to handle A (slave controller):  TE
WT5
MG_SAA0              ;'Display the content of the operand_SAA (first response to
                     ;'TE command)
: 132                --response from controller--
MG_SAA1              ;'Display the content of the operand_SAA (2nd response to TE
                     ;'command)
: 12                 --response from controller--
SAA="TEMP=",16       ;'Sets variable temp equal to 16 on handle A controller
EN                   ;'End Program
```

SB

FUNCTION: Set Bit

DESCRIPTION:

The SB command sets one of the output bits.

ARGUMENTS: SB n where

n is an integer which represents a specific controller output bit to be set high (output = 1).

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

RELATED COMMAND

CB	Clear Bit

EXAMPLES:

```
SB 4            Set output line 4
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	10	Stopped after homing (HM)
1	Motors stopped at commanded independent position	11	Stopped by Selective Abort Input
2	Decelerating or stopped by FWD limit switch or soft limit FL		
3	Decelerating or stopped by REV limit switch or soft limit BL		
4	Decelerating or stopped by Stop Command (ST)	16	Stepper Position Maintenance Mode error exceeded (QS)
6	Stopped by Abort input	50	Contour running
7	Stopped by Abort command (AB)	51	Contour Stop
8	Decelerating or stopped by Off on Error (OE1)		
9	Stopped after Finding Edge (FE)	100	Motors are running, vector sequence
		101	Motors stopped at commanded vector

ARGUMENTS: SC nnnnnn where

n is A,B,C,D, or any combination to specify the axis or axes

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	3.0

OPERAND USAGE:

_SCn contains the value of the stop code for the specified axis.

RELATED COMMANDS:

LCD Update

EXAMPLES:

```
Tom =_SCD          Assign the Stop Code of D to variable Tom
```

SD

FUNCTION: Switch Deceleration

DESCRIPTION:

The Limit Switch Deceleration command (SD) sets the linear deceleration rate of the motors when a limit switch has been reached. The parameters will be rounded down to the nearest factor of 1024 and have units of counts per second squared.

ARGUMENTS: SD n,n,n,n, or SDA=n where

n is an unsigned numbers in the range 1024 to 1073740800

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

USAGE:

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

DEFAULTS:

Default Value	256000
Default Format	10.0

* SD command cannot be specified while moving.

OPERAND USAGE:

_SDn contains the deceleration rate for the specified axis.

RELATED COMMANDS:

AC	Acceleration
DC	Deceleration
PR	Position Relative
PA	Position Absolute
SP	Speed

EXAMPLES:

```
PR 10000                Specify position
AC 2000000              Specify acceleration rate
DC 1000000              Specify deceleration rate
SD 5000000              Specify Limit Switch Deceleration Rate
SP 5000                 Specify slew speed
```

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

SH

FUNCTION: Servo Here

DESCRIPTION:

The SH commands tells the controller to the amplifer.

This command can be useful when the position of a motor has been manually adjusted following a motor off (MO) command.

ARGUMENTS: SH nnnn where

n is A,B,C,D or any combination to specify the axis or axes

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

MO	Motor-off

EXAMPLES:

`SH`	`Activate A,B,C,D motors`
`SHA`	`Only activate the A motor, the B,C and D motors remain in its previous state.`
`SHB`	`Activate the B motor; leave the A,C and D motors unchanged`
`SHC`	`Activate the C motor; leave the A,B and D motors unchanged`
`SHD`	`Activate the D motor; leave the A,B and C motors unchanged`

Note: *The SH command changes the coordinate system. Therefore, all position commands given prior to SH, must be repeated. Otherwise, the controller produces incorrect motion.*

@SIN[n]

FUNCTION: Sine

DESCRIPTION:

Returns the sine of the given angle in degrees

ARGUMENTS: @SIN[n] where

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

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

@ASIN[n]	Arc sine
@COS[n]	Cosine
@ATAN[n]	Arc tangent
@ACOS[n]	Arc cosine
@TAN[n]	Tangent

EXAMPLES:

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

SL

FUNCTION: Single Step

DESCRIPTION:

For debugging purposes. Single Step through the program after execution has paused at a breakpoint (BK). Optional argument allows user to specify the number of lines to execute before pausing again. The BK command resumes normal program execution.

ARGUMENTS: SL n where

n is an integer representing the number of lines to execute before pausing again

USAGE:

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

DEFAULTS:

Default Value	1

RELATED COMMANDS:

BK	Breakpoint
TR	Trace

EXAMPLES:

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

SM

FUNCTION: Subnet Mask

DESCRIPTION:

The SM command assigns a subnet mask to the controller. All packets sent to the controller whose source IP address is not on the subnet will be ignored by the controller. For example, for SM 255, 255, 0, 0 and IA 10, 0, 51, 1, only packets from IP addresses of the form 10.0.xxx.xxx will be accepted.

ARGUMENTS: SM sm0, sm1, sm2, sm3 or SM n where

sm0, sm1, sm2, sm3 are 1 byte numbers (0 to 255) separated by commas and represent the individual fields of the subnet mask.

