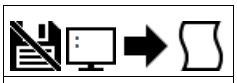
DMC-18x6 Firmware Command Reference

01/16/15

Table of Content

' *Comment*



```
' str
```

Description

The ' allows for a user to insert in a comment on a blank line after a command following a semicolon ";". See examples for valid uses of '.

Arguments

Argument	Value	Description	Notes
str	String	Comments added into program	Comment strings are restricted to the maximum row size for a program. This will vary per controller.

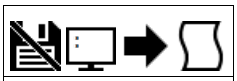
Remarks

- Comments will be downloaded to controller, thus taking up program space.
  - See REM for comments that will not download to controller

Examples

```
'Galil DMC Code Example
'Include an example like this one in the program.
SH AB;'Comments following a command MUST be proceeded by a semi-colon.
KP 10'This is NOT valid use of the '
```

- *Subtraction Operator*



n_0 - n_1

Usage	variable = (value1 - value2)	Performs an operation between two values or evaluated statements

Description

Subtraction operator. Takes as arguments any two values and returns a value equal to the difference of the arguments.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n_0	-2,147,483,648	2,147,483,647	N/A	1/65,536	Value to subtract from	
n_1	-2,147,483,648	2,147,483,647	N/A	1/65,536	Value to subtract	

Remarks

- An operator is not a command and is not valid individually.
- Evaluation occurs left to right. Use parenthesis for operator precedence.
- n_0 and n_1 may also be variables, array elements, operands, or @ functions (e.g. @SIN[]).

Examples

```
'Galil DMC Code Example
:var1 = 10-4
:var2 = var1 - 3
:MG var2 - 1
 2.0000
:
```

```
'Galil DMC Code Example
'It is recommended that parenthesis be used when more than one mathmatical operation is combined in one command.
'Example:
var = ((10*30)+(60/30));'      evaluates as 302
var = 10*30+60/30;'            evalutes as 12
```

Label Designator

#str

Description

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 either user-defined or are automatic subroutines that are run automatically by the firmware when a particular event occurs.

There is a maximum of 510 labels available.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
str	1 char	7 chars	N/A	String	Name of label	

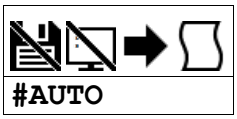
Remarks

- A label can only be defined at the beginning of a new line
- The number of labels available can be queried with MG _DL
- LL returns the current label table in the controller
- Galil recommends that at least the first character be lowercase to differentiate from Automatic subroutines.
- Label must be the first element on a line of code

Examples

```
'Galil DMC Code Example
'A simple example of iteration.  The loop will run 10 times
i= 0;'              Create a counter
#loop;'             Label
 i= i+1;'           Increment counter
JP #loop, i<10;'    Spin in #Loop until i >= 10
EN;'                End the subroutine or thread
```

#AUTO *Subroutine to run automatically upon power up*



Description

Defines the automatic entry point of embedded DMC code. When power is applied to the controller, or after the controller is reset, the program will automatically begin executing at this label. When no host software is used with the controller, #AUTO is required to run an application program on the controller stand-alone.

Arguments

Label must be the first element on a line of code.

Remarks

- Use EN to end the routine
- Thread 0 is used to execute #AUTO on startup
- The BP command must be used to burn a program into EEPROM for the #AUTO to function.

Examples

```
'Galil DMC Code Example
'On startup, this code will create a 50% duty cycle square wave on output 1 with a period of 1 second.
#AUTO;'      Start on powerup
 SB 1;'      Set bit 1
 WT 500;'    Wait 500msec
 CB 1;'      Clear bit 1
 WT 500;'    Wait 500msec
JP #AUTO;'   Jump back to #AUTO
```

#AUTOERR *Bootup Error Automatic Subroutine*

Description

Automatic subroutine that runs code upon power up if the firmware detects errors. If the EEPROM is corrupted, #AUTOERR will run. The EEPROM is considered corrupt if the checksum calculated on the bytes in the EEPROM do not match the checksum written to the EEPROM.

For SSI and BiSS operation, #AUTOERR will also run if the time to acquire serial position data exceeds 90% of the hardware sample loop. This type of error is very rare and should never occur in normal operation.

Arguments

Label must be the first element on a line of code.

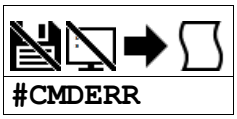
Remarks

- Use EN to end the routine
- The type of checksum error can be queried with MG_RS

Examples

```
'Galil DMC Code Example
'Code detects a checksum error and notifies the user
#AUTOERR
 MG "EEPROM ERROR ",_RS
EN
```

#CMDERR *Command error automatic subroutine*



Description

Automatic subroutine that runs code when a DMC code error occurs. Without #CMDERR defined, if an error (see TC command) occurs in an application program running on the Galil controller, the program (and all threads) will stop.

Arguments

Label must be the first element on a line of code.

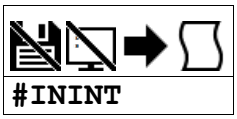
Remarks

- Use EN to end the routine
- #CMDERR will only run from errors generated within embedded DMC code, not from the terminal or host
- In a single threaded application (Thread 0 only), the EN command in the #CMDERR routine will restart thread 0 where it left off.
- In a multi-threaded application, the thread that has an error will be halted when a command error occurs. Thread 0 will be interrupted to run the #CMDERR routine but other threads will continue to run.
  - In order to restart the thread that encountered the error, see the example in Chapter 7 of the User Manual and the _ED operand.
- Thread 0 does not need to be running in order for the #CMDERR routine to execute.

Examples

```
'Galil DMC Code Example
'This code will put the motion controller in Position Tracking mode.
'Variable "target" is updated from the terminal or from a host program
'to specify a new target.  #CMDERR is used to detect a bad target value.
#start
DPA= 0;'      Define current position as zero
PTA= 1;'      Turn on position tracking
target= 0;'   Initialize target variable
#track;'     Start tracking
PAA= target;' Track to current value of target
WT 500;'      Wait 500 ms
JP #track;'   Continue to track
'
'
#CMDERR;' runs if an error occurs
JP #done,_TC<>6 ;'check that an out of range occured (See TC)
MG "Value ",target," is out of range for Position Tracking"
target= _PAA ;' reset target
#done
EN 1 ;'return to tracking logic
```

#ININT *Input interrupt automatic subroutine*



Description

Automatic subroutine that runs upon a state transition of digital inputs. #ININT is configured with II. #ININT runs in thread 0.

Arguments

Label must be the first element on a line of code.

Remarks

- Use the II command to enable the routine.
- Use RI to exit the routine.
- To make an unconditional jump from #ININT, there are two methods for re-enabling the interrupt capability
  - Issue a ZS and then re-issue the command II before the JP
  - or, use a "null" routine. The "null" routine allows for the execution of the RI command before the unconditional jump. For more information see Application Note #2418, http://www.galilmc.com/support/appnotes/optima/note2418.pdf

Examples

```
'Galil DMC Code Example
II 1;                 'arm digital input 1
EN;                  'End thread zero
'
#ININT;               'Automatic sub.  Runs on input event
 MG "Inputs:",_TI0;   'Display status of inputs 1-8
 WT 100;              'Debounce input
RI;                  'Return from interrupt
```

#ININT applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

#LIMSWI *Limit switch automatic subroutine*

Description

Automatic sub for running user-defined code on a limit switch event. A limit switch event requires the following conditions.

1. Motion profiling in the direction of the given limit. I.E. RPm increasing for forward switch, RPm decreasing for reverse switch.
2. Limit switch toggles active. See CN for inverting the active sense of the limit swithes.

Without #LIMSWI defined, the controller will issue ST on the axis when its limit switch is tripped during motion in the direction of the switch. With #LIMSWI defined, code is executed in addition to the stop.

In lieu of a controlled stop, the motor can turn off and coast stop in the event of a limit switch event. See OE for this feature.

Arguments

Label must be the first element on a line of code.

Remarks

- Use RE to terminate the subroutine
- See _LF and _LR for switch state operands

- #LIMSWI runs on thread 0. Code does not need to be running in thread 0 for #LIMSWI to be enabled.
- LD can be used to disable the limit operation
- SD can be used to set the deceleration speed on the limit.

Examples

```
'Galil DMC Code Example
#main         ;'print a message every second
  MG "Main"
  WT 1000
JP #main
EN
'
#LIMSWI ;'runs when a limit switch is tripped
MG "Limit switch:"{N}
IF ((_LFA = 0) | (_LRA = 0))
 MG "Axis A"
ENDIF
IF ((_LFB = 0) | (_LRB = 0))
 MG "Axis B"
ENDIF
RE 1;' RE used to exit the #LIMSWI sub
```

#LIMSWI applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

#MCTIME *MC command timeout automatic subroutine*

Description

Automatic sub used to run user-code if a Motion Complete (MC) trippoint times out. If the motor position does not reach or pass the target within the specified timeout (TW), #MCTIME wil run if present.

MC uses position from TP for servos, or TD for steppers.

Arguments

Label must be the first element on a line of code.

Remarks

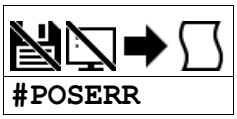
- Use EN to terminate the subroutine

Examples

```
'Galil DMC Code Example
#begin;'                Begin main program
 TWA= 1000;'            Set the time out to 1000 ms
 PRA= 10000;'           Position relative
 BG A;'                 Begin motion
 MC A;'                 Motion Complete trip point
EN;'                    End main program
'
#MCTIME;'               Motion Complete Subroutine
 MG "A fell short";'    Send out a message
EN 1;'                  End subroutine
```

#MCTIME applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

#POSERR *Position error automatic subroutine*



Description

Automatic subroutine that runs user code when a position error event occurs. The factory default behavior of the Galil controller upon a position error (_TEn > _ERn) is to drive the error signal low only, turning on the red error LED. If OE is set to 1, the motor whose position error (TE) equals or exceeds its threshold (ER) will be turned off (MO). #POSERR is used to run code upon a position error, for example to notify a host computer.

Arguments

Label must be the first element on a line of code.

Remarks

- Use RE to end the routine.

- #POSERR runs on thread 0. Code does not need to be running in thread 0 for #POSERR to be enabled.
- #POSERR will also run when OE1 is set for an axes and that axis is also setup for encoder failure detection (see OA, OT, OV commands).

Examples

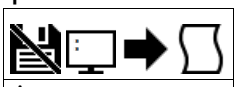
```
'Galil DMC Code Example
#main;'       main program
'
JP #main

REM simple example of #POSERR
#POSERR
 MG "#POSERR"
RE
```

```
'Galil DMC Code Example
REM example of #POSERR that checks for position error on each axis
#POSERR
 ~a= 0;'       axis designator
 IF ((_TE~a>_ER~a)&(_OE~a))
  MG "Position Error occured on ",~a{F1.0}," axis"
 ENDIF
 ~a= ~a+1
JP #POSERR,~a<_BV;' loop until axes have been checked
 AI 1;'       wait until input 1 goes high (ex. safety switch)
 SH
RE 1;'         retrurn to main program
```

```
'Galil DMC Code Example
REM #POSERR example for checking to see if encoder failure occured
REM The stop code will only update of the profilier is running at the time
REM the encoder failure is detected.
#POSERR
~a= 0
#loop
IF _MO~a=1
 IF ((_TE~a<_ER~a)&(_OE~a)&(_OA~a))
  MG "possible encoder failure on ",~a{Z1.0}," axis"
 ENDIF
ENDIF
~a= ~a+1
JP #loop,~a<_BV
AI 1;'              wait for input 1 to go high
SH ;'               enable all axes
RE
```

#POSERR applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

$ *Hexadecimal*



$n

Description

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

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	$80000000.0000	$7FFFFFFF.FFFF	N/A	$0.0001	Value of hexadecimal number	32 bits of integer and 16 bits of fraction in total

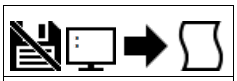
Remarks

- None

Examples

```
'Galil DMC Code Example
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
```

% *Modulo Operator*



```
n % n
```

Usage	variable = (value1 % value2)	Performs an operation between two values or evaluated statements

Description

The % symbol is the modulo operator. It takes as arguments any two values, variables, array elements, operands, or At functions (@SIN[]) and returns a value equal to the modulo of the arguments.

Mathmatical operations are calculated left to right rather than multiplication and division calculations performed prior to addition and subraction.
Example:
1+2*3 = 9, not 7

It is recommended that parenthesis be used when more than one mathmatical operation is combined in one command.
Example:
var = ((10*30)+(60/30));' evaluates as 302
var = 10*30+60/30;' evalutes as 12

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	-2,147,483,648	-2,147,483,647.9999	N/A	1/65,536	Value to use in modulo operation	

Remarks

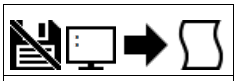
- This is a binary operator (takes two arguments and returns one value). The result of this operation is a value, which is not valid on its own. It must be coupled with a command. See examples below.
- Mathmatical operations are calculated left to right rather than multiplication and division calculations performed prior to addition and subraction.
  - Example: 1+2*3 = 9, not 7
- It is recommended that parenthesis be used when more than one mathmatical operation is combined in one command.
  - Example: var = ((10*30)+(60/30));' evaluates as 302
  - var = 10*30+60/30;' evalutes as 12

Examples

```
'Galil DMC Code Example
'Determine the day of week in n days
DM name[7];'Strings for day of week
name[0]= "SUN"
name[1]= "MON"
name[2]= "TUE"
name[3]= "WED"
name[4]= "THU"
name[5]= "FRI"
name[6]= "SAT"
today= 2;'Tuesday
days= 123;'Days from now
dow= ((days + today)%7);'calculate future day of week
MG "The day of week in ",days{Z10.0}," days will be ", name[dow]{S3.0}
EN

REM Code Returns: The day of week in  123 days will be SAT
```

% applies to DMC40x0,DMC42x0,DMC41x3,RIO,DMC18x6,DMC30010,DMC500x0

& *Bitwise AND Operator*



n & n

Usage	variable = (value1 & value2)	Performs an operation between two values or evaluated statements

Description

The & symbol is the bitwise AND operator used with IF, JP, and JS decisions, and also to perform bitwise ANDING of values.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	-2,147,483,648	2,147,483,647	N/A	1/65,536	Value to use with AND operator	

Remarks

- The result of this operation is a value, which is not valid on its own. It must be coupled with a command. See examples below.
- For IF, JP, and JS, the values used for n are typically the results of logical expressions such as (x > 2) & (y=8)

Examples

```
'Galil DMC Code Example
'Bitwise use
:var1= $F;'00001111
:var2= $F0;'1111000
:MG (var1 & var2)
0.0000
:MG var1
15.0000
:MG var2
240.0000
:
```

```
'Galil DMC Code Example
'Conditional Use
var1= $F;'00001111
var2= $F0;'1111000
IF (var1 = $F) & (var2 = $F1)
 MG "True"
ELSE
 MG "False"
ENDIF
EN

REM Returned: False
```

& *JS subroutine pass variable by reference*

```
JS#str0(&str1,&str1,&str1,&str1,&str1,&str1,&str1,&str1)
```

Description

The & symbol is used to pass a variable by reference on the subroutine stack. When passed by reference, a change to the local-scope variable changes the global value.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
str0	1 char	7 chars	N/A	String	Name of label to use for subroutine call	
str1	1 char	8 chars	N/A	String	Name of variable to pass by reference to the subroutine	

Remarks

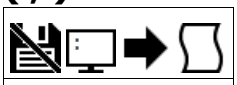
- Variables sent to a subroutine must be global variables that are already dimensioned.
- **Do not dimension any variables in a subroutine when passing variables by reference. This can break the variable pointer.**
- If the global variable should not get changed, omit the & symbol to send a local copy of the variable to the stack.

Examples

```
'Galil DMC Code Example
REM Pass By Reference Example:

#main
value= 5;'          a value to be passed by reference
global= 8;'         a global variable
JS #sum(&value,1,2,3,4,5,6,7);' note first arg passed by reference
MG value;'         message out value after subroutine.
MG _JS;'           message out returned value
EN
'
#sum;'             (* ^a,^b,^c,^d,^e,^f,^g)
^a= ^b+^c+^d+^e+^f+^g+^h+global
EN ,,^a
'notes-
'do not use spaces when working with ^
'If using global variables, they MUST be created before the subroutine is run

'From Terminal
:
Executed program from program2.dmc
36.0000
36.0000
```

& applies to DMC40x0,DMC42x0,DMC41x3,DMC18x6,DMC30010,DMC500x0

(,) *Parentheses (order of operations)*



(n)

Description

The parentheses denote the order of math and logical operations.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	-2,147,483,648	2,147,483,647.9999	N/A	1/65,536	Math or logical expression for evaluation	

Remarks

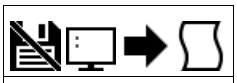
- Note that the controller evaluates expressions from left to right, and does **not** follow academic algebraic standards (e.g. multiplication and division first, followed by addition or subtraction)
- It is required to use parentheticals to ensure intended mathematical precedence

Examples

```
'Galil DMC Code Example
:MG 1+2*3
 9.0000
:MG 1+(2*3)
 7.0000
```

```
'Galil DMC Code Example
:var1= $1F
:var2= $F
:MG var1&var2/$10
0.9375       ($0.F000)
:MG var1&(var2/$10)
0.0000       ($0.0000)
```

* *Multiplication Operator*



n * n

Usage	variable = (value1 * value2)	Performs an operation between two values or evaluated statements

Description

The * symbol is the multiplication operator. It takes as arguments any two values, variables, array elements, operands, or At functions (@SIN[]) and returns a value equal to the product of the arguments.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	-2,147,483,648	-2,147,483,647	N/A	1/65,536	Value to use in multiplication operation	

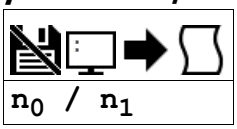
Remarks

- This is a binary operator (takes two arguments and returns one value). The result of this operation is a value, which is not valid on its own. It must be coupled with a command. See examples below.
- Mathmatical operations are calculated left to right rather than multiplication and division calculations performed prior to addition and subraction.
  - Example: 1+2*3 = 9;' not 7
- It is recommended that parenthesis be used when more than one mathmatical operation is combined in one command.
  - Example: var = ((10*30)+(60/30));' evaluates as 302
  - var = 10*30+60/30;' evalutes as 12

Examples

```
'Galil DMC Code Example
:var1 = (2 + 3) * 2
:var2 = var1 * 10
:MG var2 * 0.5
 50.0000
:
```

/ *Division Operator*



Usage	variable = (value1 / value2)	Performs an operation between two values or evaluated statements

Description

The / symbol is the division operator. It takes as arguments any two values, variables, array elements, operands, or At functions (@SIN[]) and returns a value equal to the quotient of the arguments.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n0	-2,147,483,648	2,147,483,647	N/A	1/65,536	Numerator of divide operation	
n1	-2,147,483,648	2,147,483,647	N/A	1/65,536	Denominator of divide operation	

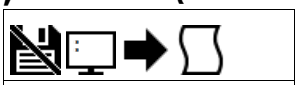
Remarks

- This is a binary operator (takes two arguments and returns one value). The result of this operation is a value, which is not valid on its own. It must be coupled with a command. See examples below.
- Mathmatical operations are calculated left to right rather than multiplication and division calculations performed prior to addition and subraction.
  - Example: 1+2*3 = 9;' not 7
- It is recommended that parenthesis be used when more than one mathmatical operation is combined in one command.
  - Example: var = ((10*30)+(60/30));' evaluates as 302
  - var = 10*30+60/30;' evalutes as 12

Examples

```
'Galil DMC Code Example
:var1 = 100/10
:var2 = var1/2
:MG var2 + 1
 6.0000
:
```

; *Semicolon (Command Delimiter)*



```
arg;arg;arg;arg
```

Description

The semicolon operator allows multiple Galil commands to exist on a single line.