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

SM? will return the subnet mask of the controller

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes

DEFAULTS:

Default Value	SM 0, 0, 0, 0
Default Format	

OPERAND USAGE:

_SM0 contains the subnet mask representing a 32 bit signed number (Two's complement)

RELATED COMMANDS:

IH	Internet Handle
IA	IP address

EXAMPLES:

SM 255, 255, 255, 255	Ignore all incoming Ethernet packets
SM 0, 0, 0, 0	Process all incoming Ethernet packets

SP

FUNCTION: Speed

DESCRIPTION:

This command sets the slew speed of any or all axes for independent moves.

Note: Negative values will be interpreted as the absolute value.

ARGUMENTS: SP n,n,n,n or SPA=n where

n is an unsigned even number in the range 0 to 3,000,000 motors. The units are stops per second.

n = ? Returns the speed for the specified axis.

USAGE:

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

DEFAULTS:

Default Value	25000
Default Format	8.0

OPERAND USAGE:

_SPn contains the speed for the specified axis.

RELATED COMMANDS:

AC	Acceleration
DC	Deceleration
PA	Position Absolute
PR	Position Relative
BG	Begin

EXAMPLES:

```
PR 2000,3000,4000,5000  Specify a,b,c,d parameter
SP 5000,6000,7000,8000  Specify a,b,c,d speeds
BG                      Begin motion of all axes
```

Note: *For vector moves, use the vector speed command (VS) to change the speed. SP is not a "mode" of motion like JOG (JG).*

@SQR[n]

FUNCTION: Square Root

DESCRIPTION:

Takes the square root of the given number. If the number is negative, the absolute value is taken first.

ARGUMENTS: @SQR[n] where

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

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

@ABS[n]	Absolute value

EXAMPLES:

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

ST

FUNCTION: Stop

DESCRIPTION:

The ST command stops motion on the specified axis. Motors will come to a decelerated stop. If ST is sent from the host without an axis specification, program execution will stop in addition to motion.

ARGUMENTS: ST nnnnnn where

n is A,B,C,D, or any combination to specify the axis or sequence. If the specific axis or sequence is specified, program execution will not stop.

No argument will stop motion on all axes.

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

RELATED COMMANDS:

BG	Begin Motion
DC	Deceleration rate

EXAMPLES:

```
ST A              Stop A-axis motion
ST S              Stop coordinated sequence
ST ABCD           Stop A,B,C,D motion
ST                Stop ABCD motion
ST SCD            Stop coordinated AB sequence, and C and D motion
```

Hint: *Use the after motion complete command, AM, to wait for motion to be stopped.*

@TAN[n]

FUNCTION: Tangent

DESCRIPTION:

Returns the tangent of the given angle in degrees

ARGUMENTS: @TAN[n] where

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

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

RELATED COMMANDS:

@ASIN[n]	Arc sine
@COS[n]	Cosine
@ATAN[n]	Arc tangent
@ACOS[n]	Arc cosine
@SIN[n]	Sine

EXAMPLES:

```
:MG @TAN[-90]
 -2147483647.0000
:MG @TAN[0]
 0.0000
:MG @TAN[90]
 2147483647.0000
:
```

TB

FUNCTION: Tell Status Byte

DESCRIPTION:

The TB command returns status information from the controller as a decimal number. Each bit of the status byte denotes the following condition when the bit is set (high):

BIT	STATUS
Bit 7	Executing application program
Bit 6	N/A
Bit 5	Contouring
Bit 4	Executing error or limit switch routine
Bit 3	Input interrupt enabled
Bit 2	Executing input interrupt routine
Bit 1	N/A
Bit 0	Echo on

ARGUMENTS:

TB ? returns the status byte

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	3.0

OPERAND USAGE:

_TB Contains the status byte

EXAMPLES:

```
TB?
:65                Data Record Active and Echo is on (26 + 20 = 64 + 1 = 65)
```

TC

FUNCTION: Tell Error Code

DESCRIPTION:

The TC command returns a number between 1 and 255. This number is a code that reflects why a command was not accepted by the controller. This command is useful when the controller halts execution of a program at a command or when the response to a command is a question mark. The TC command will provide the user with a diagnostic tool. After TC has been read, the error code is set to zero.

ARGUMENTS: TC n where

n = 0 Returns code only

n = 1 Returns code and message

n = ? Returns the error code

No argument will provide the error code for all axes

CODE	EXPLANATION	CODE	EXPLANATION
1	Unrecognized command	56	Array index invalid or out of range
2	Command only valid from program	57	Bad function or array
3	Command not valid in program	58	Bad command response (i.e._GNX)
4	Operand error	59	Mismatched parentheses
5	Input buffer full	60	Download error - line too long or too many lines
6	Number out of range	61	Duplicate or bad label
7	Command not valid while running	62	Too many labels
8	Command not valid when not running	63	IF statement without ENDIF
9	Variable error	65	IN command must have a comma
10	Empty program line or undefined label	66	Array space full
11	Invalid label or line number	67	Too many arrays or variables
12	Subroutine more than 16 deep	71	IN only valid in task #0
13	JG only valid when running in jog mode	80	Record mode already running
14	EEPROM check sum error	81	No array or source specified
15	EEPROM write error	82	Undefined Array
16	IP incorrect sign during position move or IP given during forced deceleration	83	Not a valid number
17	ED, and DL not valid while program running	84	Too many elements
18	Command not valid when contouring	90	Only A B C D valid operand
19	Application strand already executing	98	Binary Commands not valid in application program
20	Begin not valid with motor off	99	Bad binary command number
21	Begin not valid while running	100	Not valid when running ECAM
22	Begin not possible due to Limit Switch	101	Improper index into ET (must be 0-256)