Arguments

arg represents any valid Galil command

Remarks

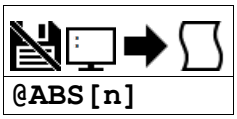
- The semicolon operator is used for the following reasons:
  1. To put comments on the same line as the command (STX ;'stop)
  2. To compress DMC programs to fit within the program line limit (Note: use a compression utility to do this. Do not program this way because it is hard to read.)
  3. To give higher priority to a thread. All commands on a line are executed before the thread scheduler switches to the next thread.

Examples

```
'Galil DMC Code Example
SB 1;WT 500;CB 1;' multiple commands separated by semicolons with a comment
```

```
'Galil DMC Code Example
#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
```

@ABS *Absolute value*



Usage	variable = @ABS[value]	Performs a function on a value or evaluated statement and returns a value

Description

The @ABS[] operation takes the absolute value of the given number. Returns the value if positive, and returns -1 times the value if negative.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	-2,147,483,648	2,147,483,647	N/A	1/65,535	Number to display as absolute value	

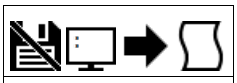
Remarks

- @ABS[] is an operand, not a command. It can only be used as an argument to other commands and operators

Examples

```
'Galil DMC Code Example
:MG @ABS[-2147483647]
 2147483647.0000
```

@ACOS *Inverse cosine*



@ACOS [n]

Usage	variable = @ACOS[value]	Performs a function on a value or evaluated statement and returns a value

Description

The @ACOS operator returns in degrees the arc cosine of the given number.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	-1	1	N/A	1/65,536	Value used for arc cosine operation	

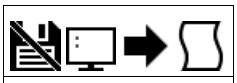
Remarks

- @ACOS[] is an operand, not a command. It can only be used as an argument to other commands and operators
- @ACOS[] is also referred to as the inverse cosine function

Examples

```
'Galil DMC Code Example
:MG @ACOS[-1]
 180.0000
:MG @ACOS[0]
 90.0000
:MG @ACOS[1]
 0.0001
```

@AN *Analog Input Query*



@AN[n]

Usage	variable = @AN[value]	Performs a function on a value or evaluated statement and returns a value

Description

The @AN[] operator returns the value of the given analog input in volts.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	1	8	N/A	1	Analog input to query	

Remarks

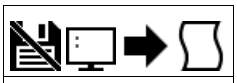
- @AN[] is an operand, not a command. It can only be used as an argument to other commands and operators

Examples

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

@AN applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,RIO,DMC18x6,DMC30010,DMC500x0

@ASIN *Inverse sine*



@ASIN[n]

Usage	variable = @ASIN[value]	Performs a function on a value or evaluated statement and returns a value

Description

The @ASIN operator returns in degrees the arc sine of the given number.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	-1	1	N/A	1/65,536	Value used for arc sine operation	

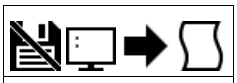
Remarks

- @ASIN[] is an operand, not a command. It can only be used as an argument to other commands and operators
- @ASIN[] is also referred to as the inverse sine function

Examples

```
'Galil DMC Code Example
:MG @ASIN[-1]
 -90.0000
:MG @ASIN[0]
 0.0000
:MG @ASIN[1]
 90.0000
```

@ATAN *Inverse tangent*



@ATAN[n]

Usage	variable = @ATAN[value]	Performs a function on a value or evaluated statement and returns a value

Description

The @ATAN operator returns in degrees the arc tangent of the given number.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	-2,147,483,638	2,147,483,647	N/A	1/65,536	Value used for arc tangent operation	

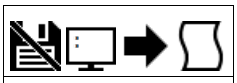
Remarks

- @ATAN[] is an operand, not a command. It can only be used as an argument to other commands and operators
- @ATAN[] is also referred to as the inverse tangent function

Examples

```
'Galil DMC Code Example
:MG @ATAN[-10]
 -84.2894
:MG @ATAN[0]
 0.0000
:MG @ATAN[10]
 84.2894
```

@COM *Bitwise complement*



@COM[n]

Usage	variable = @COM[value]	Performs a function on a value or evaluated statement and returns a value

Description

The @COM[] operation performs the bitwise complement (NOT) operation to the given number.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	-2,147,483,648	2,147,483,647	N/A	1	Value to perform bitwise complement operation.	Integer interpreted as a 32-bit field

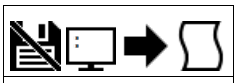
Remarks

- @COM[] is an operand, not a command. It can only be used as an argument to other commands and operators

Examples

```
'Galil DMC Code Example
:MG {$8.0} @COM[0]
$FFFFFFFF
:MG {$8.0} @COM[$FFFFFFFF]
$00000000
```

@COS *Cosine*



@COS[n]

Usage	variable = @COS[value]	Performs a function on a value or evaluated statement and returns a value

Description

The @COS[] operation returns the cosine of the given angle in degrees

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	-32,768	32,767	N/A	1/65,536	Value in degrees to use for cosine operation	

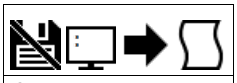
Remarks

- @COS[] is an operand, not a command. It can only be used as an argument to other commands and operators

Examples

```
'Galil DMC Code Example
: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
```

@FLOT *Convert Galil 4.2 to Floating Point*



@FLOT[n]

Usage	variable = @FLOT[value]	Performs a function on a value or evaluated statement and returns a value

Description

The @FLOT operation returns the 32bit floating representation of a number

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	-2,147,483,648	2,147,483,647	N/A	1/65,536	Value to use for floating point conversion	

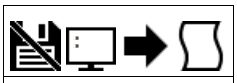
Remarks

- @FLOT[] is an operand, not a command. It can only be used as an argument to other commands and operators
- A useful utility for determining the 32 bit floating point value for a given fractional number can be found here: http://babbage.cs.qc.cuny.edu/IEEE-754/index.xhtml

Examples

```
'Galil DMC Code Example
:MG @FLOT[2.5] {$8.0}
$40200000
:MG @REAL[$40200000]
 2.5000
:
```

@FRAC *Fractional part*



@FRAC[n]

Usage	variable = @FRAC[value]	Performs a function on a value or evaluated statement and returns a value

Description

The @FRAC operation returns the fractional part of the given number

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	-2,147,483,648	2,147,483,647	N/A	1/65,536	Value to use in fractional operation	

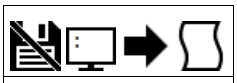
Remarks

- The sign of the number input to the operation will be maintained in the fractional output.
- @FRAC[] is an operand, not a command. It can only be used as an argument to other commands and operators

Examples

```
'Galil DMC Code Example
:MG @FRAC[1.2]
 0.2000
:MG @FRAC[-2.4]
 -0.4000
```

@IN *Read digital input*



@IN[n]

Usage	variable = @IN[value]	Performs a function on a value or evaluated statement and returns a value

Description

The @IN operand returns the value of the given digital input (either 0 or 1)

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	1	16	N/A	1	General input to query	Inputs 9-16 only valid for 5-8 axis controller
	17	80	N/A	1	Extended input to query	DB-14064 required. See Remarks
	81	96	N/A	1	Aux encoder input to query	Used when repurposing aux encoder inputs as digital inputs

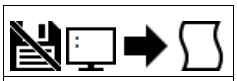
Remarks

- @IN[] is an operand, not a command. It can only be used as an argument to other commands and operators
- Extended IO must be configured as inputs by the CO command for valid results

Examples

```
'Galil DMC Code Example
:MG @IN[1]
1.0000
:x = @IN[1]
:x = ?;'  print digital input 1
1.000
```

@INT *Integer part*



@INT[n]

Usage	variable = @INT[value]	Performs a function on a value or evaluated statement and returns a value

Description

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

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	-2,147,483,648	2,147,483,647	N/A	1/65,536	Value to use in integer operation	

Remarks

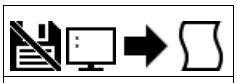
- @INT[] is an operand, not a command. It can only be used as an argument to other commands and operators

Examples

```
'Galil DMC Code Example
:MG @INT[1.2]
 1.0000
:MG @INT[-2.4]
 -2.0000
```

```
'Galil DMC Code Example
#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
```

@OUT *Read digital output*



@OUT[n]

Usage	variable = @OUT[value]	Performs a function on a value or evaluated statement and returns a value

Description

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

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	1	16	N/A	1	General output to query	Outputs 9-16 only valid for 5-8 axis controller
	17	80	N/A	1	Extended output to query	DB-14064 required. See Remarks

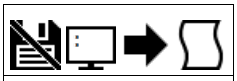
Remarks

- Extended IO must be configured as outputs with the CO command for valid response
- @OUT[] is an operand, not a command. It can only be used as an argument to other commands and operators

Examples

```
'Galil DMC Code Example
:MG @OUT[1];'     print state of digital output 1
1.0000
:x = @OUT[1];'    assign state of digital output 1 to a variable
```

@REAL *Convert Floating Point to Galil 4.2*



@REAL[n]

Usage	variable = @REAL[value]	Performs a function on a value or evaluated statement and returns a value

Description

The @REAL operation returns the Galil 4.2 equivalent of a 32 bit floating point number

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	-2,147,483,648	2,147,483,647	N/A	1	32 bit floating point number to convert to Galil 4.2 integer	

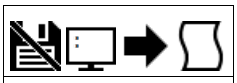
Remarks

- @REAL[] is an operand, not a command. It can only be used as an argument to other commands and operators
- A useful utility for determining the 32 bit floating point value for a given fractional number can be found here: http://babbage.cs.qc.cuny.edu/IEEE-754/index.xhtml

Examples

```
'Galil DMC Code Example
:MG @FLOT[2.5] {$8.0}
$40200000
:MG @REAL[$40200000]
 2.5000
:
```

@RND *Round*



@RND[n]

Usage	variable = @RND[value]	Performs a function on a value or evaluated statement and returns a value

Description

The @RND operation rounds the given number to the nearest integer.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	-2,147,483,648	2,147,483,647	N/A	1/65,536	Value to use in rounding operation	

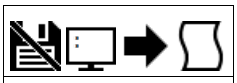
Remarks

- @FRAC[] is an operand, not a command. It can only be used as an argument to other commands and operators
- The sign of the number input to the operation will be maintained in the rounded output.

Examples

```
'Galil DMC Code Example
:MG @RND[1.2]
 1.0000
:MG @RND[1.6]
 2.0000
:MG @RND[-1.2]
 -1.0000
:MG @RND[5.7]
 6.0000
:MG @RND[-5.7]
 -6.0000
:MG @RND[5.5]
 6.0000
:MG @RND[-5.5]
 -5.0000
```

@SIN *Sine*



@SIN[n]

Usage	variable = @SIN[value]	Performs a function on a value or evaluated statement and returns a value

Description

The @SIN[] operation returns the sine of the given angle in degrees

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	-32,768	32,767	N/A	1/65,536	Value in degrees to use for sine operation	

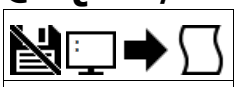
Remarks

- @SIN[] is an operand, not a command. It can only be used as an argument to other commands and operators

Examples

```
'Galil DMC Code Example
: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
```

@SQR *Square Root*



@SQR[n]

Usage	variable = @SQR[value]	Performs a function on a value or evaluated statement and returns a value

Description

The @SQR operation takes the square root of the given number.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	-2,147,483,648	2,147,483,647	N/A	1/65,536	Value to use in square root operation	If n < 0, the absolute value is taken first.

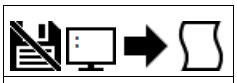
Remarks

- @SQR[] is an operand, not a command. It can only be used as an argument to other commands and operators

Examples

```
'Galil DMC Code Example
:MG @SQR[2]
 1.4142
:MG @SQR[-2]
 1.4142
```

@TAN *Tangent*



@TAN[n]

Usage	variable = @TAN[value]	Performs a function on a value or evaluated statement and returns a value

Description

The @TAN[] operation returns the tangent of the given angle in degrees.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	-32,768	32,767	N/A	1/65,536	Value in degrees to use for tangent operation	

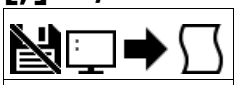
Remarks

- @TAN[] is an operand, not a command. It can only be used as an argument to other commands and operators

Examples

```
'Galil DMC Code Example
:MG @TAN[23]
 0.4245
:
```

[,] *Square Brackets (Array Index Operator)*



`str[n]`

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

Argument	Min	Max	Default	Resolution	Description	Notes
str	1 char	8 chars	N/A	String	Name of array to access	Must be a valid dimensioned array name.
n	-1	15,999	N/A	1	Element of array to query	n = -1 returns the array length

Argument	Min	Max	Default	Resolution	Description	Notes
str	1 char	8 chars	N/A	String	Name of array to access	Must be a valid dimensioned array name.
n	0	399	N/A	1	Element of array to query	For RIO-47xx0
	0	999	N/A	1	Element of array to query	For RIO-47xx2 and RIO-473xx

Remarks

- If the array will be passed by reference on the subroutine stack (JS), the array name MUST be 6 characters or less.

Examples

```
'Galil DMC Code Example
DM a[50]          ;'define a 50 element array
a[0] = 3          ;'set first element to 3
MG a[0]           ;'print element 0
```

```
'Galil DMC Code Example
#array
 DM a[5];'                  define a 5 element array
 a[0] = 3;'                 set first element to 3
 MG "A[0]=",a[0];'          print element 0
 len= a[-1];'               len now contains the length of A[]
 QU a[],0,len-1,1;MG "";'    print entire array
 MG "A[] length=",len;'     display variable len
EN


'Example Output from terminal
:XQ #array
:
A[0]= 3
3, 4320, 216666, 217522, 607950
A[] length= 5

:
```

^ *JS subroutine stack variable*

^s

Description

The ^ character provides local subroutine access for variables passed on the subroutine stack. Passing values on the stack is advanced DMC programming, and is recommended for experienced DMC programmers familiar with the concept of passing arguments by value and by reference.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
s	a	h	N/A	N/A	Stack variable name	a,b,c,d,e,f,g,h supported

Remarks

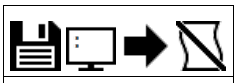
- See the JS command for a full explanation of passing stack variables.
- Passing parameters has no type checking, so it is important to exercise good programming style when passing parameters. See examples below for recommended syntax.
- Do not use spaces in expressions containing ^.
- Global variables MUST be assigned prior to any use in subroutines where variables are passed by reference.
- Arrays passed on the stack must have names no longer than 6 chars.
- Stack zero has no local-scope variables. Accessing these variables from stack zero writes to stack 1's variable table.

Examples

```
'Galil DMC Code Example
#add
JS #sum(1,2,3,4,5,6,7,8) ;' call subroutine, pass values
MG _JS ;' print return value
EN
'
#sum ;NO(^a,^b,^c,^d,^e,^f,^g,^h) Sums the values ^a to ^h and returns the result
EN ,,(^a+^b+^c+^d+^e+^f+^g+^h) ;' return sum

'Output from the previous program
:XQ #add
36.0000
```

^ applies to DMC40x0,DMC42x0,DMC41x3,DMC18x6,DMC30010,DMC500x0

^L^K *Lock program*



^L^K str,n

Usage	^L^K n ...	Arguments specified with an implicit, comma-separated order

Description

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. Once the program is unlocked, it will remain accessible until a lock command or a reset (with the locked condition burned in) occurs.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
str	0 char	8 chars	""	String	Controller password string	Password assigned with the PW command.
n	0	1	0	1	Set lock/unlock state for controller	n = 1 locks the application program. n = 0 unlocks the application program.

Remarks

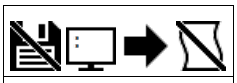
- The PW command can only be set while the application program is unlocked.
- ^L^K ? will return a 0 if the controller is not locked, and a 1 if it is locked.

Examples

```
'Galil DMC Code Example
:PW test,test;'              Set password to "test"
:^L^K test,1;'               Lock the program
:LS;'                        Attempt to list the program
?
:TC 1
106 Privilege violation
:
```

^L^K applies to DMC40x0,DMC42x0,DMC41x3,RIO,DMC18x6,DMC30010,DMC500x0

^R^S *Master Reset*



^R^S

Usage	^R^S	Command takes no arguments

Description

The Master Reset command resets the controller to factory default settings and erases EEPROM. A master reset can also be performed by installing a jumper at the location labeled MRST and resetting the board (power cycle or pressing the reset button). Remove the jumper after this procedure.

Arguments

^R^S has no parameters

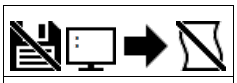
Remarks

- None

Examples

```
'Galil DMC Code Example
REM Example burns-in a non-default value for KP, does a standard reset with
REM the RS command, then performs a master reset with ^R^S.

:KP ?
 6.00
:KP 10
:BN
:RS

:KP ?
 10.00
:^R^S

:KP ?
 6.00
:
```

^R^V *Revision Information*



^R^V

Usage	^R^V	Command takes no arguments

Description

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

Arguments

^R^V has no arguments

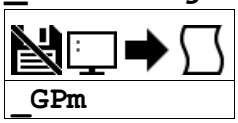
Remarks

- Do not use ^ symbols to send ^R^V command. ^ symbols denote using the control (Ctrl) key when pressing the characters.

Examples

```
'Galil DMC Code Example
:^R^V
DMC1846 Rev 1.0c
:
```

_GP *Gearing Phase Differential Operand*



Usage	variable= _GP	Holds a value
Operands	_GPm	Operand has special meaning, see Remarks

Description

The _GP operand contains the value of the "phase differential" 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.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis of interest	

Remarks

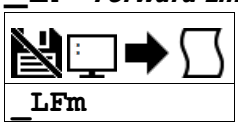
- An operand is not valid individually. Instead, _GP would be used in an expression. See example below.
- Phase Differential is a term that is used to describe the lead or lag between the master axis and the slave axis due to gradual gear shift
  - Pd= GR * Cm - Cs where
    - Pd is the phase differential
    - GR is the gear ratio
    - Cm is the number of encoder counts the master axis moved
    - Cs is the number of encoder counts the slave moved.