24	Begin not valid because no sequence defined	102	No master axis defined for ECAM
25	Variable not given in IN command	103	Master axis modulus greater than 256*EP value
28	S operand not valid	104	Not valid when axis performing ECAM
29	Not valid during coordinated move	105	EB1 command must be given first
30	Sequence segment too short	106	Privilege violation
31	Total move distance in a sequence > 2 billion		
32	More than 31 segments in a sequence		
33	VP or CR commands cannot be mixed with LI commands		
41	Contouring record range error		
42	Contour data being sent too slowly		
46	Gear axis both master and follower	119	Not valid for axis configured as stepper
50	Not enough fields	133	Command not valid when locked
51	Question mark not valid	134	All motors must be in MO for this command
52	Missing " or string too long	135	Motor must be in MO
53	Error in {}	136	Invalid Password
54	Question mark part of string	137	Invalid lock setting
55	Missing [or []	138	Passwords not identical

USAGE: **DEFAULTS:**

While Moving	Yes	Default Value	---
In a Program	Yes	Default Format	3.0
Not in a Program	Yes		
Controller Usage	**ALL CONTROLLERS**		

USAGE:

_TC contains the error code

EXAMPLES:

```
GF32            Bad command
:?              Tell error code
TC
:001            Unrecognized command
```

#TCPERR

FUNCTION: Ethernet communication error automatic subroutine

DESCRIPTION:

The following error (see TC) occurs when a command such as MG “hello” {EA} is sent to a failed Ethernet connection:

123 TCP lost sync or timeout

This error means that the client on handle A did not respond with a TCP acknowledgement (for example because the Ethernet cable was disconnected). Handle A is closed in this case.

#TCPERR allows the application programmer to run code (for example to reestablish the connection) when error 123 occurs.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	No
Controller Usage	**B140**

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**: Use RE to end the routine*

TD

FUNCTION: Tell Stepper Position

DESCRIPTION:

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

ARGUMENTS: TD nnnnnn where

n is A,B,C,D or any combination to specify the axis or axes

No argument will provide the dual encoder position for all axes

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	0
In a Program	Yes	Default Format	Position Format
Not in a Program	Yes		
Controller Usage	**ALL CONTROLLERS**		

OPERAND USAGE:

_TDn contains value of dual encoder register.

RELATED COMMANDS:

DE	Define Encoder Position

EXAMPLES:

```
TD                          Return A,B,C,D Stepper Positions
:200, -10, 0, -110
TDA                         Return the A Stepper Position
:200
DStep=_TDA                  Assign the variable, Step, the value of TDA
```

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

DEFAULTS:

Default Value	-----
Default Format	-----

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
```

TI

FUNCTION: Tell Inputs

DESCRIPTION:

This command returns the state of the inputs including the extended I/O configured as inputs. The value returned by this command is decimal and represents an 8 bit value (decimal value ranges from 0 to 255). Each bit represents one input where the LSB is the lowest input number and the MSB is the highest input bit.

ARGUMENTS: TIn where

n = 0 Return Input Status for Inputs 1 through 8

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	3.0

OPERAND USAGE:

_TIn contains the status byte of the input block specified by 'n'. Note that the operand can be masked to return only specified bit information.

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
:255                All inputs high
```

TIME

FUNCTION: Time Operand (Keyword)

DESCRIPTION:

The TIME operand returns the value of the internal free running, real time clock. The returned value represents the number of servo loop updates and is based on the TM command. The default value for the TM command is 1000. With this update rate, the operand TIME will increase by 1 count every update of approximately 1000usec. Note that a value of 1000 for the update rate (TM command) will actually set an update rate of 976 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 does the other operands.

EXAMPLES:

```
MG TIME             Display the value of the internal clock
```

TM

FUNCTION: Update Time

DESCRIPTION:

The TM command sets the sampling period of the update loop. A zero or negative number turns off the control loop. The units of this command are μsec.

ARGUMENTS: TM n where

With the normal firmware: Using the normal firmware the minimum sample times are the following: 250 microseconds.

n = ? returns the value of the sample time.

USAGE:

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

DEFAULTS:

Default Value	1000
Default Format	1.0

OPERAND USAGE:

_TM contains the value of the sample time.

EXAMPLES:

```
TM -1000            Turn off internal clock
TM 2000             Set sample rate to 2000 msec
TM 1000             Return to default sample rate
```

***NOTE**: TM1000 actually specifies a servo update rate of 976 μs*

TN

FUNCTION: Tangent

DESCRIPTION:

The TN m,n command describes the tangent axis to the coordinated motion path. m is the scale factor in counts/degree of the tangent axis. n is the absolute position of the tangent axis where the tangent axis is aligned with zero degrees in the coordinated motion plane. The tangent axis is specified with the VM n,m,p command where p is the tangent axis. The tangent function is useful for cutting applications where a cutting tool must remain tangent to the part.

ARGUMENTS: TN m,n where

m is the scale factor in counts/degree, in the range between -127 and 127 with a fractional resolution of 0.004

m = ? Returns the first position value for the tangent axis.

When operating with stepper motors, m is the scale factor in steps / degree

n is the absolute position at which the tangent angle is zero, in the range between -8388608to 8388607.