Examples

```
'Galil DMC Code Example
GA DA;'   Sets the A axis auxs encoder as the gearing master for the A axis.
GD 1000;'  Set the distance that the master will travel to 1000
'         counts before the gearing is fully engaged for the A
'         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.
GR 1;'     Engage gearing between the master and slave
p1= _TDA;' Sets the current A axis position to variable P1.  This
'         variable is used in the next command
#wait
'         Wait for the aux encoder 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.
JP #wait,  (_TDA < (p1+1000))
IP _GPA;' Increment the difference to bring the master/slave in
'         position sync from the point that the GR1 command was
'         issued.
EN;'      End Program
```

_GP applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

_LF *Forward Limit Switch Operand*



Usage	variable= _LF	Holds a value
Operands	_LFm	Operand has special meaning, see Remarks

Description

The _LF operand contains the state of the forward limit.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis of forward limit switch	

Remarks

- _LF is an operand only with the following output:
  - _LFm = 1 when the limit switch state will allow motion in the positive direction.
  - _LFm = 0 when the limit switch state will not allow motion in the positive direction.
- This operand is not a direct readout of the digital input and is affected by the command CN.
- See Connecting Hardware in User Manual for active/inactive state

Values of _LF

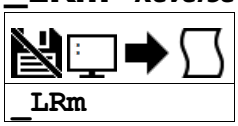
Digital Input activation	_LF value for CN-1	_LF value for CN1
On. Grounded for TTL, or sufficient activation current flowing for optos.	0 (forward motion prohibited)	1 (forward motion allowed)
Off. Pullup for TTL, or insufficient activation current flowing for optos.	1 (forward motion allowed)	0 (forward motion prohibited)

Examples

```
'Galil DMC Code Example
MG _LFA;' Display the status of the A axis forward limit switch
```

_LF applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

_LRm *Reverse Limit Switch Operand*



Usage	variable= _LRm	Holds a value
Operands	_LRm	Operand has special meaning, see Remarks

Description

The _LR operand contains the state of the reverse limit.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis of reverse limit switch	

N/A

Remarks

- _LR is an operand with the following output
  - _LRm= 1 when the limit switch state will allow motion in the reverse direction.
  - _LRm= 0 when the limit switch state will not allow motion in the reverse direction.
- This operand is not a direct readout of the digital input and is affected by the command CN.
- See Connecting Hardware in User Manual for active/inactive state

Values of _LR

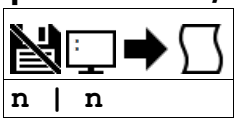
Digital input activation	_LR value for CN-1	_LR value for CN1
On. Grounded for TTL, or sufficient activation current flowing for optos.	0 (reverse motion prohibited)	1 (reverse motion allowed)
Off. Pullup for TTL, or insufficient activation current flowing for optos.	1 (reverse motion allowed)	0 (reverse motion prohibited)

Examples

```
'Galil DMC Code Example
MG _LRA display the status of the a axis reverse limit switch
```

_LR applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

| *Bitwise OR Operator*



n | n

Usage	variable = (value1 | value2)	Performs an operation between two values or evaluated statements

Description

The | symbol is the bitwise OR operator used with IF, JP, and JS decisions, and also to perform bitwise ORING of values.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	-2,147,483,648	2,147,483,647	N/A	1/65,536	Value to use with OR operator	

Remarks

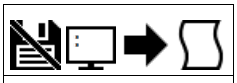
- For IF, JP, and JS, the values used for m are typically the results of logical expressions such as (x > 2) | (y=8)
- The result of this operation is a value, which is not valid on its own. It must be coupled with a command. See examples below.

Examples

```
'Galil DMC Code Example
'Bitwise use
var1= $F;'00001111
var2= $F0;'1111000
MG (var1 | var2)
EN

REM Returned: 255.0000 (same as 11111111)
```

```
'Galil DMC Code Example
'Conditional Use
var1= $F;'00001111
var2= $F0;'1111000
IF (var1 = $F) | (var2 = $F1)
 MG "True"
ELSE
 MG "False"
ENDIF
EN

REM Returned: True
```

~ *Variable Axis Designator*



~s= "str"
~s= n

Description

Variable axis designator. Each variable can be assigned an indivudal axis, a vector plane, or a virtual axis. Motion commands on the variable will then apply to the assigned axis.

Commands supporting variable axes are denoted in this command reference with the following icon.

Variable axis supported icon

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
s	a	h	N/A	N/A	Variable axis name	a,b,c,d,e,f,g,h supported
str	"A"	"H"	N/A	String	Name of axis	"A", "B", "C", "D", "E", "F", "G", "H" supported
	"M"	"N"	N/A	String	Virtual axis	"M", "N" supported
	"S"	"T"	N/A	String	Coordinate System	"S","T" supported
n	0	7	N/A	1	Index of the axis	A= 0, B= 1, C= 2, etc.
	8	9	N/A	1	Coordinate System	S=8, T=9
	10	11	N/A	1	Virtual Axis	M= 11, N=10

Remarks

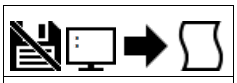
- ~s contains the axis number as defined by n and can be used in expressions (see example)

Examples

```
'Galil DMC Code Example
~a= 2;~b= 6;'     Sets ~a to 2(Z axis).  Sets ~b to 6 (G axis)
MG "~a=",~a;'    Print axis number
MG "~b=",~b;'    Printe axis number
PR~a= 1000;'     Relative position move 1000 counts on ~a variable (set as Z axis)
JG~b= 9000;'     Set jog speed of ~b variable (set as G axis) to 9000 cts/sec
BG ~a~b;'        Begin motion on ~a and ~b variables (Z and G)
```

~ applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

+ *Addition Operator*



n + n

Usage	variable = (value1 + value2)	Performs an operation between two values or evaluated statements

Description

The + symbol is the addition operator. It takes as arguments any two values, variables, array elements, operands, or At functions (@SIN[]) and returns a value equal to the sum of the arguments.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	-2,147,483,648	2,147,483,647	N/A	1/65,536	Value to use in addition operation	

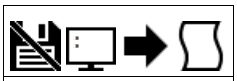
Remarks

- This is a binary operator (takes two arguments and returns one value). The result of this operation is a value, which is not valid on its own. It must be coupled with a command. See examples below.
- Mathmatical operations are calculated left to right rather than multiplication and division calculations performed prior to addition and subraction.
  - Example: 1+2*3 = 9;' not 7
- It is recommended that parenthesis be used when more than one mathmatical operation is combined in one command.
  - Example: var = ((10*30)+(60/30));' evaluates as 302
  - var = 10*30+60/30;' evalutes as 12

Examples

```
'Galil DMC Code Example
:var1 = 1+2
:var2 = var1 + 1
:MG var2 + 2
 6.0000
:
```

< *Less than comparator*



$n_0 < n_1$

Usage	variable = (value1 < value2)	Performs an operation between two values or evaluated statements

Description

"Less than" comparator for testing if one value is less than another. Comparators are used in mathematical expressions, IFs, and in conditional jumps. The result is a boolean.

Comparators in DMC Code

Symbol	Comparator
<	Less than
>	Greater than
=	Equal to
<=	Less than or equal to
>=	Greater than or equal to
<>	Not equal to

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n0	-2,147,483,648	2,147,483,647	N/A	1/65,536	Value to test	
n1	-2,147,483,648	2,147,483,647	N/A	1/65,536	Value to test	

Remarks

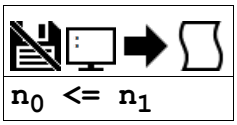
- A comparator is not a command and is not valid individually. Instead, the above expression would be used as part of a jump (JP,JS), IF expression, or assignment. See examples below.
- If $n_0 < n_1$, the expression will evaluate to 1.0000. If the comparision is false, it will evaluate to 0.0000.
- Evaluation occurs left to right. Use parenthesis for operator precedence.

Examples

```
'Galil DMC Code Example
:bool= (1<2)
:MG bool
 1.0000
:bool= (1<0)
:MG bool
 0.0000
:
```

```
'Galil DMC Code Example
REM Example to find the largest
REM value in an array
REM
REM ****************************
REM Create an array and fill it
len= 5
DM array[len]
array[0]= 5
array[1]= 100.0001
array[2]= 42
array[3]= 3.14
array[4]= 100
JS #max;' call max subroutine
MG "Max value is ", max
EN
REM
REM ****************************
REM Find max element in array
#max
i= 0
max = -2147483648;' start at min
#max_h
IF (array[i] > max)
 max = array[i]
ENDIF
i= i+1
JP #max_h, (i < len)
EN
REM
REM ****************************
REM Program output
REM :XQ
REM :
REM Max value is 100.0001
```

<= *Less than or Equal to comparator*



Usage	variable = (value1 <= value2)	Performs an operation between two values or evaluated statements

Description

"Less than or Equal to" comparator for testing if one value is less than or equal to another. Comparators are used in mathematical expressions, IFs, and in conditional jumps. The result is a boolean.

Comparators in DMC Code

Symbol	Comparator
<	Less than
>	Greater than
=	Equal to
<=	Less than or equal to
>=	Greater than or equal to
<>	Not equal to

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n0	-2,147,483,648	2,147,483,647	N/A	1/65,536	Value to test	
n1	-2,147,483,648	2,147,483,647	N/A	1/65,536	Value to test	

Remarks

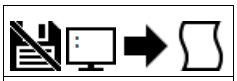
- A comparator is not a command and is not valid individually. Instead, the above expression would be used as part of a jump (JP,JS), IF expression, or assignment. See examples below.
- If n_0 <= n_1, the expression will evaluate to 1.0000. If the comparision is false, it will evaluate to 0.0000.
- Evaluation occurs left to right. Use parenthesis for operator precedence.

Examples

```
'Galil DMC Code Example
:bool= (1 <= 2)
:MG bool
 1.0000
:bool= (2 <= 2)
:MG bool
 1.0000
:bool= (3 <= 2)
:MG bool
 0.0000
:
```

```
'Galil DMC Code Example
max= 2.05
min= 1.47
value = 0.025
JS #check
value = 1.471
JS #check
EN
REM
REM ********************************
REM Determine if in range
#check
inrange= 0
IF ((value >= min) & (value <= max))
 inrange= 1
ENDIF
IF (inrange)
 MG "Value ",value," in range"
ELSE
  MG "Value ",value," NOT in range"
ENDIF
EN
REM
REM ********************************
REM Program output
REM :XQ
REM :
REM  Value 0.0250 NOT in range
REM  Value 1.4710 in range
```

<> *Not Equal to comparator*



n_0 <> n_1

Usage	variable = (value1 <> value2)	Performs an operation between two values or evaluated statements

Description

"Not Equal to" comparator for testing if one value is not equal to another. Comparators are used in mathematical expressions, IFs, and in conditional jumps. The result is a boolean.

Comparators in DMC Code

Symbol	Comparator
<	Less than
>	Greater than
=	Equal to
<=	Less than or equal to
>=	Greater than or equal to
<>	Not equal to

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n0	-2,147,483,648	2,147,483,647	N/A	1/65,536	Value to test	
n1	-2,147,483,648	2,147,483,647	N/A	1/65,536	Value to test	

Remarks

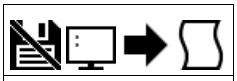
- A comparator is not a command and is not valid individually. Instead, the above expression would be used as part of a jump (JP,JS), IF expression, or assignment. See examples below.
- If n_0 <> n_1, the expression will evaluate to 1.0000. If the comparision is false, it will evaluate to 0.0000.
- Evaluation occurs left to right. Use parenthesis for operator precedence.

Examples

```
'Galil DMC Code Example
:bool= (1 <> 2)
:MG bool
 1.0000
:bool= (2 <> 2)
:MG bool
 0.0000
```

```
'Galil DMC Code Example
REM Lock out code until
REM a particular digital
REM input pattern is detected
#AUTO
JS #lock;'block until pattern
REM
REM
REM Rest of code here
REM
REM
EN
REM
REM *************************
#lock
JP #lock, (_TI0 <> 170)
EN
```

= *Assignment Operator*



```
str = n
```

Description

The = operator is the assignment operator for the controller. 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

Argument	Min	Max	Default	Resolution	Description	Notes
str	1 char	8 chars	N/A	String	Variable name to access	
n	-2,147,483,648	2,147,483,647	see Notes	1/65,536	Value to assign to specified variable	Default n, or n = null results in a query of the value of variable

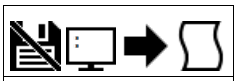
Remarks

- None

Examples

```
'Galil DMC Code Example
:x= 5
:x= ?
 5.0000
:MG x
 5.0000
'define and initialize x to 5
'print x two different ways
```

= *Equal to comparator*



$n_0 = n_1$

Usage	variable = (value1 = value2)	Performs an operation between two values or evaluated statements

Description

"Equal to" comparator for testing if one value is equal to another. Comparators are used in mathematical expressions, IFs, and in conditional jumps. The result is a boolean.

Comparators in DMC Code

Symbol	Comparator
<	Less than
>	Greater than
=	Equal to
<=	Less than or equal to
>=	Greater than or equal to
<>	Not equal to

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n0	-2,147,483,648	2,147,483,647	N/A	1/65,536	Value to test	
n1	-2,147,483,648	2,147,483,647	N/A	1/65,536	Value to test	

Remarks

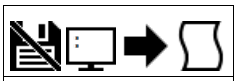
- A comparator is not a command and is not valid individually. Instead, the above expression would be used as part of a jump (JP,JS), IF expression, or assignment. See examples below.
- If $n_0 = n_1$, the expression will evaluate to 1.0000. If the comparision is false, it will evaluate to 0.0000.
- Evaluation occurs left to right. Use parenthesis for operator precedence.

Examples

```
'Galil DMC Code Example
:bool= (1=0)
:MG bool
 0.0000
:bool= (3.14=3.14)
:MG bool
 1.0000
:
```

```
'Galil DMC Code Example
REM Checks for a digital
REM input pattern and
REM sets a bit if matched
#loop
IF (_TI0 = 170)
 SB 1
ELSE
 CB 1
ENDIF
JP #loop
```

> *Greater than comparator*



$n_0 > n_1$

Usage	variable = (value1 > value2)	Performs an operation between two values or evaluated statements

Description

"Greater than" comparator for testing if one value is greater than another. Comparators are used in mathematical expressions, IFs, and in conditional jumps. The result is a boolean.

Comparators in DMC Code

Symbol	Comparator
<	Less than
>	Greater than
=	Equal to
<=	Less than or equal to
>=	Greater than or equal to
<>	Not equal to

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n0	-2,147,483,648	2,147,483,647	N/A	1/65,536	Value to test	
n1	-2,147,483,648	2,147,483,647	N/A	1/65,536	Value to test	

Remarks

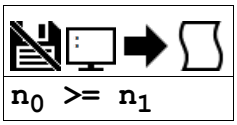
- A comparator is not a command and is not valid individually. Instead, the above expression would be used as part of a jump (JP,JS), IF expression, or assignment. See examples below.
- If $n_0 > n_1$, the expression will evaluate to 1.0000. If the comparision is false, it will evaluate to 0.0000.
- Evaluation occurs left to right. Use parenthesis for operator precedence.

Examples

```
'Galil DMC Code Example
:bool= (1>2)
:MG bool
 0.0000
:bool= (1>0)
:MG bool
 1.0000
:
```

```
'Galil DMC Code Example
REM Example to find the largest
REM value in an array
REM
REM ****************************
REM Create an array and fill it
len= 5
DM array[len]
array[0]= 5
array[1]= 100.0001
array[2]= 42
array[3]= 3.14
array[4]= 100
JS #max;' call max subroutine
MG "Max value is ", max
EN
REM
REM ****************************
REM Find max element in array
#max
i= 0
max = -2147483648;' start at min
#max_h
IF (array[i] > max)
 max = array[i]
ENDIF
i= i+1
JP #max_h, (i < len)
EN
REM
REM ****************************
REM Program output
REM :XQ
REM :
REM Max value is 100.0001
```

>= *Greater than or Equal to comparator*



Usage	variable = (value1 >= value2)	Performs an operation between two values or evaluated statements

Description

"Greater than or Equal to" comparator for testing if one value is greater than or equal to another. Comparators are used in mathematical expressions, IFs, and in conditional jumps. The result is a boolean.

Comparators in DMC Code

Symbol	Comparator
<	Less than
>	Greater than
=	Equal to
<=	Less than or equal to
>=	Greater than or equal to
<>	Not equal to

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n0	-2,147,483,648	2,147,483,647	N/A	1/65,536	Value to test	
n1	-2,147,483,648	2,147,483,647	N/A	1/65,536	Value to test	

Remarks

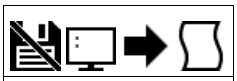
- A comparator is not a command and is not valid individually. Instead, the above expression would be used as part of a jump (JP,JS), IF expression, or assignment. See examples below.
- If n_0 >= n_1, the expression will evaluate to 1.0000. If the comparision is false, it will evaluate to 0.0000.
- Evaluation occurs left to right. Use parenthesis for operator precedence.

Examples

```
'Galil DMC Code Example
:bool= (1 >= 2)
:MG bool
 0.0000
:bool= (2 >= 2)
:MG bool
 1.0000
:bool= (3 >= 2)
:MG bool
 1.0000
:
```

```
'Galil DMC Code Example
max= 2.05
min= 1.47
value = 0.025
JS #check
value = 1.471
JS #check
EN
REM
REM ********************************
REM Determine if in range
#check
inrange= 0
IF ((value >= min) & (value <= max))
 inrange= 1
ENDIF
IF (inrange)
 MG "Value ",value," in range"
ELSE
  MG "Value ",value," NOT in range"
ENDIF
EN
REM
REM ********************************
REM Program output
REM :XQ
REM :
REM  Value 0.0250 NOT in range
REM  Value 1.4710 in range
```

AB *Abort*



AB

AB n

Usage	AB n ...	Arguments specified with an implicit, comma-separated order
Operands	_AB	Operand has special meaning, see Remarks

Description

The AB command is a command to issue an abort to controller operation.

AB (Abort) stops motion instantly without a controlled deceleration. If there is a program operating, AB can also be specified to abort the program and all running threads. The command, AB, will shut off the motors for any axis in which the off on error function is enabled (see command "OE").

Arguments

Argument	Value	Description	Notes
n	0	Abort motion and the program operation	Default if omitted
	1	Abort motion only	

Remarks

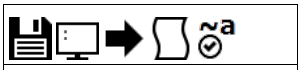
- _AB gives state of Abort Input, 1 inactive and 0 active.
- AB aborts motion on all axes in motion and cannot stop individual axes.

Examples

```
'Galil DMC Code Example
:AB;'     Stops motion
:OE*= 1;' Enable off on error on axes
:AB;'     Shuts off motor command and stops motion
```

```
'Galil DMC Code Example
#a;'         Label - Start of program
JG 20000;'   Specify jog speed on A-axis
BG A;'       Begin jog on A-axis
WT 5000;'    Wait 5000 msec
AB 1;'       Stop motion without aborting program
WT 5000;'    Wait 5000 milliseconds
SH ;'         Servo Here
JP #a;'      Jump to Label A
EN;'         End of the routine
'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.
```

AC *Acceleration*



ACm= n

AC n,n,n,n,n,n,n,n

Usage	ACm= n	Arguments specified with a single axis mask and an assignment (=)
	AC n ...	Arguments specified with an implicit, comma-separated order
Operands	_ACm	Operand holds the value last set by the command

Description

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.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
	M	N	N/A	Axis	Virtual axis to assign value	
n	1,024	1,073,740,800	256,000	1,024	Value of proportional term	Resolution and Min depends on TM, see remarks

Remarks

- The DC command is used to designate deceleration
- Specify realistic acceleration rates based on your physical system such as
  - motor torque rating
  - loads
  - amplifier current rating.
- Specifying an excessive acceleration will cause large following error during acceleration and the motor will not follow the commanded profile.
- The acceleration feedforward command (FA) will help minimize the error for aggressive accelerations

Resolution

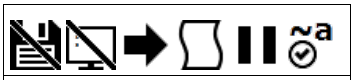
- The Min and Resolution depends upon the update rate setting (TM). The equation to calculate these values is:
  - Resolution = Min = 1024*(1000/TM)^2
  - example:
    - With TM 500 the minimum AC setting and resolution is 4096 cnts/second^2
    - resolution = 1024*(1000/500)^2 = 4096

Examples

```
'Galil DMC Code Example
REM 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 150000,200000,300000,400000
a= _ACB;' Assigns the B acceleration to the variable a
```

AC applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

AD *After Distance*



ADm= n

AD n,n,n,n,n,n,n,n

Usage	ADm= n	Arguments specified with a single axis mask and an assignment (=)
	AD n ...	Arguments specified with an implicit, comma-separated order

Description

Trippoint to block command execution until a given distance is traversed. This is a profiled trippoint which means it depends on the motion profiler and not the actual motor encoder. AD can only be used when there is commanded motion on the axis.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	0	2,147,483,647	N/A	1	Distance of motion	Cannot specify more than 1 argument at a time

Remarks

- AD will hold up the execution of the following command until one of the following conditions have been met
  - The commanded motor position crosses the specified relative distance from the start of the move
  - The motion profiling on the axis is complete
  - If in jog (JG) mode, the commanded motion is in the direction which moves away from the specified position
- Not valid for a slave during ECAM or Gearing, use MF and MR
- 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
- The AD command is accurate to the number of counts that occur in 2*TM msec
- AD command will be affected when the motion smoothing time constant, IT, is not 1. See IT command for further information
- AD measures incremental distance from start of move on one axis

Examples

```
'Galil DMC Code Example
#a
DP 0,0;'                    Zero position
PR 10000,20000;'            Specify position relative moves
BG ;'                       Begin motion
AD 5000;'                   After A reaches 5000
MG "Halfway to A";TP A;'    Send message
AD ,10000;'                 After B reaches 10000
MG "Halfway to B";TP B;'    Send message
EN;'                        End Program
```

AD applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

AF *Analog Feedback Select*

AFm= n

AF n,n,n,n,n,n,n,n

Usage	AFm= n	Arguments specified with a single axis mask and an assignment (=)
	AF n ...	Arguments specified with an implicit, comma-separated order
Operands	_AFm	Operand holds the value last set by the command

Description

The AF command configures analog feedback mode for the PID filter.