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	PF

OPERAND USAGE:

_TN contains the first position value for the tangent axis. This allows the user to correctly position the tangent axis before the motion begins.

RELATED COMMANDS:

VM	Vector mode
CR	Circle Command

EXAMPLES:

```
VM A,B,C            Specify coordinated mode for A and B-axis; C-axis is
                    tangent to the motion path
TN 100,50           Specify scale factor as 100 counts/degree and  50 counts
                    at which tangent angle is zero
VP 1000,2000        Specify vector position A,B
VE                  End Vector
BGS                 Begin coordinated motion with tangent axis
```

TP

FUNCTION: Tell Position

DESCRIPTION:

This command returns the current encoder position of the motor(s).

ARGUMENTS: TP nnnnnn where

n is A,B,C,D or any combination to specify the axis or axes

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	Position Format

OPERAND USAGE:

_TPx contains the current position value for the specified axis.

RELATED COMMANDS:

PF	Position Formatting

EXAMPLES:

Assume the A-axis is at the position 200 (decimal), the B-axis is at the position -10 (decimal), the C-axis is at position 0, and the D-axis is at -110 (decimal). The returned parameter units are in quadrature counts.

```
TP                                  Return A,B,C,D positions
:200,-10,0,-110
TPA                                 Return the A motor position
:200
TPB                                 Return the B motor position
:-10
PF-6.0                              Change to hex format
TP                                  Return A,B,C,D in hex
:$0000C8,$FFFFF6,$000000,$FFFF93
Position =_TPA                      Assign the variable, Position, the value of
                                    TPA
```

TR

FUNCTION: Trace

DESCRIPTION:

The TR command causes each instruction in a program to be sent out the communications port prior to execution. TR1 enables this function and TR0 disables it. The trace command is useful in debugging programs.

ARGUMENTS: TR n, m where

n = 0 Disables the trace function

n = 1 Enables the trace function

m is an integer between 0 and 15 and designates which threads to trace. A binary weighted bit is set per thread. Thread 0=1, Thread 1=2, Thread 2=4 ... Thread 3 =15. The default is 15 (all threads)

The least significant bit represents thread 0 and the most significant bit represents thread 3. The decimal value can be calculated by the following formula.

$n = n_0 + 2*n_1 + 4*n_2 + 8*n_3$

where n_x represents the thread. To turn tracing on for a thread, substitute a one into that n_x in the formula. If the n_x value is a zero, then tracing will be off for that thread. For example, if threads 2 and 3 are to be traced, TR12 is issued.

USAGE:

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

DEFAULTS:

Default Value	TR0,15
Default Format	--

RELATED COMMANDS:

CW	Set/clear most significant bit

EXAMPLES:

```
:ED                    ;'define a small looping program

0 #L
1 WT1000
2 JP#L
3
<control>q

:XQ                     ;'run the program
:TR1                   ;'turn the trace on
:2 JP#L
0 #L
1 WT1000

2 JP#L
0 #L
1 WT1000
TR0                    ;'turn the trace off
```

TS

FUNCTION: Tell Switches

DESCRIPTION:

TS returns status information of the Home switch, Forward Limit switch Reverse Limit switch, error conditions, motion condition and motor state. The value returned by this command is decimal and represents an 8 bit value (decimal value ranges from 0 to 255). Each bit represents the following status information:

Bit	Status
Bit 7	Axis in motion if high
Bit 6	Axis error exceeds error limit if high
Bit 5	A motor off if high
Bit 4	Undefined
Bit 3	Forward Limit Switch Status inactive if high
Bit 2	Reverse Limit Switch Status inactive if high
Bit 1	Home A Switch Status
Bit 0	Latched

Note: For active high or active low configuration (CN command), the limit switch bits are '1' when the switch is inactive and '0' when active.

ARGUMENTS: TS nnnnnn where

n is A,B,C,D, or any combination to specify the axis or axes

No argument will provide the status for all axes

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	3.0

OPERAND USAGE:

_TS contains the current status of the switches.

EXAMPLES:

```
V1=_TSB          Assigns value of TSB to the variable V1
V1=              Interrogate value of variable V1
:15              Decimal value corresponding to bit pattern 00001111
                 Y axis not in motion (bit 7 - has a value of 0)
                 Y axis error limit not exceeded (bit 6 has a value of 0)
                 Y axis motor is on (bit 5 has a value of 0)
                 Y axis forward limit is inactive (bit 3 has a value of 1)
                 Y axis reverse limit is inactive (bit 2 has a value of 1)
                 Y axis home switch is high (bit 1 has a value of 1)
                 Y axis latch is not armed (bit 0 has a value of 1)
```

TV

FUNCTION: Tell Velocity

DESCRIPTION:

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

ARGUMENTS: TV nnnnnn where

n is A,B,C,D, or any combination to specify the axis or axes

No argument will provide the velocity for all axes.

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	8.0
Command Line	Yes		
Controller Usage	**ALL CONTROLLERS**		

OPERAND USAGE:

_TVn contains the value of the velocity for the specified axis.

EXAMPLES:

```
VELA=_TVA          Assigns value of A-axis velocity to the variable VELA
TVA                Returns the A-axis velocity
:3420
```

Note: *The TV command is computed using a special averaging filter (over approximately 0.25 sec for TM1000). Therefore, TV will return average velocity, not instantaneous velocity.*

UL

FUNCTION: Upload

DESCRIPTION:

The UL command transfers data from the controller to a host computer. Programs are sent without line numbers. The Uploaded program will be followed by a <control>Z as an end of text marker.