The controller ADC can be used as position feedback for the axis control law. The analog input used for feedback is fixed and uses the input that corresponds with the axis letter. For example, Analog input 1 is used for the A axis.

Sinusoidal feedback encoders are also configured by the AF command.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	0	1	0	1	Use the controller ADC as servo feedback	1= analog, 0= digital feedback, DB-28040 Required
	-1	-1	0	0	Analog hardware sampled in the servo interrupt	This provides evenly sampled analog data for both the data record and the RA/RD/RC function.
	5	12	0	1	Sinusoidal encoder input used with 2^n interpolation counts per encoder cycle	DB-28104 Required

Remarks

- Below is the feedback in counts decoded by the controller hardware when reading in analog feedback for certain analog input ranges.

	12 Bit ADC	16 Bit ADC
+/-5 V, +/-10 V	-2048 to 2047 counts	-32768 to 32767 counts
0-5 V, 0-10 V	0 to 4095 counts	0 to 65535 counts

- The DB-28040 is needed to use Analog Feedback. The analog input used for feedback is fixed and uses the input that corresponds with the axis letter. For example, Analog input 1 is used for the A axis, Analog input 2 is used for the B axis, etc.
- The DB-28104 is needed to use Sine Feedback. Differential encoder inputs must be used when using digital encoders with the DB-28104. Consult the factory for single-ended use.

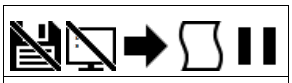
- When using Sin/Cos encoders (AF5-12)
  - The encoder must be connected to the controller prior to issuing the AF command.
  - TP will provide position resolution of $2^{(\_AFm)}$ counts per cycle. One cycle is four quadrature counts.
    - For example, if an encoder shows a change in TP of 8000 counts with AF0. The same distance at AF 5 would be give by $8000/4 * 2^5 = 64000$

Examples

```
'Galil DMC Code Example
AF 1;'           Analog feedback on A axis
v1= _AFA;'       Assign feedback type to variable
KP 1;'           Assigns PID's for motor using analog feedback on A-axis
KD 10;'
KI 0.5;'
```

```
'Galil DMC Code Example
AF 12;'          Sets sine/consine feedback to 2^12= 4096 counts/period
AF 8;'           Sets sine/cosine feedback to 2^8= 256 counts/period
```

AI *After Input*



AI n

Usage	AI n ...	Arguments specified with an implicit, comma-separated order

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

Argument	Min	Max	Default	Resolution	Description	Notes
n	1	16	N/A	1	General inputs to use for trippoint	+n = High trigger. -n = low trigger. 9-16 only valid for 5-8 axis controller
	17	80	N/A	1	Extended inputs to use for trippoint	DB-14064 Required
	81	96	N/A	1	Aux encoder inputs to use for trippoint	

Remarks

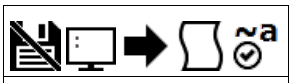
- The AI command actually halts execution until specified input is at desired logic level. Use the conditional Jump command (JP) or input interrupt (II) if you do not want the program sequence to halt.
- AI functions only on local input points. See Example below for network based digital inputs.

Examples

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

```
'Galil DMC Code Example
REM When using a remote I/O device (e.g. the RIO), the following provides
REM a similar function as AI. Assume that the remote device is already
REM configured on handle C (see IH)
'code before
JS #remote;' this call blocks and waits for the remote logic to return
'code after
EN
'***** The example subroutine *****
#remote
WT 10;' wait a reasonable interval so we don't flood the network
JP #remote,(@IN[3001] = 1);'loop while input 1 on the remote device is high
EN;' return to calling code.
```

AL *Arm Latch*



AL mm

Usage	AL mm	Argument is an axis mask
Operands	_ALm	Operand has special meaning, see Remarks

Description

The AL command enables the latch 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 from the specified digital input.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
mm	A	ABCDEFGH	N/A	Multi-Axis Mask	Encoder to latch	Latch main encoder
mm	SA	SASBSCSDSESFSGSH	N/A	Multi-Axis Mask	Encoder to latch	Latch aux encoder
mm	TA	TATBTCTDTETFTGTH	N/A	Multi-Axis Mask	Index input to trigger latch	Main encoder is latched from the index pulse instead of a digital input

Remarks

Latch input by Axis

Axis	Latch Input
A	Input 1
B	Input 2
C	Input 3
D	Input 4
E	Input 9
F	Input 10
G	Input 11
H	Input 12

- The command RL returns the latched position
- _ALm contains the state of the specified latch. 0 = not armed, 1 = armed
- The CN command can be used to change the polarity of the latch function
- The latch function is available on incremental quadrature encoder inputs only. For other position capture methods contact Galil.

Examples

```
'Galil DMC Code Example
#start
AL A;'                  Arm A-axis latch
JG 50000;'              Set up jog at 50000 counts/sec
BG A;'                  Begin the move
#loop;'                 Loop until latch has occurred
JP #loop,(_ALA=1)
RL A;'                  Transmit the latched position
EN;'                    End of program
```

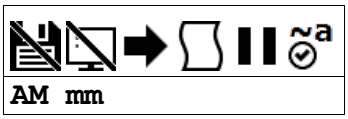
```
'Galil DMC Code Example
REM Homing routine using the AL command to detect the Motor's index position
#start
AL TA;'                  Arm A-axis latch. Latch will trigger off the index pulse
JG 50000;'               Set up jog at 50000 counts/sec
BG A;'                   Begin the move
#loop;'                  Loop until latch has occurred
JP #loop,(_ALA=1)
ST A;'                    Stop the jog
AM A
PAA= _RLA;'               Set up a move to return to the latched position
BG A
AM A
WT 100;'                  Allow for settling.
REM Checking that KI has eliminated error (TE) would be more thorough
DP 0;'                    Zero position
MG "A Homed";'           Report status
EN;'                     End of program
```

```
'Galil DMC Code Example
REM manual find index using latch off of index pulse using varible axes
#index
'settings
~a= 0;'axis number
s1spd= 10000;'stage 1 speed
s2spd= 1000;'stage 2 speed
```

```
'jog until index is latched
JG~a= s1spd;BG ~a
WT 1000
AL T~a
#a;JP #a,_AL~a=1
ST ~a;AM ~a
'Return to latched position
SP~a= s2spd
PA _RL~a;BG ~a;MC ~a
JS #stl;DP~a= 0
EN

'wait for axis to settle
#stl
WT 2;JP #stl,@ABS[_TE~a]>2
WT 2;JP #stl,@ABS[_TE~a]>0
WT 2;JP #stl,@ABS[_TE~a]>0
EN
```

AL applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

AM *After Move*



Usage	AM mm	Argument is an axis mask

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 to be complete.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
mm	A	ABCDEFGH	ABCDEFGH	Multi-Axis Mask	Axes to wait for profiled motion to complete	
	S	T	N/A	Multi-Axis Mask	Vector plane to wait for profiled motion to complete	

Remarks

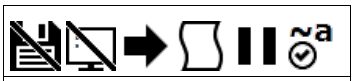
- AM is a very important command for controlling the timing between multiple move sequences.
  - For example, if the A-axis is in the middle of a position relative move (PR) you cannot make a position absolute move (PAA, BGA) until the first move is complete. Use AMA to halt the program sequence until the first profiled motion is complete.
  - AM tests for profile completion only. The actual motor may still be moving. To halt the program sequence until the actual physical motion has completed, use the MC command.
  - To test motion complete without halting the program sequence, use the operand _BGn, which will be zero when profiled motion is complete (see BG command).

Examples

```
'Galil DMC Code Example
#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
```

AM applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

AP *After Absolute Position*



APm= n

AP n,n,n,n,n,n,n,n

Usage	APm= n	Arguments specified with a single axis mask and an assignment (=)
	AP n ...	Arguments specified with an implicit, comma-separated order

Description

The AP command will hold up the execution of the following command until the actual motor position crosses the specified position. This trippoint does not rely on the profiler, but on actual encoder position.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	-2,147,483,648	2,147,483,647	N/A	1	Position trippoint value	Only one axis may be specified at a time.

Remarks

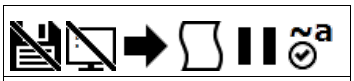
- For AP command to clear, one of the following conditions have been met:
  - The actual motor position crosses the specified absolute position.
  - The motion profiling on the axis is complete.
  - The commanded motion is in the direction which moves away from the specified position.
- The units of the command are quadrature counts.
- When using a stepper motor, the AP trippoint condition is satisfied when the stepper position (TD) has crossed the specified position.
  - For further information see Chapter 6 of the User Manual "Stepper Motor Operation".
- Not valid for a slave during ECAM or Gearing - use MF and MR.
- The motion profiler must be active before the AP command is used.
- AP is accurate to the number of counts that occur in 2*TM msec
- AP tests for absolute position. Use the AD command to measure incremental distances.

Examples

```
'Galil DMC Code Example
#test;' Program B
DP 0;' 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
```

AP applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

AR *After Relative Distance*



ARm= n

AR n,n,n,n,n,n,n,n

Usage	ARm= n	Arguments specified with a single axis mask and an assignment (=)
	AR n ...	Arguments specified with an implicit, comma-separated order

Description

The After Relative (AR) command is a trippoint used to control the timing of events. This command will hold up the execution of the following command until one of the following conditions have been met:
1. The commanded motor position crosses the specified relative distance from either the start of the move or the last AR or AD command.
2. The motion profiling on the axis is complete.
3. If in jog (JG) mode, the commanded motion is in the direction which moves away from the specified position.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	-2,147,483,648	2,147,483,647	N/A	1	Relative position for trippoint	Only one axis may be specified at a time.

Remarks

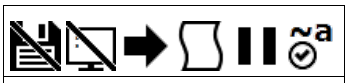
- The units of the command are quadrature counts.
- When using a stepper motor, this condition is satisfied when the stepper position (as determined by the output buffer) has crossed the specified Relative Position.
  - For further information see Chapter 6 of the User Manual "Stepper Motor Operation".
- If the direction of the motion is reversed when in position trackig mode (see PT command), the starting point for the trippoint is reinitialized to the point at which the motion reversed.
- The motion profiler must be active before the AR command is issued.
- Not valid for a slave during ECAM or Gearing - use MF and MR.
- Note: AR will be affected when the motion smoothing time constant, IT, is not 1. See IT command for further information.
  - AP is accurate to the number of counts that occur in 2*TM msec
- 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.

Examples

```
'Galil DMC Code Example
#a;'               Begin Program
DP 0
JG 50000;'         Specify speed
BG A;'             Begin motion
#b;'               Label
AR 25000;'         After passing 25000 counts of relative distance on A-axis
MG "Passed";TP A;' Send message on A-axis
JP #b;'            Jump to Label #B
EN;'               End Program
```

AR applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

AS *At Speed*



AS mm

Usage	AS mm	Argument is an axis mask

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.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
mm	A	ABCDEFGHST	ABCDEFGH	Multi-Axis Mask	Axes to use for AS trippoint	

Remarks

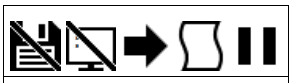
- If the speed is not reached, the trippoint will be triggered after the speed begins diverging from the AS value.
- 'The AS command applies to a trapezoidal velocity profile only with linear acceleration. AS used with Smoothing profiling will be inaccurate.

Examples

```
'Galil DMC Code Example
#speed;'          Program
PR 100000;'       Specify position
SP 10000;'        Specify speed
BG A;'            Begin A
AS A;'            After speed is reached
MG "At Speed";'   Print Message
EN;'              End programm
```

AS applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

AT *At Time*



AT n

Usage	AT n ...	Arguments specified with an implicit, comma-separated order

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

Argument	Min	Max	Default	Resolution	Description	Notes
n	-2,147,483,648	2,147,483,647	0	2	Specify a wait time for AT trippoint	See Remarks

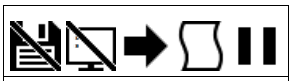
Remarks

- 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)

Examples

```
'Galil DMC Code Example
#ex
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)
EN

' jog propotional to analog input example with AT in ms
#main
AT 0;'             set time reference for AT command
JG 0;BG A;'         start Jog mode
gain= 1
#atloop
 jgspd= gain*@AN[1]
 JG jgspd
 AT -100;'          wait 100 ms from last time reference (last AT-n or AT0)
REM same functionality as AT-100 would be
REM AT 100;AT0
JP #atloop
```

AV *After Vector Distance*



AV n_0,n_1

Usage	AV n ...	Arguments specified with an implicit, comma-separated order
Operands	_AVS _AVT	Operand has special meaning, see Remarks

Description

The AV command 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.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n_0	0	2,147,483,647	0	1	Vector distance to be executed in the S coordinate system	
n_1	0	2,147,483,647	0	1	Vector distance to be executed in the T coordinate system	

Remarks

- The units of the command are quadrature counts.
- _AVS contains the vector distance from the start of the sequence in the S coordinate system
- _AVT contains the vector distance from the start of the sequence in the T coordinate system.

Examples

```
'Galil DMC Code Example
#move;'         Label
DP 0,0
CA T;'           Specify the T coordinate system
LM AB;'          Linear move for A,B
LI 1000,2000;' Specify distance
LI 2000,3000;' Specify distance
LE
BG T;'           Begin motion in the T coordinate system
AV ,500;'       After path distance = 500,
MG "Path>500"
TP AB;'          Print position of A and B axes
EN;'            End Program
'Vector Distance is calculated as the square root of the sum of the
'squared distance for each axis in the linear or vector mode.
```

AV applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

BA *Brushless Axis*

BA mm

Usage	BA mm	Argument is an axis mask
Operands	_BAm	Operand has special meaning, see Remarks

Description

BA is used to configure the controller for sinusoidal operation.

Third-Party Sine Drives Requiring Dual Analog Inputs (Rare)

In rare cases, some third-party sinusoidal drives require two analog signals to perform commutation. In this case, the BA command configures the controller axes for sinusoidal commutation and reconfigures the controller to reflect the actual number of motors that can be controlled. In this configuration, each axis requires 2 motor command signals. The second motor command signals will always be associated with the highest axis on the controller. For example a 3 axis controller with A and C configured for sinusoidal commutation will require 5 command outputs (a 5 axis controller), where the second outputs for A and C will be the D and E axes respectively.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
mm	A	ABCDEFGH	N/A	Multi-Axis Mask	Axes to initialize for sine amps	mm = "" removes all axes configured for sine commutation
	N	N	N/A	Multi-Axis Mask	Disable sine initialization for all axes.	

Remarks

Third-Party Sine Drives Requiring Dual Analog Inputs (Rare)

If more than one dual DAC setup is going to be used, both axes must be specified in the same BA command.

- _BAm indicates the axis number of the auxiliary DAC used for the second phase of the selected sinusoidal axis. The axis numbers start with zero for the A axis DAC. If the motor is configured as standard servo or stepper motor, _BAn contains 0.

Examples

```
'Galil DMC Code Example
BA A;'        Configure axis A for sine amp
BM 200;'      Length of electrical cycle in counts--required setting for commutation
BZ 3<1000;'   Commutate motor with BZ method using 3V and timeout after 1000 msec
SH A;'        Enable motor, ready for commands
EN
```

```
'Galil DMC Code Example
'this exmaple is exclusively for Third-Party Sine Drives
'that require Dual Analog Inputs which is rare
'
'controller revision string will show DMC4040
BA AB
MG _BAA;'  will return 2.0000
MG _BAB;'  will return 3.0000
'Controller revision string will show DMC4020
```

BA applies to DMC40x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

BB *Brushless Phase Begins*

BBm= n

BB n,n,n,n,n,n,n,n

Usage	BBm= n	Arguments specified with a single axis mask and an assignment (=)
	BB n ...	Arguments specified with an implicit, comma-separated order
Operands	_BBm	Operand holds the value last set by the command

Description

The BB function describes the position offset between the Hall transition point and theta = 0, for a sinusoidally commutated motor. This is used when doing hall initialization of a sine commutated drive.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	-359.98	359.98	0	1/32	Phase offset of hall sensors	

Remarks

- This command must be saved in non-volatile memory to be effective upon reset.

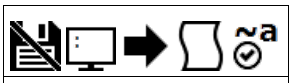
Examples

```
'Galil DMC Code Example
BB ,30,,60;' The offsets for the Y and W axes are 30  and 60  respectively
```

```
'Galil DMC Code Example
BB 30;' set offset of 30 degrees for A axis
```

BB applies to DMC40x0,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

BC *Brushless Calibration*



BC mm

Usage	BC mm	Argument is an axis mask
Operands	_BCm	Operand has special meaning, see Remarks

Description

The BC command is used to initialize a motor for sine commutation using hall sensors.

The function BC monitors the status of the Hall sensors of a sinusoidally commutated motor, and resets the commutation phase upon detecting the first hall sensor. This procedure replaces the estimated commutation phase value with a more precise value determined by the hall sensors.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
mm	A	ABCDEFGH	ABCDEFGH	Multi-Axis Mask	Axes to initialize with hall commutation	

Remarks

Steps for BC sine initialization

1. Specify the axis/axes for initialization with the BA command
2. Specify the number of encoder counts per magnetic phase of the motor with the BM command (see command for examples)
3. Issue BI to select the inputs to use as hall inputs.
4. Servo the motor and verify it holds position
    1. If the motor will not servo, verify encoder is functional. If it is, then re-verify hall wiring
5. Issue the BC command, then issue a small jog until a hall transition occurs.
6. The motor is now fully commutated based off of the hall sensor feedback.
7. (Optional) Use the BB command to correct for hall offsets from true magnetic 0 of the motor.

Operand Usage

- _BCm contains the state of the Hall sensor inputs. This value should be between 1 and 6. 0 and 7 are invalid hall states.

Examples

```
'Galil DMC Code Example
BA A;'       Enable sine drive
BMA= 2000;' Set brushless modulus to 2000 cnts
BIA= 4;'    Hall inputs on IN4,5,and 6
MG _BCA;'   Read hall state
EN
```

BC applies to DMC40x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

BD *Brushless Degrees*

BDm= n

BD n,n,n,n,n,n,n,n

Usage	BDm= n	Arguments specified with a single axis mask and an assignment (=)
	BD n ...	Arguments specified with an implicit, comma-separated order
Operands	_BDm	Operand has special meaning, see Remarks

Description

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

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	0	360	6	1/32	Brushless motor angle in degrees	

Remarks

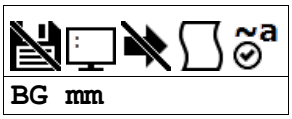
- Using BD to set a brushless degree overrides the current brushless degrees set by the BZ/BX/BI initialization routines.
- Once initialized, BD is updated by the firmware to the current brushless degree value.
- n = ? queries the current brushless degrees
- _BDm contains the commutation phase of the specified axis.

Examples

```
'Galil DMC Code Example
BDA= 100;'      Set Brushless degrees for A axis to 100
MG _BDA;'       Report the brushless degrees for A axis
```

BD applies to DMC40x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

BG *Begin*



Usage	BG mm	Argument is an axis mask
Operands	_BGm	Operand has special meaning, see Remarks

Description

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

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
mm	A	ABCDEFGH	ABCDEFGH	Multi-Axis Mask	Axes to begin motion	Any combination of axes is acceptable. BG with no arguments begins motion on all axes
	S	T	N/A	Multi-Axis Mask	Vector plane axes to begin motion	Any combination of axes is acceptable
	M	N	N/A	Multi-Axis Mask	Virtual axis to begin motion	Any combination of axes is acceptable

Remarks

- Any combination of Axes, Vector Planes, and Virtual Axes may be mixed to begin motion
- A BG command cannot be executed for any axis in which motion has not completed
  - Slaving to a master in gearing mode is an exception. Gearing does not require the axis to profile a motion and therefore Independent moves may be superimposed on top of gearing.
- Use the AM trippoint to wait for motion complete between moves from embedded code.
- From host code, use one of the following methods to determine motion is complete
  - Poll MG_BGm
  - Use the data record (DR/QR)
  - Use interrupts (EI), if available

Operands

- _BGm contains a '0' if motion complete on the specified axis or coordinate system, otherwise contains a '1'
  - _BGm can be used from host programs to determine if motion is complete by polling the axes of interest

Examples

```
'Galil DMC Code Example
PR 2000,3000,,5000;'  Set up for a relative move
BG ;'                 Start the A,B and D motors moving
```

```
'Galil DMC Code Example
HM ;'   Set up for the homing
BG A;'  Start only the A-axis moving
```

```
'Galil DMC Code Example
JG 1000,4000;'   Set up for jog
BG B;'           Start only the B-axis moving
```

```
'Galil DMC Code Example
bstate= _BGB;'   Assign a 1 to bstate if the B-axis is performing a move
```

```
'Galil DMC Code Example
VM AB;'          Vector Mode
VP 1000,2000;'   Specify vector position
VS 20000;'       Specify vector velocity
BG S;'           Begin coordinated sequence
VP 4000,-1000;' Specify vector position
VE;'             Vector End
```

BG applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

BI *Brushless Inputs*

BIm= n

BI n,n,n,n,n,n,n,n

Usage	BIm= n	Arguments specified with a single axis mask and an assignment (=)
	BI n ...	Arguments specified with an implicit, comma-separated order
Operands	_BIm	Operand holds the value last set by the command

Description

The BI command is used to define the inputs which are used when Hall sensors have been wired for sinusoidally commutated motors. See the BC command for more information about initialization of sine amplifiers via hall inputs

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	0	16	0	1	Select starting General input for hall sensor use	n = 0 clears configuration. Inputs 9-16 only valid for 5-8 axis controller
	17	80	N/A	1	Select starting extended input for hall sensor use	DB-14064 required. See Remarks
	81	94	N/A	1	Select starting auxiliary encoder input for hall sensor use.	

Remarks

- The inputs can be the general use inputs, the auxiliary encoder inputs, or the extended I/O inputs. The Hall sensors of each axis must be connected to consecutive input lines,
  - For example: BI 3 indicates that inputs 3,4 and 5 are used for halls sensors.
  - When using extended IO, the bits must be configured as inputs by the CO command for proper operation

Examples

```
'Galil DMC Code Example
BI , 5;'  The Hall sensor of the Y axis are on inputs 5, 6 and 7.
```

BI applies to DMC40x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

BK *Breakpoint*

BK n_0,n_1

Usage	BK n ...	Arguments specified with an implicit, comma-separated order
Operands	_BK	Operand has special meaning, see Remarks

Description

The BK command causes the controller to pause execution of the given thread at the given program line number. When that line is reached, program execution halts before the line is executed, while all other threads continue running. 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.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n_0	0	1,999	N/A	1	Line number to set breakpoint	n = null resumes execution
n_1	0	7	0	1	Thread number to set breakpoint	If n omitted, default value used.

Remarks

- Only one breakpoint may be armed at any time.
- BK can be armed before or during thread execution.

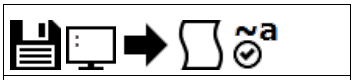
Operand Usage

- _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

Examples

```
'Galil DMC Code Example
: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 *Reverse Software Limit*



BLm= n

BL n,n,n,n,n,n,n,n

Usage	BLm= n	Arguments specified with a single axis mask and an assignment (=)
	BL n ...	Arguments specified with an implicit, comma-separated order
Operands	_BLm	Operand holds the value last set by the command

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.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	-2,147,483,648	2,147,483,647	-2,147,483,648	1	Position for reverse soft limit	

Remarks

- The reverse limit is activated at the position n-1. n = -2147483648 effectively disables the reverse soft limit
- The software limit is specified in counts for a servo system or in microsteps for a stepper system.
- When the reverse software limit is activated, the automatic subroutine #LIMSWI will be executed if it is included in the program.
- If motion is commanded when the axis is already passed the BL value, the axis will profile a small move before the software limit is again detected.
  - This is typically encountered when commanding motion in loops, such as a jog loop.
  - In these scenarios it is recommended to use the #LIMSWI routine to stop the loop when the BL limit has been exceeded.

Examples

```
'Galil DMC Code Example
#test;'  Test Program
AC 1000000;' Acceleration Rate
DC 1000000;' Deceleration Rate
BL -15000;' Set Reverse Limit
JG  -5000;' Jog Reverse
BG A;'    Begin Motion
AM A;'    After Motion (limit occurred)
TP A;'    Tell Position
EN;'     End Program
'
'Galil Controllers also provide hardware limits.
```

BL applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

BM *Brushless Modulo*

BMm= n

BM n,n,n,n,n,n,n,n

Usage	BMm= n	Arguments specified with a single axis mask and an assignment (=)
	BM n ...	Arguments specified with an implicit, comma-separated order
Operands	_BMm	Operand holds the value last set by the command

Description

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

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	1	10,000,000	2,000	1/65,536	Encoder counts per magnetic cycle	

Remarks

- For rotary motors, the magnectic cycle (BM value) is calculated by:
  - BM = encoder counts per revolution / # of pole pairs
- The issuance of BM is required for commutation using one of the following methods:
  - BX
  - BZ
  - BI/BC

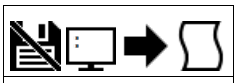
Examples

```
'Galil DMC Code Example
REM Using Galil's BLM motor specifications as an example
cts= 4000;'       4000 encoder counts per revolution
pole= 2;'         2 pole pairs (4 poles total)
BA A
BMA= cts/pole;'   Calculation of BM
BZA= 3.5;'        Commutate using BZ method and 3.5V
SH A
MG "Commutation complete."
EN
```

```
'Galil DMC Code Example
BM ,60000;'      Set brushless modulo for B axis to be 60000
BMC= 100000/3;'   Set brushless modulo for C axis to be 100000/3 (33333.333)
BM ,,,?;'        Interrogate the Brushless Module for the D axis

'example calculating brushless modulus using calculated integers
cts= 4096;'     Counts per rev
pp= 3;'         Pole pairs
BMA= cts/pp

'Changing the BM parameter causes an instant change in the commutation phase.
```

BM applies to DMC40x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

BN *Burn*



Usage	BN	Command takes no arguments
Operands	_BN	Operand has special meaning, see Remarks

Description

The BN command saves certain board parameters in non-volatile EEPROM memory. This command typically takes 1 second to execute and must not be interrupted. The controller returns a colon (:) when the Burn is complete.
This command reference will denote comands that can and cannot be burned with BN with the following usage icons.

Burnable with BN icon

Not burnable with BN icon

Arguments

The BN command has no arguments

Remarks

- The following table shows the commands that have their parameters saved with the BN command:

Parameters saved during burn

AC	CO	GA	LC	OP	TM
AF	CW	GM	LZ	OT	TR
BA	DC	GR	MO	OV	VA
BB	DV	HV	MT	PF	VD
BI	EI	IL	NB	PL	VF
BL	EO	IT	NF	PW	VS
BM	ER	KD	NZ	SB	YA
BO	FA	KI	OA	SP	YB
CE	FL	KP	OE	TK	YC
CN	FV	KS	OF	TL	

Operand Usage

- _BN contains the serial number of the processor board.

Examples

```
'Galil DMC Code Example
SB 1;'   Set bit 1
CB 2;'   Clear bit 2
CW 1;'   Set data adjustment bit
BN;' Burn all parameter states
```

BO *Brushless Offset*

BOm= n
BO n,n,n,n,n,n,n,n

Usage	BOm= n	Arguments specified with a single axis mask and an assignment (=)
	BO n ...	Arguments specified with an implicit, comma-separated order
Operands	_BOm	Operand holds the value last set by the command

Description

The BO command sets a fixed offset on the command signal for sinusoidally commutated motors. This may be used to offset any bias in the amplifier, or can be used for phase initialization.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	-5	5	0	20/65,536	Offset applied to DAC output in volts	

Remarks

External Sine Drive

- When using a third party, external sine drive, each motor axis requires two control axes. Therefore, for 4 axes of external sine control, an 8 axis controller is required.
- In this configuration, BO sets the offset for both DACs. Each member of a pair of axes has its own BO value.
- When measuring DAC output voltage, to assure that the output voltage equals the BO parameters, set the PID and OF parameters to zero.

Examples

```
'Galil DMC Code Example
'Assume a two axis controller
BA A;' BA allows the control of an external sine drive with the use of two axis. This is now a one axis controller.
' Axis B is used as the secondary DAC for axis A commutation.
'
BO -2,1;' Generates the DAC voltage -2 on the first DAC A, and 1 on the second DAC B of a sinusoidally commutated drive.
```

BO applies to DMC40x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

BP *Burn Program*

BP

Usage	BP	Command takes no arguments

Description

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

Arguments

The BP command has no arguments

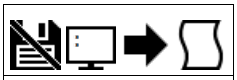
Remarks

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

Examples

```
'Galil DMC Code Example
:BP;'     Burn in program to controller
:'        Get colon response when done
```

BV *Burn Variables and Array*



BV

Usage	BV	Command takes no arguments
Operands	_BV	Operand has special meaning, see Remarks

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

The BV command has no arguments

Remarks

- _BV returns the number of controller axes.
- This command will store the ECAM table values in non-volatile EEPROM memory.
- This command may cause the Galil software to timeout. 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 period. The timeout can be changed in the Galil software. This warning does not affect the operation of the board or software.

Examples

```
'Galil DMC Code Example
:BV;'  burn in variables
:'     colon response returned
```

BZ *Brushless Zero*

```
BZm= n
BZ n,n,n,n,n,n,n,n
BZ <o
```

Usage	BZm= n	Arguments specified with a single axis mask and an assignment (=)
	BZ n ...	Arguments specified with an implicit, comma-separated order
Operands	_BZm	Operand has special meaning, see Remarks

Description

The BZ command is used for axes which are configured for sinusoidal commutation to initialize the motor at zero magnetic phase. To do this, the command drives the motor to zero magnetic phase and then sets the commutation phase to zero.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	-4.998	4.998	0	20/65,536	Voltage to be applied during amp initialization	-n = end BZ with SH. +n = end BZ with MO
o	100	32,767	1,000	1	Number of samples for BZ to hold torque.	o should be set before BZm= n command.

Remarks

- _BZm contains the distance in encoder counts from the motor's current position and the position of commutation zero for the specified axis.
  - This can useful to command a motor to move to the commutation zero position for phase initialization.
- This command may be given when the motor is off.
- The BZ command causes instantaneous movement of the motor. It is recommended to start with small voltages and increase as needed. The BZ command voltage must be large enough to move the motor.
- Always use the Off On Error function (OE command) to avoid motor runaway whenever testing sinusoidal commutation.

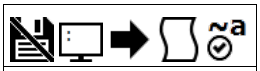
Examples

```
'Galil DMC Code Example
BZ , -3;' Drive B axis to zero phase with 3 volt signal, and end with motor enabled.
```

```
'Galil DMC Code Example
:BZ 2;'  Drive A axis to zero phase with 3V torque, and end with Motor off
```

BZ applies to DMC40x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

CA *Coordinate Axes*



CA m

Usage	CA mm	Argument is an axis mask
Operands	_CAm	Operand has special meaning, see Remarks

Description

The CA command specifies the coordinate system to apply proceeding vector commands. The following commands apply to the active coordinate system as set by the CA command:

CR	ES	LE	LI	LM
TN	VE	VM	VP	

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	S	T	S	Axis	Coordinate plane to specify	

Remarks

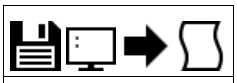
- CA ? returns a 0 if the S coordinate system is active and a 1 if the T coordinate system is active.
- _CA contains a 0 if the S coordinate system is active and a 1 if the T coordinate system is active.

Examples

```
'Galil DMC Code Example
CA T;' Specify T coordinate system
VM AB;' Specify vector motion in the A and B plane
VST= 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.
VE;' End Sequence
BG T;' Start motion of T coordinate system
```

CA applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC500x0

CB *Clear Bit*



CB n

Usage	CB n ...	Arguments specified with an implicit, comma-separated order

Description

The CB command clears a particular digital output. The SB and CB (Clear Bit) instructions can be used to control the state of output lines.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	1	16	N/A	1	General output bit to be set	Range 1-8 for 1-4 axis controllers
n	17	80	N/A	1	Extended I/O output bit to be set	Requires DB-14064 and I/O must be configured for outputs, see CO command

Remarks

- The state of the output can be read with the @OUT command

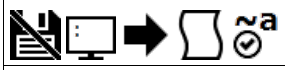
CB via Modbus Slave

Examples

```
'Galil DMC Code Example
#main
SB 5;'      Set digital output 5
SB 1;'      Set digital output 1
CB 5;'      Clear digital output 5
CB 1;'      Clear digital output 1
EN
```

For detailed information on connecting to a Modbus slave, see:
http://www.galilmc.com/techtalk/io-control/setting-up-and-rio-as-extended-io-for-a-controller/

CD *Contour Data*



CDm=n
CD n,n,n,n,n,n,n,n=t

Usage	CDm= n	Arguments specified with a single axis mask and an assignment (=)
	CD n ...	Arguments specified with an implicit, comma-separated order

Description

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

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	-32,768	32,767	0	1	Contour position segment	Incremental position move
t	1	8	0	1	Time override option	t = 1-8 specifies 2^n samples for the given interval.
	0	0	0	0	Time override option	t=0 with n=0 disables Contour mode. See Remarks
	-1	-1	0	0	Time override option	Pauses contour buffer at the segment with t=-1. Reissue DT to re-engage contour mode.

Remarks

- The units of the command are in encoder counts.
- The = operator can be used to override the global DT time by transmitting the time in a CD with the position data.
- n=t=0 terminates Contour mode similar to VE or LE for vector mode and linear interpolation mode.
  - Example. CMBC is terminated with CD 0,0=0.
  - The user must have a space after CD in order to terminate the Contour Mode correctly.
    - The command CD0=0 (no space) will assign a variable CD0 the value of 0 rather than terminate Contour mode.

Examples

```
'Galil DMC Code Example
#contour;'                  Program Label
CM AB;'                      Enter Contour Mode
DT 4;'                      Set time interval
CD 1000,2000;'              Specify data
CD 2000,4000;'              Next data
CD 0,0=0;'                  End of Contour Buffer
#wait;'                     Wait for all segments to process (buffer to empty)
WT 16,1;'                   wait for 1 DT time segment (2^4)
JP #wait,(_CM<>511)
EN;'                        End Program
```

```
'Galil DMC Code Example
#contour;'                  Program Label
CM A;'                      Enter Contour Mode
DT 4;'                      Set time interval
CD 1000;'                  Specify data
CD 2000;'                 Next data
CD 0=0;'                  End of Contour Buffer
#wait;'                     Wait for all segments to process (buffer to empty)
WT 16,1;'                   wait for 1 DT time segment (2^4)
JP #wait,(_CM<>31)
EN;'                        End Program
```

CD applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

CE *Configure Encoder*

CEm= n
CE n,n,n,n,n,n,n,n

Usage	CEm= n	Arguments specified with a single axis mask and an assignment (=)
	CE n ...	Arguments specified with an implicit, comma-separated order
Operands	_CEm	Operand holds the value last set by the command

Description

The CE command configures the encoder to quadrature type or 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.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	0	15	0	1	Encoder configuration setting	n is the sum of 2 integers M and N which configure main and auxiliary encoders. See table below for configuration description.

Configure Encoder Types. Add value from Column 1 and Column 2 to make n

Column 1	Main Encoder Type	Column 2	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 2 + 8, thus both encoders are reversed quadrature.

Remarks

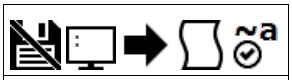
- When using a servo motor, changing the CE type can cause the motor to run away.
- When the MT command is configured for a stepper motor, the auxiliary encoder (used to count stepper pulses) will be forced to pulse and direction.
- When using pulse and direction encoders, the pulse signal is connected to CHA and the direction signal is connected to CHB.

Examples

```
'Galil DMC Code Example
:CE 0, 3, 6, 2;' Configure encoders
:CE ?,?,?,?;'     Interrogate configuration
0,3,6,2
:v = _CEB;'       Assign configuration to a variable
:v = ?
 3
:
```

CE applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

CM *Contour Mode*



CM mm

Usage	CM mm	Argument is an axis mask
Operands	_CM	Operand has special meaning, see Remarks

Description

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 specifies the position interval between subsequent contour segments. The DT command specifies the time interval between subsequent contour segments.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
mm	A	ABCDEFGH	N/A	Multi-Axis Mask	Axes to initialize to Contour mode	Disabled by default

Remarks

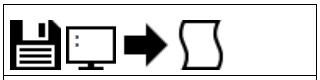
- mm = ? Returns a 0 if the contour buffer is full and 511 if the contour buffer is empty.
- _CM contains a '0' if the contour buffer is full; otherwise it contains the number of available contour segments.
- Issuing the CM command will clear the contour buffer when contour mode is not running.

Examples

```
'Galil DMC Code Example
#cont0;'                  Define label #cont0
CM ABCD;'                  Specify Contour Mode Axes ABCD
DT 4;'                    Specify time increment for contour (2^4 servo loops, 16ms at TM1000)
CD 200,350,-150,500;'     Specify incremental positions on A,B,C and D axes
'                         A-axis moves 200 counts B-axis moves 350 counts C-
'                         axis moves -150 counts D-axis moves 500 counts
'
CD 100,200,300,400;'      Next position data
CD 0,0,0,0=0;'            Special syntax to terminate Contour mode
#wait;JP #wait,_CM<>511;' Spin on #Wait label until buffer is empty
'                         End of Contour Buffer/Sequence
EN;'                      End program
'
'
#cont1;'                   Define label #cont1
CM ABC;'                   Specify Contour Mode
DT 8;'                     Specify time increment for contour (2^8 servo loops, 256ms at TM1000)
CD 100,100,100;'           New position data
CD 100,100,100;'           New position data
CD 0,0,0 =-1;'             Pause contour 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;'             Special syntax to terminate Contour mode
#wait2;JP #wait2,_CM<>511;'Spin on #wait2 label until buffer is empty
'End of Contour Buffer/Sequence
EN
```

CM applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

CN *Configure*



CN n_0,n_1,n_2,n_3,n_4

Usage	CN n ...	Arguments specified with an implicit, comma-separated order
Operands	_CN0 _CN1 _CN2 _CN3 _CN4	Operand holds the value last set by the command

Description

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

Arguments

Argument	Value	Description	Notes
n_0	1	Limit switches active high	
	-1	Limit switches active low	Default
n_1	1	HM will drive motor forward when Home input is high. See HM and FE commands.	
	-1	HM will drive motor backward when Home input is high. See HM and FE commands	Default
n_2	1	Latch input is active high	
	-1	Latch input is active low	Default
n_3	1	Configures inputs 5,6,7,8,13,14,15,16 as selective abort inputs for axes A,B,C,D,E,F,G,and H respectively.	Will also trigger #POSERR automatic subroutine if program is running.
	0	Inputs 5,6,7,8,13,14,15,16 are configured as general use inputs	Default
n_4	1	Abort input will not terminate program execution	
	0	Abort input will terminate program execution	Default

Remarks

- n_0 is useful for testing the operation of the #LIMSWI automatic subroutine. See example below.

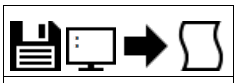
Examples