ARGUMENTS: None

USAGE:

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

DEFAULTS:

Default Value	0
Default Format	-

OPERAND USAGE:

When used as an operand, _UL gives the number of available variables. The number of available 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 where

s is an unsigned integers in the range 1024 to 1073740800. s represents the vector acceleration for the S coordinate system. The parameter input will be rounded down to the nearest factor of 1024. The units of the parameter is counts per second squared.

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

USAGE:

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

DEFAULTS:

Default Value	256000
Default Format	10.0

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
IT	Smoothing constant - S-curve

EXAMPLES:

```
VA 1024             Set vector acceleration to 1024 counts/sec2
VA ?                Return vector acceleration
:1024
VA 20000            Set vector acceleration
VA ?
:19456              Return vector acceleration
ACCEL=_VA           Assign variable, ACCEL, the value of VA
```

VD

FUNCTION: Vector Deceleration

DESCRIPTION:

This command sets the deceleration rate of the vector in a coordinated motion sequence.

ARGUMENTS: VD s where

s is unsigned integers in the range 1024 to 1073740800. s represents the vector deceleration for the S coordinate system. The parameter input will be rounded down to the nearest factor of 1024. The units of the parameter is counts per second squared.

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

USAGE:

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

DEFAULTS:

Default Value	256000
Default Format	10.0

OPERAND USAGE:

_VDn contains the value of the vector deceleration.

RELATED COMMANDS:

VA	Vector Acceleration
VS	Vector Speed
VP	Vector Position
CR	Circle
VE	Vector End
VM	Vector Mode
BG	Begin Sequence
IT	Smoothing constant - S-curve

EXAMPLES:

```
#VECTOR             ;'Vector Program Label
VMAB                ;'Specify plane of motion
VA1000000           ;'Vector Acceleration
VD 5000000          ;'Vector Deceleration
VS 2000             ;'Vector Speed
VP 10000, 20000     ;'Vector Position
VE                  ;'End Vector
BGS                 ;'Begin Sequence
AMS                 ;'Wait for Vector sequence to complete
EN                  ;'End Program
```

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.

The VE command will apply to the selected coordinate system, S.

ARGUMENTS: VE n

No argument specifies the end of a vector sequence

n = ? Returns the length of the vector in counts.

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	PF

OPERAND USAGE:

_VEn contains the length of the vector in counts for the specified coordinate system, S or T.

RELATED COMMANDS:

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

EXAMPLES:

```
#A                  ;'Program Label
VM AB               ;'Vector move in AB
VP 1000,2000        ;'Linear segment
CR 0,90,180         ;'Arc segment
VP 0,0              ;'Linear segment
VE                  ;'End sequence
BGS                 ;'Begin motion
AMS                 ;'Wait for VE to execute in buffer
EN                  ;'End program
```

VF

FUNCTION: Variable Format

DESCRIPTION:

The VF command formats the number of digits to be displayed when interrogating the controller.

If a number exceeds the format, the number will be displayed as the maximum possible positive or negative number (i.e. 999.99, -999, $8000 or $7FF).

ARGUMENTS: VF m.n where

m and n are unsigned numbers in the range 0<m<10 and 0<n<4.

m represents the number of digits before the decimal point. A negative m specifies hexadecimal format. When in hexadecimal, the string will be preceded by a $ and Hex numbers are displayed as 2's complement with the first bit used to signify the sign.

n represents the number of digits after the decimal point.

m = ? Returns the value of the format for variables and arrays.

USAGE:

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

DEFAULTS:

Default Value	10.4
Default Format	2.1

OPERAND USAGE:

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

RELATED COMMANDS:

PF	Position Format

EXAMPLES:

```
VF 5.3     Sets 5 digits of integers and 3 digits after the decimal point
VF 8.0     Sets 8 digits of integers and no fractions
VF -4.0    Specify hexadecimal format with 4 bytes to the left of the decimal
```

VM

FUNCTION: Coordinated Motion Mode

DESCRIPTION:

The VM command specifies the coordinated motion mode and the plane of motion. This mode may be specified for motion on any set of two axes.

The motion is specified by the instructions VP and CR, which specify linear and circular segments. Up to 32 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.

The VM command will apply to the coordinate system S.

ARGUMENTS: VM nmp where

n and m specify plane of vector motion and can be any two axes. Vector Motion can be specified for one axis by specifying 2nd parameter, m, as N. Specifying one axis is useful for obtaining sinusoidal motion on 1 axis.

p is the tangent axis and can be specified as any axis except the imaginary axis, N.

USAGE:

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

DEFAULTS:

Default Value	A,B
Default Format	8.0

OPERAND USAGE:

_VMn contains instantaneous commanded vector velocity for the specified coordinate system, S

RELATED COMMANDS:

VP	Vector Position
CR	Circle
VE	End Vector Sequence
CS	Clear Sequence
IT	Vector smoothing constant -- S-curve
AV	Trippoint for Vector distance

EXAMPLES:

```
#A                ;'Program Label
VM AB             ;'Vector move in AB
VP 1000,2000      ;'Linear segment
VE                ;'End sequence
BGS               ;'Begin motion
AMS               ;'Wait for vector motion to complete
EN                ;'End program
```

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 elliptical scale factor set using the command ES. For three or more axes linear interpolation, use the LI command. The VP command will apply to the selected coordinate system, S.