```
'Galil DMC Code Example
CN 1,1;'   Sets limit and home switches to active high
CN ,, -1;' Sets input latch active low
```

```
'Galil DMC Code Example
REM n0 is useful for testing the operation of the #LIMSWI automatic subroutine
#test
CN -1;'     Switches are active low
JGA= 100
BG A;'      Start a slow jog move
WT 1000
CN 1;'      Cause a limit fault by inverting the limit polarity
EN
'
#LIMSWI;'   Automatic sub will automatically launch on limit detection
MG "Limit Switch Routine"
WT 100
CN -1;'     Return to correct polarity
RE
```

CN applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

CO *Configure Extended I/O*



CO n

Usage	CO n ...	Arguments specified with an implicit, comma-separated order
Operands	_CO	Operand holds the value last set by the command

Description

The CO command configures which banks are inputs and which are outputs on the extended I/O. The extended I/O points of the controller can be configured in banks of 8.

CO only applies if the DB-14064 is present. The extended I/O is denoted as bits 17-80 and banks 2-9.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	0	255	0	1	Bitmask to configure extended IO as inputs or outputs	DB-14064 required. Bit=1 is outputs. Bit=0 is inputs. See Table below

CO Bitmask Description

Bit #	IO Bank	IO points
7	Bank 9	73-80
6	Bank 8	65-72
5	Bank 7	57-64
4	Bank 6	49-56
3	Bank 5	41-48
2	Bank 4	33-40
1	Bank 3	25-32
0	Bank 2	17-24

Remarks

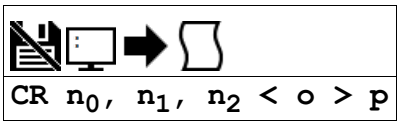
- CO only applies if the DB-14064 is present.

Examples

```
'Galil DMC Code Example
CO 255;' Configure all points as outputs
CO 0;'  Configure all points as inputs
CO 1;' Configures bank 2 to outputs on extended I/O
'See user manual appendix for more information on the extended I/O boards.
```

CO applies to DMC40x0,DMC42x0,DMC21x3,DMC18x6,DMC18x2,DMC500x0

CR *Circle*



CR n_0, n_1, n_2 < o > p

Usage	CR n ...	Arguments specified with an implicit, comma-separated order

Description

When using the vector mode (VM), the CR command specifies a 2-dimensional arc segment. 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. Parameters for radius, starting angle and traverse angle must all be entered for each CR command.

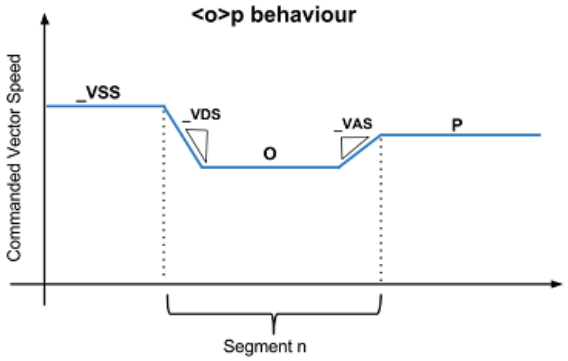
Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n_0	10	6,000,000	N/A	1	Radius of circle segment	
n_1	-32,000	32,000	N/A	1/65,536	Starting angle of circle segment	
n_2	-32,000	32,000	N/A	1/65,536	Degrees to traverse for circle segment	
o	2	22,000,000	N/A	2	Specifies the vector speed to be commanded at the beginning of the linear segment. The controller will start accelerating or decelerating at the start of the sequence to this speed.	For MT 1,-1,1.5,-1.5.
	2	6,000,000	N/A	2	Specifies the vector speed to be commanded at the beginning of the linear segment. The controller will start accelerating or decelerating at the start of the sequence to this speed.	For MT 2,-2,2.5,-2.5.
p	2	22,000,000	N/A	2	Specifies the vector speed to be achieved at the end of the linear segment. The controller will decelerate or accelerate during the segment and will reach the specified speed at the end of the segment.	For MT 1,-1,1.5,-1.5.
	2	6,000,000	N/A	2	Specifies the vector speed to be achieved at the end of the linear segment. The controller will decelerate or accelerate during the segment and will reach the specified speed at the end of the segment.	For MT 2,-2,2.5,-2.5.

Argument	Value	Description	Notes
o	-1	Specifies vector speed to be set by Vector Speed Variable (VV command)	See VV command

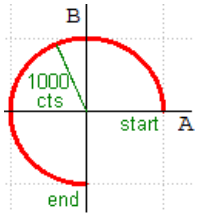
Remarks

- The product of n_0 * n_2 must be less than 450,000,000
- A positive n_2 denotes counterclockwise traverse, -n_2 denotes clockwise.
- n_0 units are in quadrature counts.
- n_1 and n_2 have units of degrees.



Examples

```
'Galil DMC Code Example
'A starting position of zero degrees denotes that the radius lies along
'a vector following the positive X axis, on a 2D Cartesian space:

VM AB
CR 1000,0,270
VE
BG S
EN

'The 2-d map out the position output can be seen below
```




```
'Galil DMC Code Example
VM AB;'                Specify vector motion in the A and B plane
VS 1000;'              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
'                 direction and use a vector speed of 40000.
VE;'               End Sequence
BG S;'              Start motion
```

```
'Galil DMC Code Example
'Generate a sine wave output on the A axis
VM AN;'             Specify vector motion in the A and N plane
VS 1000;'          Specify vector speed
CR 1000,0,360;'   Generate sine wave with amplitude of 1000 counts
'                  start at 0 degrees and complete one cycle
CR 1000,0,360<40000;' Generate same sine wave with same amplitude
'                  but run at faster speed (higher frequency)
'
VE;'               End Sequence
BG S;'              Start motion
```

CR applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

CS *Clear Sequence*

CS mm

Usage	CS mm	Argument is an axis mask
Operands	_CSm	Operand has special meaning, see Remarks

Description

The CS command will remove VP, CR or LI commands stored in a motion sequence for a coordinated axis. 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

Argument	Min	Max	Default	Resolution	Description	Notes
m	S	T	N/A	Axis	Coordinate plane specified to clear buffer	

Remarks

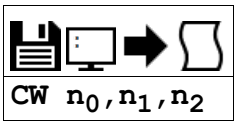
- _CSm contains the segment number in the sequence specified by m= S or T.
- This operand is valid in the Linear mode, LM, and Vector mode, VM.

Examples

```
'Galil DMC Code Example
#clear;' Label
CA T;' Specify the T coordinate system vector points
VP 1000,2000;' Vector Position
VP 4000,8000;' Vector Position
CS T;' Clear vectors specified in T coordinate system
CA S;' Specify the T coordinate system vector points
VP 1000,5000;' New vector
VP 8000,9000;' New vector
CS S;' Clear vectors specified in S coordinate system
```

CS applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

CW *Copyright information and Data Adjustment bit on/off*



Usage	CW n ...	Arguments specified with an implicit, comma-separated order

Description

The CW command will return the copyright information when the argument, n, is 0 or is omitted. Otherwise, the CW command is used as a communications enhancement for use by the Galil terminal software programs. When turned on, the most significant bit of unsolicited ASCII characters is set to 1. Unsolicited ASCII characters are characters that are returned from a program running on the controller (usually from the MG command). This command does not affect solicited characters, which are characters that are returned as a response to a command sent from a host PC (e.g. TP).

Arguments

Argument	Value	Description	Notes
n_0	0	Causes controller to return a copyright information string	Equivalent to n_0 = ?
	1	Controller will set the MSB of unsolicited message characters	
	2	Controller will not set the MSB of unsolicited message characters	Default. Must be set when viewing unsolicited messages from non-Galil software
n_1	0	Controller will pause program execution when output FIFO is full	Default
	1	Controller will continue program execution when output FIFO is full	Output characters will be lost if serial buffer is full.
n_2	0	Disable communication interrupts	Status bytes $BA, $BB, $BC
	1	Enable communication interrupts	Status bytes $BA, $BB, $BC

Remarks

- Galiltools automatically sends CW 1 during connection to a controller.
  - If also reading unsolicited data through a non-Galil software (eg. Hyperterminal), issue CW 2

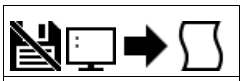
Operand Usage

- _CW contains the value set for n_0
- _CW4 contains the value set for n_1

Examples

```
'Galil DMC Code Example
CW 1;'     Set CW to Galil Driver mode (MSB set on unsolicited characters)

'         The CW command can cause garbled (non-ASCII) characters to be returned
'         by the controller when using third-party software. Use CW2.
CW 2;'     Set CW to third-party device mode (normal ASCII on unsoliticed characters)
```

DA *Deallocate Variables and Arrays*



```
DA str[],str
```

Usage	DA n ...	Arguments specified with an implicit, comma-separated order
Operands	_DAm	Operand has special meaning, see Remarks

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.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
str	1 char	8 chars	N/A	String	Array name to deallocate	If str = *, deallocate all arrays
	1 char	8 chars	N/A	String	Variable name to deallocate	If str = *, deallocate all variables

where
c[] - Defined array name
d - Defined variable name
d = * deallocates all the variables
c = *[] - Deallocates all the arrays
DA? Returns the number of arrays available.

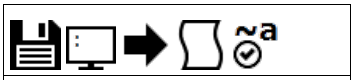
Remarks

- _DA contains the total number of arrays available.
- 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 a standard command.
- Variables and arrays that are deallocated are not set to zero. A routine that writes zeros to the array and/or variables should be created if this is desired.

Examples

```
'Galil DMC Code Example
'Cars' and 'Salesmen' are arrays, and 'Total' is a variable.
DM cars[40],salesmen[50];'        Dimension 2 arrays
total= 70;'                        Assign 70 to the variable Total
DA cars[0],salesmen[0],total;'    Deallocate the 2 arrays & variable
DA *[0];'                          Deallocate all arrays
DA *,*[0];'                       Deallocate all variables and all arrays
```

DC *Deceleration*



DCm= n

DC n,n,n,n,n,n,n,n

Usage	DCm= n	Arguments specified with a single axis mask and an assignment (=)
	DC n ...	Arguments specified with an implicit, comma-separated order
Operands	_DCm	Operand holds the value last set by the command

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

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	1,024	1,073,740,800	256,000	see Notes	Deceleration rate	See Remarks for resolution details

Remarks

- DC may be changed during a move in Jog mode, but not in a PA or PR move.
  - However, directly following an axis stop (ST m or a limit switch, #LIMSWI), the DC value of a PA or PR move may be changed while the axis is still decelerating.

Resolution

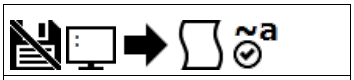
- The resolution of the DC command is dependent upon the update rate setting (TM). With the default rate of TM 1000 the resolution is 1024 cnts/second^2. The equation to calculate the resolution of the DC command is:
  - resolution = 1024*(1000/TM)^2
- example: With TM 500 the minimum DC setting and resolution is 4096 cnts/second^2
  - resolution = 1024*(1000/500)^2 = 4096

Examples

```
'Galil DMC Code Example
PR 10000;' Specify position
AC 2000000;' Specify acceleration rate
DC 1000000;' Specify deceleration rate
SP 5000;' Specify slew speed
BG ;'     Begin motion
```

DC applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

DE *Dual (Auxiliary) Encoder Position*



DEm= n

DE n,n,n,n,n,n,n,n

Usage	DEm= n	Arguments specified with a single axis mask and an assignment (=)
	DE n ...	Arguments specified with an implicit, comma-separated order
Operands	_DEm	Operand has special meaning, see Remarks

Description

The DE command defines the position of the auxiliary (dual) encoders.

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

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	-2,147,483,648	2,147,483,647	0	1	Position set for auxiliary encoders	For MT 1,-1,1.5,-1.5
	-2,147,483,648	2,147,483,647	0	1	Position set for main encoders	For MT 2,-2,2.5,-2.5

Remarks

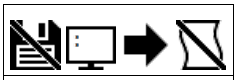
- When using stepper motors, the DE command defines the main encoder position.
- The auxiliary encoders are not available for the stepper axis or for any axis where output compare is active.
- The operand _DEm, as well as _TDm, holds the current aux encoder position.
- n=? will return the encoder position, as returned by TD.

Examples

```
'Galil DMC Code Example
DE 0,100,200,400;' Set the current auxiliary 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
```

DE applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

DL *Download*



DL n
DL #str
DL s

Usage	DL n ...	Arguments specified with an implicit, comma-separated order

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 \.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	0	3,999	0	1	Line number to begin program download	Firmware Rev 1.1a and later
n	0	1,999	0	1	Line number to begin program download	
str	1 char	8 chars	""	String	Name of label in RAM to begin download from.	
s	#	#	N/A	Symbol	Begins download at end of program in RAM	

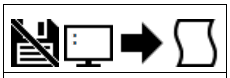
Remarks

- Do not insert spaces before each command.
- _DL gives the number of available labels (510 maximum)
- Maximum program dimensions are 2000 lines by 80 characters

Examples

```
'Galil DMC Code Example
:DL;'       Begin Download
#A;PR 4000;BGA
AMA;MG DONE
EN
\
:'End download
```

DM *Dimension Array*



```
DM str[n]
```

Usage	DM n ...	Arguments specified with an implicit, comma-separated order

Description

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

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
str	1 char	8 chars	N/A	String	Name of array to dimension	
n	1	16,000	N/A	1	Number of array elements to assign to dimensioned array	

where
c is a array name of up to eight alphanumeric characters, starting with an alphabetic character.
i is the number of array elements.
n = ? returns the number of array elements available.

Remarks

- The first character of str must be alphabetic. The rest can be any alphanumeric characters.
- When assigning array elements, the number specified must be less than the current available array space
- _DM contains the available array space.

Examples

```
'Galil DMC Code Example
DM pets[5],dogs[2],cats[3];' Define dimension of arrays, Pets with 5 elements, Dogs with 2 elements, Cats with 3 elements
DM tests[1600];' Define dimension of array Tests with 1600 elements
```

```
'Galil DMC Code Example
:DM ?
16000
:DM myarray[1000]
:DM ?
15000
'DMC-18x6 provide length of array with array[-1]
:MG "MyArray contains",myarray[-1]," elements"
MyArray contains 1000.0000 elements
:
```

DP *Define Position*

DPm= n

DP n,n,n,n,n,n,n,n

Usage	DPm= n	Arguments specified with a single axis mask and an assignment (=)
	DP n ...	Arguments specified with an implicit, comma-separated order
Operands	_DPm	Operand has special meaning, see Remarks

Description

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

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
	M	N	N/A	Axis	Virtual axis to assign value	
n	-2,147,483,648	2,147,483,647	0	1	Value assigned to motor/commanded position (RP and TP registers)	For MT 1,-1,1.5,-1.5
	-2,147,483,648	2,147,483,647	0	1	Value assigned to step/commanded position (RP and TD registers)	For MT 2,-2,2.5,-2.5

Remarks

- The DP command sets the commanded reference position for axes configured as steppers. 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. When equipped with an encoder, use the DE command to set the encoder position for stepper mode.
- The DP command is useful to redefine the absolute position.
  - For example, you can manually position the motor by hand using the Motor Off command, MO. Turn the servo motors back on with SH and then use DP0 to redefine the new position as your absolute zero.
- The operand _DPm, as well as _TPm, holds the current main encoder position.
- n=? will return the encoder position, as returned by TP.

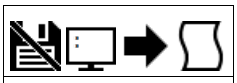
Examples

```
'Galil DMC Code Example
: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 A, C and D axes remain unchanged.
:DP ?,?,?,?;'          Interrogate the position of A, B, C and D axis.
0, -50000, 200, 400
:DP ?;'                Interrogate the position of A axis
0
:
```

```
'Galil DMC Code Example
:DP 0;'       Sets the current position of the A-axis to 0
:DP -50000;'  Sets the current position of A-axis to -50000.
:DP ?;'       Interrogate the position of A
-50000
```

DP applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

DR *Configures I O Data Record Update Rate*



DR n

Usage	DR n ...	Arguments specified with an implicit, comma-separated order
Operands	_DR	Operand has special meaning, see Remarks

Description

DR specifies and enables the rate for the controller to output its data record.

DR specifies the data update rate as 2n samples between updates. The controller creates a record and places it in the DPRAM location at this rate. See the User Manual for the data record map

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	0	8	0	1	Data update rate specified with a 2^n sample period	n = 0 turns off data record output.

Remarks

- If a small sample period and a small update rate is used, the controller may become noticeably slower as a result of maintaining a high update rate.
- _DR contains the data record update rate (n)

Examples

```
'Galil DMC Code Example
:WH
 IHA
:DR 8,0
GX~P
_`@~P
_H`~P
_0~P
:DR 0

'Note:  The data record is in a binary, non-printable format
'(the output above is normal when printing to the terminal)
```

DR applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,RIO,DMC18x6,DMC30010,DMC500x0

DT *Delta Time*

DT n

Usage	DT n ...	Arguments specified with an implicit, comma-separated order

Description

The DT command sets the time interval for Contour Mode. The time interval is 2^N samples. With TM 1000, there are 1024 samples per second. 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

Argument	Min	Max	Default	Resolution	Description	Notes
n	1	8	1	1	Set time interval for contour mode in 2^n samples.	
	-1	-1	N/A	0	n=-1 to pause the contour mode	See Remarks.

Remarks

- By default the sample period is 1 msec (set by the TM command); with n=1, the time interval would be 2 msec
- n = -1 allows a pre-load of the contour buffer or to asynchronously pause the contour buffer. DT-1 during contour mode will pause the contour buffer (and commanded movement).
- A positive DT will resume contour mode from paused position of buffer.
- DT can be overridden with the =t parameter within a CD segment.

Examples

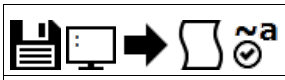
```
'Galil DMC Code Example
:DT 4;'              Specifies time interval to be 16 msec (TM1000)
:DT 7;'              Specifies time interval to be 128 msec
:
```

```
'Galil DMC Code Example
REM basic contour example
#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;'              Wait for all segments to process (buffer to empty)
WT 16,1;'            wait for 1 DT time segment (2^4)
JP #wait,(_CM<>511)
EN;'                 End program
```

```
'Galil DMC Code Example
REM contour example for pre-loading of contour buffer
#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;'              Wait for all segments to process (buffer to empty)
WT 16,1;'            wait for 1 DT time segment (2^4)
JP #wait,(_CM<>511)
EN;'                 End program
```

```
'Galil DMC Code Example
REM contour example for pre-loading of contour buffer
#cont1;'             Define label #Cont1
CM A;'               Specify Contour Mode
DT -1;'              Pause Contour Mode to allow pre-load of buffer
CD 100;'             Countour Data pre-loaded in buffer
CD 400;'             Countour Data pre-loaded in buffer
CD 200;'             Countour Data pre-loaded in buffer
CD 300;'             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;'             End of Contour Buffer/Sequence
#wait;'              Wait for all segments to process (buffer to empty)
WT 16,1;'            wait for 1 DT time segment (2^4)
JP #wait,(_CM<>31)
EN;'                 End program
```

DV *Dual Velocity (Dual Loop)*



DVm= n

DV n,n,n,n,n,n,n,n

Usage	DVm= n	Arguments specified with a single axis mask and an assignment (=)
	DV n ...	Arguments specified with an implicit, comma-separated order
Operands	_DVm	Operand holds the value last set by the command

Description

The DV function changes the operation of the PID filter to work off of dual encoders. DV enabled causes the KD (derivative) term to operate on the dual encoder instead of the main encoder. This results in improved stability in the cases where there is a backlash between the motor and the main encoder, and where the dual encoder is mounted on the motor.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	0	1	0	1	State of dual loop mode	n = 0 disables Dual loop. n = 1 enables Dual loop

Remarks

- The DV command is useful in backlash and resonance compensation.

Using DV with Large motor/load encoder ratio

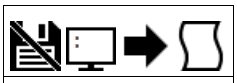
- When using Dual Loop mode with a large motor:load ratio and/or running at high velocities where low position error at speed is required, FV should be used to compensate for the derivative contribution from the higher resolution motor encoder.
  - The estimated FV setting required to compensate for the derivative contribution can be calculated by the equation:
    - FV = (KD/4)*(motor/load)
    - motor/load = effective motor to load ratio
  - For example: KD = 200, motor encoder changes 5000 counts per 1000 counts of load encoder (motor/load = 5/1)
    - FV = (200/4)*(5/1) = 250
- Ensure the motor encoder and load encoder count in the same direction to avoid dual loop positive feedback.
  - With motor off (MO) check the motor encoder with TD and load encoder with TP. Manually move the motor/load and reissue the TD and TP commands to confirm both encoders count in the same direction.
  - If the encoders count in opposing directions, change the polarity of one encoder using the CE command or by changing the wiring. Consult user manual.
  - Now the system has encoders with similar direction but a positive feedback situation may still exist. Off on error (OE) and error limits (ER) can be used to prevent a motor runaway condition. Positive feedback can be corrected by switching motor polarity or by reversing the direction of both encoders.

Examples

```
'Galil DMC Code Example
DV 1,1,1,1;' Enables dual loop on all axes
DV 0;' Disables DV on A axis
DV ,,1,1;' Enables dual loop on C axis and D axis.  Other axes remain unchanged.
DV 1,0,1,0;' Enables dual loop on A and C axis.  Disables dual loop on B and D axis.
MG _DVA;' Returns state of dual velocity mode for A axis
```

DV applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC30010,DMC500x0

EA *Choose ECAM master*



EA m

Usage	EA mm	Argument is an axis mask

Description

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

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign as ECAM master	
	M	N	N/A	Axis	Virtual axis to assign as ECAM master	N is default

Remarks

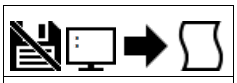
- The ECAM mode runs off of the master's main encoder (TP) even when the axis is running in stepper mode.
- When using the M or N imaginary axes, the commanded position is used.
- m=? will return the currently set ECAM master.

Examples

```
'Galil DMC Code Example
REM example using A axis as ECAM master and B axis as ECAM slave
#camone
master= 400
slave= 8192
EB 0; 'Disable ECAM Mode
EA A; 'Set Master Axis as A
EM master,slave
EP master/4,0
ET[0]= ,0
ET[1]= ,2048
ET[2]= ,4096
ET[3]= ,6144
ET[4]= ,8192
DP 0,0
SH AB
'NOTE - (EP Value)*(# of Cam Points) must be >= to Master Modulus
JG 100;BG A
EB 1
EG ,0; 'Start ECAM profile
EN
```

EA applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

EB *Enable ECAM*



EB n

Usage	EB n ...	Arguments specified with an implicit, comma-separated order
Operands	_EB	Operand has special meaning, see Remarks

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.

Arguments

Argument	Value	Description	Notes
n	0	Stop ECAM mode	Default
	1	Start ECAM mode	

Remarks

- When the EB command is given, the master axis position is modularized.
- _EB holds the enabled state, 1 or 0

Examples

```
'Galil DMC Code Example
EB 1;'  Starts ECAM mode
EB 0;' Stops ECAM mode
var = _EB;' Return status of cam mode
```

EB applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

EC *ECAM Counter*

EC n

Usage	EC n ...	Arguments specified with an implicit, comma-separated order
Operands	_EC	Operand has special meaning, see Remarks

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

Argument	Min	Max	Default	Resolution	Description	Notes
n	0	256	0	1	Set the ECAM table index	

Remarks

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

Examples

```
'Galil DMC Code Example
EC 0;'  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
var= _EC;'    Set the ECAM index value to a variable
```

EC applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

ED *Edit*

ED n

Usage	ED n ...	Arguments specified with an implicit, comma-separated order
Operands	_ED _ED1 _ED4	Operand has special meaning, see Remarks

Description

The ED command puts the controller into the Edit subsystem. The ED command is used when using Telnet style interface (not Galil Software). In the Edit subsystem, programs can be created, changed, or destroyed.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	0	3,999	see Notes	1	Line number to begin editing	Firmware Rev 1.1a and later. Default n is the last line of program space with commands.
n	0	1,999	see Notes	1	Line number to begin editing	Default n is the last line of program space with commands.

Remarks

- The commands in the Edit subsystem are the following.

ED Commands

Key Combination	Function
<ctrl>D	Deletes a Line
<ctrl>I	Inserts a line before the current
<ctrl>P	Displays the previous line
<ctrl>Q	Exits the ED subsystem
Enter	Saves a line and moves cursor to next

Operand Usage

- _ED0 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).
- _ED0 returns 0 if no error has occurred.
- _ED1 returns -1 if no error has occurred.

- _ED4 when evaluated in an embedded code thread, this operand will contain the thread id of the calling thread. This is useful for DMC code to determine which thread it is running in. See example below.

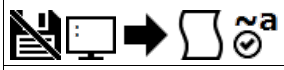
Examples

```
'Galil DMC Code Example
:ED
#START
PR 2000
BGA
xx;' bad command line
EN
#CMDERR Routine which occurs upon a command error
V=_ED0
MG "An error has occurred" {n}
MG "In line", V{F3.0}
ST
ZS0
EN
ctrl-Q
:'Hint:  Remember to quit the Edit Mode prior to executing or listing a program.
```

```
'Galil DMC Code Example
'Using _ED4
XQ #id,1
XQ #id,2
XQ #id,3
XQ #id,4
XQ #id,5
XQ #id,6
XQ #id,7
#id
MG {Z10.0}"This message is from thread",_ED4
EN

' Returns...
' :XQ
' This message is from thread 1
' This message is from thread 2
```

```
' This message is from thread 3
' This message is from thread 4
' This message is from thread 5
' This message is from thread 6
' This message is from thread 7
' This message is from thread 0
```

EG *ECAM go (engage)*



EGm= n

EG n,n,n,n,n,n,n,n

Usage	EGm= n	Arguments specified with a single axis mask and an assignment (=)
	EG n ...	Arguments specified with an implicit, comma-separated order
Operands	_EGm	Operand has special meaning, see Remarks

Description

The EG command engages an ECAM slave axis at a specified position of the master. Once a slave motor is engaged, its position is redefined to fit within the cycle.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	-2,147,483,648	2,147,483,647	0	1	Master position to engage ECAM slave	n = outside of master axis position range causes slave to engage immediately.

Remarks

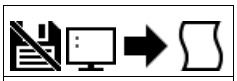
- _EGm contains ECAM status for specified slave axis. 0 = axis is not engaged, 1 = axis is engaged.
- n = ? Returns 1 if specified axis is engaged and 0 if disengaged.
- This command is not a trippoint. This command will not hold the execution of the program flow. If the execution needs to be held until master position is reached, use MF or MR command.

Examples

```
'Galil DMC Code Example
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
```

EG applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

EI *Event Interrupts*



EI n_1,n_2

Usage	EI n ...	Arguments specified with an implicit, comma-separated order
Operands	_EI	Operand has special meaning, see Remarks

Description

The EI command is used to enable interrupts on events. EI enables interrupts for the predefined event conditions in the table below. When a condition (e.g. Axis A profiled motion complete) occurs after EI is armed, a particular status byte value (e.g. $D0 or 208) is delivered to the host PC along with the interrupt.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n_1	0	65,535	0	1	16-bit interrupt mask	0 turns off interrupts. See Remarks for bit mask
n_2	0	255	0	1	8-bit input mask	Used to select the specific digital input trigger. Bit 15 of n_1 must be set for the n_2 mask to be used.

Remarks

- _EI contains the interrupt mask n_1
- n_1 = 0 means "don't interrupt" and clears the queue when issued
- The interrupts marked with * in the table below must be re-enabled with EI after each occurrence
- Bit 15 of n_1 must be set for the n_2 input mask to be used

n_1 Bit Mask

Interrupt Bits

bit	n_1=2^bit Hex (decimal)	Status Byte Hex (decimal)	Condition
0	$0001 (1)	$D0 (208)	Axis A profiled motion complete _BGA = 0
1	$0002 (2)	$D1 (209)	Axis B profiled motion complete _BGB = 0
2	$0004 (4)	$D2 (210)	Axis C profiled motion complete _BGC = 0
3	$0008 (8)	$D3 (211)	Axis D profiled motion complete _BGD = 0
4	$0010 (16)	$D4 (212)	Axis E profiled motion complete _BGE = 0
5	$0020 (32)	$D5 (213)	Axis F profiled motion complete _BGF = 0
6	$0040 (64)	$D6 (214)	Axis G profiled motion complete _BGG = 0
7	$0080 (128)	$D7 (215)	Axis H profiled motion complete _BGH = 0
8	$0100 (256)	$D8 (216)	All axes profiled motion complete _BGI = 0
9	$0200 (512)	$C8 (200)	* Excess position error _TEn >= _ERn
10	$0400 (1024)	$C0 (192)	* Limit switch _LFn = 0 Must be profiling motion in direction of activated limit switch for interrupt to occur.
11	$0800 (2048)	$D9 (217)	Watchdog timer
12	$1000 (4096)		Reserved
13	$2000 (8192)	$DB (219)	Application program stopped _XQn = -1
14	$4000 (16384)	$DA (218)	PC command done, colon response sent
15	$8000 (32768)	$E1-$E8 (225-232)	* Digital input(s) 1-8 low (use n_2 for mask)
	UI, user interrupt command	$F0-$FF (240-255)	User Interrupt, See UI command

n_2 Bit Mask

Input Interrupts

bit	n_2 = 2^bit hex (decimal)	Status Byte hex (decimal)	Condition
0	$01 (1)	$E1 (225)	* Digital input 1 is low @IN[1] = 0
1	$02 (2)	$E2 (226)	* Digital input 2 is low @IN[2] = 0
2	$04 (4)	$E3 (227)	* Digital input 3 is low @IN[3] = 0
3	$08 (8)	$E4 (228)	* Digital input 4 is low @IN[4] = 0
4	$10 (16)	$E5 (229)	* Digital input 5 is low @IN[5] = 0
5	$20 (32)	$E6 (230)	* Digital input 6 is low @IN[6] = 0
6	$40 (64)	$E7 (231)	* Digital input 7 is low @IN[7] = 0
7	$80 (128)	$E8 (232)	* Digital input 8 is low @IN[8] = 0

Examples

```
'Galil DMC Code Example
```

```
'Interrupt when motion is complete on all axes OR if a limit switch is hit:
'From the table, enable bits 8 and 10.  n1 = 256 + 1024 = 1280
EI  1280
'
'Interrupt when digital input 3 is low.
'Enable bit 15 of n1 and bit 2 of n2.
EI 32768,4
```

EI applies to DMC40x0,DMC42x0,DMC41x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

ELSE *Else function for use with IF conditional statement*

Usage	ELSE n ...	Arguments specified with an implicit, comma-separated order

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 is a command with no parameters

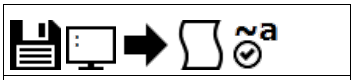
Remarks

- None

Examples

```
'Galil DMC Code Example
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 "IN1 AND IN2 ARE ACTIVE";' Message to be executed if 2nd IF conditional is true
 ELSE;'                         ELSE command for 2nd IF conditional statement
  MG "ONLY IN1 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
 IF (@IN[2]=0);'               3rd IF conditional statement executed if 1st IF conditional false
  MG "ONLY IN2 IS ACTIVE";'   Message to be executed if 3rd IF conditional statement is true
 ELSE;'                        ELSE command for 3rd conditional statement
  MG "IN1 AND IN2 INACTIVE";'Message to be executed if 3rd IF conditional statement is false
 ENDIF ;'                       End of 3rd conditional statement
ENDIF ;'                        End of 1st conditional statement
```

EM *Ecam modulus*



EMm= n

EM n,n,n,n,n,n,n,n

Usage	EMm= n	Arguments specified with a single axis mask and an assignment (=)
	EM n ...	Arguments specified with an implicit, comma-separated order
Operands	_EMm	Operand holds the value last set by the command

Description

The EM command defines 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.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	2	8,388,607	N/A	1	Position change over one full ECAM cycle	For defining master axis
	0	2,147,483,647	N/A	1	Position change over one full ECAM cycle	For defining slave axis

Remarks

- If a slave will return to its original position at the end of the cycle, then n=0.
- If the change is negative, specify the absolute value for n.

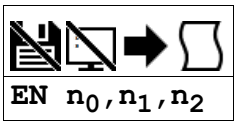
Examples

```
'Galil DMC Code Example
REM example using A axis main encoder as master B axis main encoder as the slave
#cam
REM define A axis encoder as master for ECAM
EA A
REM
REM EM command options
REM ----
REM define slave modulus as 0 (returns to original position)
REM and define master modulus as 4000
EM 4000,0
REM
REM another valid EM settings for this configuration
REM ----
'EMA= 4000;' define A axis master modulus as 0
'EMB= 0;'    define B axis slave modulus as 0
REM
REM ----
REM define master increment as 1000 counts/table entry
EP 1000
ET[0]= , 0
ET[1]= , 1000
ET[2]= , 2000
ET[3]= , 1000
ET[4]= , 0
REM enable ECAM mode
EB 1
REM engage when master is at 0 position
EG 0,0
EN
```

```
'Galil DMC Code Example
EA C;' 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
```

EM applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

EN *End*



Usage	EN n ...	Arguments specified with an implicit, comma-separated order

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.

A return parameter can be specified to EN from a subroutine to return a value from the subroutine to the calling stack.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n_0	0	1	0	1	Specify trippoint status when returning from subroutine	n_0=1 restores trippoints. n_0=0 does not restore trippoints
n_1	0	1	0	1	Set status of CI interrupt when returning from #COMINT	Parameter is not used on DMC-18x6.
n_2	-2,147,483,648	2,147,483,647	0	1	Return a value from a subroutine.	Accesible from the calling program with _JS. See JS for more information

Remarks

- The EN command is used to end the automatic subroutines #MCTIME and #CMDERR.
  - Use the RE command to end the #POSERR and #LIMSWI subroutines.
  - Use the RI command to end the #ININT subroutine

Examples

```
'Galil DMC Code Example
#a;'     Program A
PR 500;' Move A axis forward 500 counts
BG A;'    Begin motion
AM A;'    Pause the program until the A axis completes the motion
EN;'     End of Program
```

```
'Galil DMC Code Example
#example
'test program showing restoring trippoints with EN
XQ #err,1;'   Execute thread to generate error
AI 1;'        Wait for input 1 to trigger
MG "hello";'  After input, message out
EN

#err
'dummy thread that runs to cause an error
XX123;'      Invalid command
'causes CMDERR to be called, interrupting thread 0
EN

#CMDERR
'error subroutine running on thread 0
tc= _TC;'     Save error code
EN 1;'        End routine, restore AI trippoint.
```

ENDIF *End of IF conditional statement*

Usage	ENDIF n ...	Arguments specified with an implicit, comma-separated order

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 is a command with no parameters

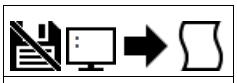
Remarks

- None

Examples

```
'Galil DMC Code Example
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 "IN1 AND IN2 ARE ACTIVE";' Message to be executed if 2nd IF conditional is true
 ELSE;'                         ELSE command for 2nd IF conditional statement
  MG "ONLY IN1 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
 IF (@IN[2]=0);'               3rd IF conditional statement executed if 1st IF conditional false
  MG "ONLY IN2 IS ACTIVE";'   Message to be executed if 3rd IF conditional statement is true
 ELSE;'                        ELSE command for 3rd conditional statement
  MG "IN1 AND IN2 INACTIVE";'Message to be executed if 3rd IF conditional statement is false
 ENDIF ;'                       End of 3rd conditional statement
ENDIF ;'                        End of 1st conditional statement
```

EO *Echo*



EO n

Usage	EO n ...	Arguments specified with an implicit, comma-separated order
Operands	_EO	Operand holds the value last set by the command

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

Argument	Value	Description	Notes
n	0	Echo Off	
	1	Echo On	Default

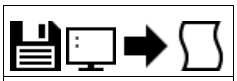
Remarks

- This command is defaulted to EO1. Galil software upon connection will set EO0

Examples

```
'Galil DMC Code Example
EO 0;' Turns echo off
EO 1;' Turns echo on
```

EP *Cam table master interval and phase shift*



EP n_0,n_1

Usage	EP n ...	Arguments specified with an implicit, comma-separated order
Operands	_EP	Operand holds the value last set by the command

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. Up to 257 points may be specified.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n_0	1	32,767	256	1	Master position interval	Cannot be changed while ECAM is running
n_1	-2,147,483,648	2,147,483,647	0	1	ECAM table phase shift	Can be modified during ECAM

Remarks

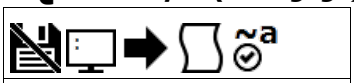
- _EP contains the value of the interval n_0.
- The offset parameter 'n_1' can also be used to instantaneously phase shift the graph of the slave position verses the master position. This can be used to make on-the-fly corrections to the slaves.
  - See application note #2502 for more details. http://www.galilmc.com/support/application-notes.php

Examples

```
'Galil DMC Code Example
EP 20;'           Sets the cam master points to 0,20,40 . . .
d =  _EP;'        Set the variable d equal to the ECAM internal master interval
EP ,100;'          Phase shift all slaves by 100 master counts
```

EP applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

EQ *ECAM quit (disengage)*



EQm= n
EQ n,n,n,n,n,n,n,n

Usage	EQm= n	Arguments specified with a single axis mask and an assignment (=)
	EQ n ...	Arguments specified with an implicit, comma-separated order
Operands	_EQm	Operand has special meaning, see Remarks

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

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	-2,147,483,648	2,147,483,647	N/A	1	Master position to disengage the slave axis specified.	If n = outside of master position range, disengage slave axis immediately.

Remarks

- _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.
- 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.
- 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.

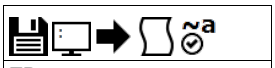
Examples

```
'Galil DMC Code Example
EQ 300,700;' Disengages the A and B motors at master positions 300 and 700 respectively.
```