ARGUMENTS: VP n,m < o > p where

n and m are signed integers in the range -2147483648 to 2147483647 The length of each segment must be limited to $8 \cdot 10^6$. The values for n and m will specify a coordinate system from the beginning of the sequence.

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

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

USAGE:

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

DEFAULTS:

Default Value	-
Default Format	-

OPERAND USAGE:

_VPn 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:

VM	Vector Mode
VE	Vector End
BG	Begin Sequence
IT	Vector smoothing

EXAMPLES:

```
#A                ;'Program Label
VM AB             ;'Specify motion plane
VP 1000,2000      ;'Specify vector position 1000,2000
VP 2000,4000      ;'Specify vector position 2000,4000
CR 1000,0,360     ;'Specify arc
VE                ;'Vector end
BGS               ;'Begin motion sequence
AMS               ;'Wait for vector motion to complete
EN                ;'End Program
```

Hint: *The first vector in a coordinated motion sequence defines the origin for that sequence. All other vectors in the sequence are defined by their endpoints with respect to the start of the move sequence.*

VR

FUNCTION: Vector Speed Ratio

DESCRIPTION:

The VR sets a ratio to be used as a multiplier of the current vector speed. The vector speed can be set by the command VS or the operators < and > used with CR, VP and LI commands. VR takes effect immediately and will ratio all the following vector speed commands. VR doesn't ratio acceleration or deceleration, but the change in speed is accomplished by accelerating or decelerating at the rate specified by VA and VD.

ARGUMENTS: VR s where

s and t are between 0 and 10 with a resolution of .0001. The value specified by s is the vector ratio to apply to the S coordinate system s = ? Returns the value of the vector speed ratio for the S coordinate plane.

USAGE:

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

DEFAULTS:

Default Value	1
Default Format	2.4

OPERAND USAGE:

_VRn contains the vector speed ratio of the specified coordinate system, S

RELATED COMMANDS:

VS	Vector Speed

EXAMPLES:

```
#A                      ;'Vector Program
 VMAB                   ;'Vector Mode
 VP 1000,2000           ;'Vector Position
 CR 1000,0,360          ;'Specify Arc
 VE                     ;'End Sequence
 VS 2000                ;'Vector Speed
 BGS                    ;'Begin Sequence
 AMS                    ;'After Motion
JP#A                    ;'Repeat Move

#SPEED                  ;'Speed Override
 VR @AN[1]*.1           ;'Read analog input compute ratio
JP#SPEED                ;'Loop

XQ#A,0; XQ#SPEED,1      Execute task 0 and 1 simultaneously
```

Note: *VR is useful for feed rate override, particularly when specifying the speed of individual segments using the operator '<' and '>'.*

VS

FUNCTION: Vector Speed

DESCRIPTION:

The VS command specifies the speed of the vector in a coordinated motion sequence in either the LM or VM modes. VS may be changed during motion.

Vector Speed can be calculated by taking the square root of the sum of the squared values of speed for each axis specified for vector or linear interpolated motion.

ARGUMENTS: VS s where

s are unsigned even numbers in the range 2 to 3,000,000 for stepper motors. The units are counts per second.

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

USAGE:

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

DEFAULTS:

Default Value	25000
Default Format	8.0

OPERAND USAGE:

_VSn contains the vector speed of the specified coordinate system, S

RELATED COMMANDS:

VA	Vector Acceleration
VP	Vector Position
CR	Circle
LM	Linear Interpolation
VM	Vector Mode
BG	Begin Sequence
VE	Vector End

EXAMPLES:

```
VS 2000             Define vector speed of S coordinate system
VS ?                Return vector speed of S coordinate system
:2000
```

Hint: *Vector speed can be attached to individual vector segments. For more information, see description of VP, CR, and LI commands.*

VV

FUNCTION: Vector Speed Variable

DESCRIPTION:

The VV command sets the speed of the vector variable in a coordinated motion sequence in either the LM or VM modes. VV may be changed during motion.

The VV command is used to set the "<" vector speed variable argument for segments that exist in the vector buffer. By defining a vector segment begin speed as a negative 1 (i.e. "<-1"), the controller will utilize the current vector variable speed as the segment is profiled from the buffer.

This is useful when vector segments exist in the buffer that use the "<" and ">" speed indicators for specific segment and corner speed control and the host needs to be able to dynamically change the nominal return operating speed.

The vector variable is supported for VP, CR and LI segments.

ARGUMENTS: VVS=n where,

n specifies the speed as an unsigned even number in the range 2 to 3,000,000 for stepper motors. The units are in counts per second.

VVS=? Returns the value of the vector speed variable for the S coordinate plane.

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes

DEFAULTS:

Default Value	0
Default Format	8.0

OPERAND USAGE:

_VVn contains the vector speed variable of the specified coordinate system (n= S)

RELATED COMMANDS:

VA	Vector Acceleration
VD	Vector Deceleration
VP	Vector Position Segment
CR	Circular Interpolation Segment
LI	Linear Interpolation Segment
VM	Vector Mode
LM	Linear Interpolation Mode

EXAMPLES:

```
VVS= 20000          Define vector speed variable to 20000 for the S coordinate system
VP1000,2000<-1>100  Define vector speed variable for specific segment.
VVS=?               Returns→  20000 <CRLF>: (as set above)
```

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 D to indicate on which handle the command was executed. The command returns RS232 if communicating serially.

ARGUMENTS: None

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

OPERAND USAGE:

_WH contains the numeric representation of the handle in which a command is executed. Handles A through D are indicated by the value 0-3, while a-1 indicates the serial port.

RELATED COMMANDS:

TH	Tell Handle

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. When this command is executed, the controller will wait for the number of ms specified before executing the next command.

ARGUMENTS: WT n where
n is an integer in the range 0 to 2 Billion decimal

USAGE:		**DEFAULTS:**	
While Moving	Yes	Default Value	-
In a Program	Yes	Default Format	-
Command Line	**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
BGA                 ;'Begin the move
AMA                 ;'After the move is over
WT 10000            ;'Wait 10 seconds
SB 0                ;'Turn on relay
EN                  ;'End Program
```

Hint: *To achieve longer wait intervals, just stack multiple WT commands.*

XQ

FUNCTION: Execute Program

DESCRIPTION:

The XQ command begins execution of a program residing in the program memory of the controller. Execution will start at the label or line number specified. Up to 4 programs may be executed with the controller.

ARGUMENTS: XQ #A,n XQm,n where

A is a program name of up to seven characters.

m is a line number

n is an integer representing the thread number for multitasking

n is an integer in the range of 0 to 3.

NOTE: The arguments for the command, XQ, are optional. If no arguments are given, the first program in memory will be executed as thread 0.

USAGE:

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

DEFAULTS:

Default Value of n:	0
Default Format	-

OPERAND USAGE:

_XQn contains the current line number of execution for thread n, and -1 if thread n is not running.

RELATED COMMANDS:

HX	Halt execution

EXAMPLES:

```
XQ #APPLE,0          Start execution at label APPLE, thread zero
XQ #DATA,2           Start execution at label DATA, thread two
XQ 0                 Start execution at line 0
```

Hint: *Don't forget to quit the edit mode first before executing a program! (DOS)*

YA

FUNCTION: Step Drive Resolution

DESCRIPTION:

The YA command specifies the resolution of the step drive, in step counts per full motor step for Stepper Position Maintenance mode.

ARGUMENTS: YA m,m,m,m, or YAn = m where

n is A,B,C,D, or any combination to specify the axis or axes.

m is 0 to 9999 which represents the drive resolution in step counts per full motor step.

USAGE:		**DEFAULTS:**	
While Moving	No	Default Value	2
In a Program	Yes	Default Format	1.4
Command Line	Yes		

OPERAND USAGE:

_YAn contains the resolution for the specified axis.

RELATED COMMANDS:

QS	Error Magnitude
YS	Stepper Position Maintenance Mode Enable, Status
YB	Step Motor Resolution
YC	Encoder Resolution
YR	Error Correction

EXAMPLES:

```
1.  Set the step drive resolution for the SDM-44140 Microstepping Drive:
    YA 64,64,64,64
2.  Query the D axis value:
    MG_YAD
    :64.0000        Response shows D axis step drive resolution
```

Notes:

1. *This value must be the same as the step drive resolution for the axis. The error magnitude (QS) will climb quickly causing a false error state if the assigned value differs from the actual.*

YB

FUNCTION: Step Motor Resolution

DESCRIPTION:

The YB command specifies the resolution of the step motor, in full steps per full revolution, for Stepper Position Maintenance mode.

ARGUMENTS: YB m,m,m,m, or YBn = m where

n is A,B,C,D, or any combination to specify the axis or axes.

m is 0 to 9999 which represents the motor resolution in full steps per revolution.

USAGE:		**DEFAULTS:**	
While Moving	No	Default Value	200
In a Program	Yes	Default Format	1.4
Command Line	Yes		

OPERAND USAGE:

_YBn contains the stepmotor resolution for the specified axis.

RELATED COMMANDS:

QS	Error Magnitude
YS	Stepper Position Maintenance Mode Enable, Status
YA	Step Drive Resolution
YC	Encoder Resolution
YR	Error Correction

EXAMPLES:

1. Set the step motor resolution of the A axis for a 1.8° step motor:

```
YBA=200
```

2. Query the A axis value:

```
YBA=?
:200            Response shows A axis step motor resolution
```

Notes:

1. *This value must be the same as the step motor resolution for that axis. The error magnitude (QS) will climb quickly causing a false error state if the assigned value differs from actual.*

YC

FUNCTION: Encoder Resolution

DESCRIPTION:

The YC command specifies the resolution of the encoder, in counts per revolution, for Stepper Position Maintenance mode.

ARGUMENTS: YC m,m,m,m, or YCn = m where

n is A,B,C,D, or any combination to specify the axis or axes.

m is 0 to 32766 which represents the encoder resolution in counts per revolution.

USAGE:		**DEFAULTS:**	
While Moving	No	Default Value	4000
In a Program	Yes	Default Format	1.4
Command Line	Yes		

OPERAND USAGE:

_YCn contains the encoder resolution for the specified axis.

RELATED COMMANDS:

QS	Error Magnitude
YS	Stepper Position Maintenance Mode Enable, Status
YA	Step Drive Resolution
YB	Step Motor Resolution
YR	Error Correction

EXAMPLES:

1. Set the encoder resolution of the D axis for a 4000 count/rev encoder:

```
YC,,,4000
```

2. Query the D axis value:

```
YCD=?
:4000           Response shows D axis encoder resolution
```

Notes:

1. *This value must be the same as the encoder resolution for that axis. The error magnitude (QS) will climb quickly causing a false error state if the assigned value differs from actual.*

YR

FUNCTION: Error Correction

DESCRIPTION:

The YR command allows the user to correct for position error in Stepper Position Maintenance mode. This correction acts like an IP command, moving the axis or axes the specified quantity of step counts. YR will typically be used in conjunction with QS.

ARGUMENTS: YR m,m,m,m, or YRn = m where

n is A,B,C,D, or any combination to specify the axis or axes.

m is a magnitude in step counts.

USAGE:

While Moving	No
In a Program	Yes
Command Line	Yes

DEFAULTS:

Default Value	0
Default Format	1.4

OPERAND USAGE:

None

RELATED COMMANDS:

QS	Error Magnitude
YA	Step Drive Resolution
YB	Step Motor Resolution
YR	Error Correction
YS	Stepper Position Maintenance Mode Enable, Status

EXAMPLES:

```
Correct for the error:
  YRB=_QSB          The motor moves _QS step counts to correct for the error,
                    is set back to 1
```

Notes:

1. *The YR command issues an increment position move. The magnitude of AC, DC, SP, KS as well as axis non-linearities will affect the accuracy of the correction. It is recommended to use a significant KS value, as well as low AC, DC, and SP for corrections.*

YS

FUNCTION: Stepper Position Maintenance Mode Enable, Status

DESCRIPTION:

The YS command enables and disables the Stepper Position Maintenance Mode function. YS also reacts to excessive position error condition as defined by the QS command.

ARGUMENTS: YS m,m,m,m, or YSn = m where

n is A,B,C,D, or any combination to specify the axis or axes.

m = 0 SPM Mode Disable

m = 1 Enable SPM Mode, Clear trippoint and QS error

m = 2 Error condition occurred

USAGE:

While Moving	Yes
In a Program	Yes
Command Line	Yes

DEFAULTS:

Default Value	0
Default Format	1.4

OPERAND USAGE:

_YSn contains the status of the mode for the specified axis.

RELATED COMMANDS:

QS	Error Magnitude
YA	Step Drive Resolution
YB	Step Motor Resolution
YC	Encoder Resolution
YR	Error Correction

EXAMPLES:

```
1.  Enable the mode:
    YSH=1
2.  Query the value:
    YS*=?
    :0,0,0,0,0,0,0,1    Response shows H axis is enabled
```

Notes:

1. *Ensure the axis is energized and stable before enabling Stepper Position Maintenance mode. Error will result from enabling YS and then energizing the axis.*
2. *Assigning a value of 1 to an axis after encountering an error condition will clear the trippoint and will also clear QS.*
3. *A value of 2 is automatically assigned to YS when the position error exceeds three full motor steps. See the QS command for more details.*

ZA

FUNCTION: User Data Record Variables

DESCRIPTION:

ZA sets the user variables in the data record. The four user variables (one per axis) are automatically sent as part of the status record from the controller to the host computer. These variables provide a method for specific controller information to be passed to the host automatically.

ARGUMENTS: ZA n,n,n,n, or ZAA=n where

n is an integer and can be a number, controller operand, variable, mathematical function, or string. The range for numeric values is 4 bytes of integer (–2,147,483,648 to +2,147,483,647). The maximum number of characters for a string is 4 characters. Strings are identified by quotations.

n = ? returns the current value

USAGE:

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

DEFAULTS:

Default Value	0
Default Format	10.0

OPERAND USAGE:

_ZAn contains the current value

RELATED COMMANDS:

DR	Data record update rate
QZ	Data record format

EXAMPLES:

```
#Thread
  ZAX=MyVar                ;'constantly update ZA
JP#Thread
```

ZS

FUNCTION: Zero Subroutine Stack

DESCRIPTION:

The ZS command is only valid in an application program and is used to avoid returning from an interrupt (either input or error). ZS alone returns the stack to its original condition. ZS1 adjusts the stack to eliminate one return. This turns the jump to subroutine into a jump. Do not use RI (Return from Interrupt) when using ZS. To re-enable interrupts, you must use II command again.

The status of the stack can be interrogated with the operand _ZSn - see operand usage below.

ARGUMENTS: ZS n where

n = 0 Returns stack to original condition

n = 1 Eliminates one return on stack

USAGE:

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

DEFAULTS:

Default Value	0
Default Format	3.0

OPERAND USAGE:

_ZSn contains the stack level for the specified thread where n = 0 to 7.

EXAMPLES:

```
#A                  ;'Main Program
II1                 ;'Input Interrupt on 1
#B;JP #B;EN         ;'Loop
#ININT              ;'Input Interrupt
MG "INTERRUPT"      ;'Print message
S=_ZS               ;'Interrogate stack
S=                  ;'Print stack
ZS                  ;'Zero stack
S=_ZS               ;'Interrogate stack
S=                  ;'Print stack
EN                  ;'End
```

Index