```
'Galil DMC Code Example
EQ 300;' Disengages the A motor at master position 300.
```

EQ applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

ER *Error Limit*



ERm= n

ER n,n,n,n,n,n,n,n

Usage	ERm= n	Arguments specified with a single axis mask and an assignment (=)
	ER n ...	Arguments specified with an implicit, comma-separated order
Operands	_ERm	Operand holds the value last set by the command

Description

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

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	-1	2,147,483,647	16,384	1	Set the position error limit in counts	n=0 enables Error output. n=-1 disables Error output.

Remarks

- The error limit specified by ER should be high enough as not to be reached during normal operation.
  - Examples of exceeding the error limit would be a mechanical jam, or a fault in a system component such as encoder or amplifier
- For debugging purposes, ER0 and ER-1 can be used to turn the red LED on and off.

Examples

```
'Galil DMC Code Example
:ER 200,300,400,600;' Set the A-axis error limit to 200, the B-axis error limit to 300, the C-axis error limit to 400, and the D-axis error limit to 600.
:ER ,1000;' Sets the B-axis error limit to 1000, leave the A-axis error limit unchanged.
:ER ?,?,?,?;' Return A,B,C and D values
00200,00100,00400,00600
:ER ?;' Return A value
00200
:v1= _ERA;' Assigns V1 value of ERA
:v1= ;' Returns V1
00200
```

ER applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

ES *Ellipse Scale*

ES n_0,n_1

Usage	ES n ...	Arguments specified with an implicit, comma-separated order

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 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

Argument	Min	Max	Default	Resolution	Description	Notes
n_0	1	65,535	1	1	First value used for resolution scaling	See Remarks for usage
n_1	1	65,535	1	1	Second value used for resolution scaling	See Remarks for usage

Remarks

- For VM xy
  - When $n_0 > n_1$, the resolution of x will be multiplied by n_0/n_1
  - When $n_0 < n_1$, the resolution of y will be multiplied by n_1/n_0
- The ES command will apply to the selected coordinate system, S or T. To select the coordinate system, use the command CAS or CAT.

Examples

```
'Galil DMC Code Example
VM AB;ES 3,4;' Divide B resolution by 4/3
VM CA;ES 2,3;' Divide A resolution by 3/2
VM AC;ES 3,2;' Divide A Resolution by 3/2
'Note:  ES must be issued after VM.
```

```
'Galil DMC Code Example
VM AN;ES 3,2;' Divide A Resolution by 3/2
'Note:  ES must be issued after VM.
```

ES applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

ET *Electronic cam table*

ET n,n,n,n,n,n,n,n
ET[n_0] = n,n,n,n,n,n,n,n

Usage	ET n ...	Arguments specified with an implicit, comma-separated order

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

Argument	Min	Max	Default	Resolution	Description	Notes
n_0	0	256	N/A	1	Index of the ECAM table entry	
n	-2,147,483,648	2,147,483,647	0	1	Position of the slave axis at the specified table point.	

Remarks

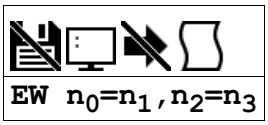
- [n_0] can be omitted only if EC has initialized the index count. In this case, each ET command will increment the index counter by 1.
- n=? Returns the slave position for the specified point.

Examples

```
'Galil DMC Code Example
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
EC 0;'               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
```

ET applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

EW *ECAM Widen Segment*



Usage	EW n ...	Arguments specified with an implicit, comma-separated order
Operands	_EW0 _EW1 _EW2 _EW3	Operand has special meaning, see Remarks

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.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n_0	1	255	-1	1	Index of first widened segment	If n_0 = -1, no segment is widened
n_1	1	2,147,483,647	0	1	Length of first widened segment	In master counts
n_2	3	255	-1	1	Index of second widened segment	If n_2 = -1, no segment is widened. n_2 must be > n_0
n_3	1	2,147,483,647	0	1	Length of second widened segment	In master counts

Remarks

- Remember that the widened segment lengths must be taken into account when determining the modulus (EM) for the master.
- The second widened segment cannot be used unless the first widened segment is also being used.
- The segments chosen should not be the first or last segments, or consecutive segments.

Operand Usage

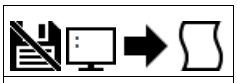
- _EW0 contains n_0, the index of the first widened segment.
- _EW1 contains n_1, the length of the first widened segment.
- _EW2 contains n_2, the index of the second widened segment
- _EW3 contains n_3, the length of the second widened segment.

Examples

```
'Galil DMC Code Example
EW 41=688;'            Widen segment 41 to 688 master counts
EW 41=688, 124=688;'   Widen segments 41 and 124 to 688 master counts
```

EW applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

EY *ECAM Cycle Count*



EY n

Usage	EY n ...	Arguments specified with an implicit, comma-separated order
Operands	_EY	Operand holds the value last set by the command

Description

The EY command 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.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	-2,147,483,648	2,147,483,647	0	1	Current ECAM cycle count	

Remarks

- _EY returns the current cycle count
- EY can be used to calculate the absolute position of an axis with the following equation:
  - Absolute position = EY * EM + TP

Examples

```
'Galil DMC Code Example
MG _EY * _EMB + _TPB;'          print absolute position of master (Y)
```

EY applies to DMC40x0,DMC42x0,DMC41x3,DMC18x6,DMC30010,DMC500x0

FA *Acceleration Feedforward*

FAm= n

FA n,n,n,n,n,n,n,n

Usage	FAm= n	Arguments specified with a single axis mask and an assignment (=)
	FA n ...	Arguments specified with an implicit, comma-separated order
Operands	_FAm	Operand holds the value last set by the command

Description

The FA command sets the acceleration feedforward coefficient. This coefficient is scaled by the set acceleration and adds a torque bias voltage during the acceleration phase and subtracts the bias during the deceleration phase of a motion.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	0	8,191	0	1/4	Value of proportional term	

Remarks

- The Feedforward Bias product is limited to 10 Volts.
- If the feedforward coefficient is changed during a move, then the change will not take effect until the next move.
- FA operates on PA, PR, IP, JG and PVT mode.
- FA does not operate in:
  - Contour Mode (CM)
  - Axis is Gearing or ECAM slave
  - Coordinated motion (LM, VM)

- Acceleration Feedforward Bias = FA * AC * (1.5 10-7) * ((TM/1000)^2)
- Deceleration Feedforward Bias = FA * DC * (1.5 10-7) * ((TM/1000)^2)
- FA is enabled during the PVT mode of motion.

Examples

```
'Galil DMC Code Example
'Set feedforward coefficient to 10 for the A-axis
'and 15 for the B-axis.  The effective bias will
'be 0.75V for A and 2.25V for B.

:AC 500000,1000000
:FA 10,15
:MG _FAA,_FAB
 10 15
```

FA applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

FE *Find Edge*

FE mm

Usage	FE mm	Argument is an axis mask

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

Argument	Min	Max	Default	Resolution	Description	Notes
mm	A	ABCDEFGH	N/A	Multi-Axis Mask	Axes to Find Edge	

Remarks

- 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
- Speed of Find Edge is set with the SP command and should be low enough to allow for a minimum of a 2 sample period pulse width on the home signal. With TM 1000, the pulse width must be at least 2ms.

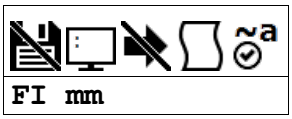
Examples

```
'Galil DMC Code Example
:FE ;' Set find edge mode
:BG ;' Begin find edge
:FE A;' Only find edge on A
:BG A
```

```
'Galil DMC Code Example
:FE B;' Only find edge on B
:BG B
:FE CD;' Find edge on C and D
:BG CD
```

FE applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

FI *Find Index*



Usage	FI mm	Argument is an axis mask

Description

The FI and BG commands move the motor until an encoder index pulse is detected.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
mm	A	ABCDEFGH	N/A	Multi-Axis Mask	Axes to Find Index	

Remarks

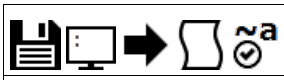
- The controller looks for a transition from low to high. There are 2 stages to the FI command. The first stage jogs the motor at the speed and direction of the JG command until a transition is detected on the index line. When the transition is detected, the position is latched and the motor will decelerate to a stop. In the second stage, the motor will reverse direction and move to the latched position of the index pulse at the speed set by the HV command. At the conclusion of FI, the position is defined as zero.
- 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.

Examples

```
'Galil DMC Code Example
#home;'             Home Routine
JG 500;'            Set speed and forward direction
FI A;'               Find index
BG A;'               Begin motion
AM A;'               After motion
MG "FOUND INDEX";' Print message
EN
```

FI applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

FL *Forward Software Limit*



FLm= n
FL n,n,n,n,n,n,n,n

Usage	FLm= n	Arguments specified with a single axis mask and an assignment (=)
	FL n ...	Arguments specified with an implicit, comma-separated order
Operands	_FLm	Operand has special meaning, see Remarks

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.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	-2,147,483,648	2,147,483,647	2,147,483,647	1	Value of software forward limit	2147483647 turns off forward limit

Remarks

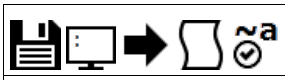
- The forward limit is activated at n+1. n = 2147483647 effectively disables the forward soft limit.
- The software limit is specified in counts for a servo system or in microsteps for a stepper system.
- When the forward software limit is activated, the automatic subroutine #LIMSWI will be executed if it is included in the program.
- If motion is commanded when the axis is already passed the FL value, the axis will profile a small move before the software limit is again detected.
  - This is typically encountered when commanding motion in loops, such as a jog loop.
  - In these scenarios it is recommended to use the #LIMSWI routine to stop the loop when the FL limit has been exceeded.

Examples

```
'Galil DMC Code Example
#test;'  Test Program
AC 1000000;' Acceleration Rate
DC 1000000;' Deceleration Rate
FL 15000;' Forward Limit
JG  5000;' Jog Forward
BG A;'    Begin
AM A;'    After Limit
RP A;'    Tell Position
EN;'     End

'Hint:  Galil controllers also provide hardware limits.
```

FL applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

FV *Velocity Feedforward*



FVm= n

FV n,n,n,n,n,n,n,n

Usage	FVm= n	Arguments specified with a single axis mask and an assignment (=)
	FV n ...	Arguments specified with an implicit, comma-separated order
Operands	_FVm	Operand holds the value last set by the command

Description

The FV command sets the velocity feedforward coefficient. This coefficient generates an output bias signal in proportions to the sample to sample change in reference position (RP).

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	0	8,191	0	1	Value of proportional term	

Remarks

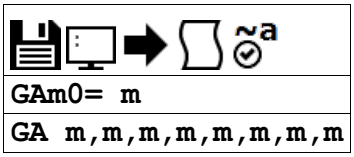
- FV also applies to Contour Mode (CM) and in gearing when an axis is a slave
- Velocity feedforward bias = FV * (Velocity [cts/s]) * (1.22 10-6) * (TM/1000)
  - With FVA=10, TM 1000 and the velocity is 200,000 count/s, the velocity feedforward bias equals 2.44 volts

Examples

```
'Galil DMC Code Example
'Set feedforward coefficients to 10 and 20 for A and B respectively.
'This effective bias will be 0.366 volts for A and 1.95 volts for B.

:FV 10,20
:JG 30000,80000
:MG _FVA,_FVB
 10  20
```

FV applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

GA *Master Axis for Gearing*



GAm0= m
GA m,m,m,m,m,m,m,m

Usage	GAm0= m	Arguments are single axis masks and are specified with a single axis mask and an assignment (=)
	GA m ...	Arguments are single axis masks specified with an implicit, comma-separated order

Description

The GA command specifies the master axes for electronic gearing. Multiple masters for gearing may be specified. A slave axis may have only one master. 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

Argument	Min	Max	Default	Resolution	Description	Notes
m0	A	H	N/A	Axis	Slave axis to assign master	m0<>m
m	A	H	N/A	Axis	Master axis main encoder as the slave's master	
	CA	CH	N/A	Axis	Master axis commanded position as the slave's master	Valid arguments: CA,CB,CC,CD,CE,CF,CG,CH
	DA	DH	N/A	Axis	Master axis aux encoder as the slave's master	Valid arguments: DA,DB,DC,DD,DE,DF,DG,DH
	S	T	N/A	Axis	Vector plane as the slave's master	
	M	N	N/A	Axis	Virtual axis as the slave's master	

Remarks

- m=? returns the GA setting
- When the geared motors must be coupled "strongly" to the master, use the gantry mode GM.
- When gearing is used in a gantry application, gearing off of the commanded position is recommended.

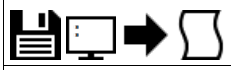
Examples

```
'Galil DMC Code Example
REM setup gearing where B axis is master for A and C axes.
#gear
MO B;'            Turn off servo to B motor
GA B,,B;'         Specify master axis as B on A and C
GR .25,,-5;'      Specify A and C gear ratios
SH B;'            Enable B axis
PRB= 1000;BG B;' Move B axis 1000 counts
'                 A axis will be commanded to move 250 counts positive
'                 C axis will be commanded to move -5000 counts
EN;'              End program
```

```
'Galil DMC Code Example
REM imaginary axis example
#imag
GAA= N;'          set the imaginary N axis as the master of the A axis
GRA= 2.5;'        set the gear ratio for the A axis as 2.5
PRN= 1000;BG N;' Move N axis 1000 counts
'                  (C axis will be commanded to move 2500 counts positive)
EN;'              End Program
```

GA applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

GD *Gear Distance*



GD n,n,n,n,n,n,n,n
GDm= n

Usage	GD n ...	Arguments specified with an implicit, comma-separated order
Operands	_GDm	Operand holds the value last set by the command

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.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	0	32,767	0	1	Absolute Value of Gearing Distance	0 engages gearing instantly

Remarks

- 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.

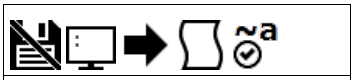
Examples

```
'Galil DMC Code Example
#a
GA ,A;'   Sets the A axis as the gearing master for the B axis
GD ,5000;' Set distance over which gearing is engaged to 5000 counts of the master axis.
JG 5000;' Set the A axis jog speed to 5000 cts/sec
BG A;'    Begin motion on the A axis
AS A;'    Wait until A axis reaches the set speed of 5000 counts/sec
GR ,1;'   Engage gearing on the B axis with a ratio of 1:1, the
'distance to fully engage gearing will be 5000 counts of the master axis
WT 1000;' Wait 1 second
GR ,3;'   Set the gear ratio to three.  The ratio will be changed
'over the distance set by the GD command
WT 1000;' Wait 1 second
GR ,0;'   Disengage the gearing between the B 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
```

```
'Galil DMC Code Example
#a
GA DA;'   Set the aux encoder input as the gearing master
GD 5000;' Set distance over which gearing is engaged to 5000 counts of the master axis.
GR 1;'    Set a gear ratio of 1:1, the distance to fully
'engage gearing will be 5000 counts of the master axis
WT 1000;' Wait 1 second
GR 3;'    Set the gear ratio to three.  The ratio will be changed
'over the distance set by the GD command
WT 1000;' Wait 1 second
GR 0;'    Disengage the gearing between the axis aux encoder
'The gearing will be disengaged over the number of
'counts of the master specified with the GD command above
EN;'      End program
```

GD applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

GM *Gantry mode*



GMm= n
GM n,n,n,n,n,n,n,n

Usage	GMm= n	Arguments specified with a single axis mask and an assignment (=)
	GM n ...	Arguments specified with an implicit, comma-separated order
Operands	_GMm	Operand holds the value last set by the command

Description

The GM command specifies the axes in which the gearing function is performed in the Gantry mode. In this mode, the geared slaves will not be stopped by the ST command or by limit switches.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	0	1	0	1	Value of GM command	1 Enables Gantry Mode, 0 disables Gantry Mode

Remarks

- The GM command is useful for driving heavy loads on both sides with two motors (Gantry Style)
- Only setting Gantry Mode of the slave to 0 (GMm= 0) will disable Gantry Mode

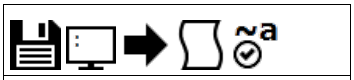
Examples

```
'Galil DMC Code Example
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
```

```
'Galil DMC Code Example
GA DA;' Set master for A axis to the A axis Aux encoder input
GM 1;'  Enable Gantry Mode on A axis
GR 1;'  Set Gear Ratio to 1
WT 1000
ST ;'    Axis will still be in gearing Mode
WT 1000
GM 0;'  Disable Gantry Mode (Axis still gearing)
WT 1000
ST ;'    Will clear gearing mode
EN
```

GM applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

GR *Gear Ratio*



GRm= n
GR n,n,n,n,n,n,n,n

Usage	GRm= n	Arguments specified with a single axis mask and an assignment (=)
	GR n ...	Arguments specified with an implicit, comma-separated order
Operands	_GRm	Operand holds the value last set by the command

Description

GR specifies the Gear Ratios for the geared axes in the electronic gearing mode. The master axis is defined by the GA command.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Slave axis to assign gear ratio	
n	-127	127	0	1/65,536	Value of Gear Ratio of Slave	n = 0 disables gearing

Remarks

- The gear ratio may be different for each geared axis.
- The master can go in both directions.
- When the geared motors must be coupled "strongly" to the master, use the gantry mode GM.
- Unless the GM command is set to 1, gearing is disabled in the following conditions:
  - The gear ratio is set to 0
  - A limit switch is reached
  - The axis is commanded to stop with the ST command

Examples

```
'Galil DMC Code Example
REM setup gearing where B axis is master for A and C axes.
#gear
MO B;'           Turn off servo to B motor
GA B,,B;'        Specify master axis as B
GR .25,,-5;'   Specify A and C gear ratios
SH B;'           Enable B axis
PRB= 1000;BG B;' Move B axis 1000 counts
'              A axis will be commanded to move 250 counts positive
'              C axis will be commanded to move 5000 counts negative (-5000)
EN;'           End program
```

```
'Galil DMC Code Example
REM setup gearing where B axis is master for A and C axes.
#gear
GA N;'   Specify master axis as N (imaginary Axis)
GR -2;'  Specify gear ratio or -2
PRN= 1000;BG N;' Move N axis 1000 counts
WT 1000
MG _RPA,_RPN;' will indicate -2000 on A and 1000 on N
EN;'           End program


:'execution of gearing example
:XQ
:
-2000.0000 1000.0000

:
```

GR applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

HM *Home*

HM mm

Usage	HM mm	Argument is an axis mask
Operands	_HMm	Operand has special meaning, see Remarks

Description

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

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
mm	A	ABCDEFGH	ABCDEFGH	Multi-Axis Mask	Axis to performing Homing Routine	No argument homes all axes

Remarks

- The FE command is derived of FE and FI commands and therefore you can create your own custom homing sequence by using the FE (Find Edge) and FI (Find Index) commands.
- The sequence of FE and FI commands varies depending upon if the axis is configured for a stepper or servo

Step One. Servos and Steppers

- During the 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 speed for step one is set with the SP command.
- The direction for this first stage is determined by the initial state of the homing input. The state of the homing input can be configured using the second field of the CN command.
- Once the homing input changes state, the motor decelerates to a stop.

Step Two. Servos and Steppers

- At the second stage, the motor changes directions and approaches the transition again at the speed set with the HV command. When the transition is detected, the motor is stopped instantaneously.

Step Three. Servos only

- At the third stage, the motor moves forward at the speed set with the HV command until it detects an index pulse via latch from the encoder. It returns to the latched position and defines it as position 0.

Operand

_HMm state as a function of CN,n and Home digital input

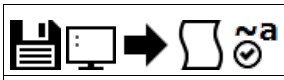
_CN1 value	Home input digital state	_HMn state	Direction of travel if HM begun in this state
-1	1 (pull-up or non-active opto)	1	Backward
-1	0 (grounded or active opto)	0	Forward
1	1 (pull-up or non-active opto)	0	Forward
1	0 (grounded or active opto)	1	Backward

Examples

```
'Galil DMC Code Example
:HM ;'      Set Homing Mode for all axes
:BG ;'      Home all axes
:HM A;'      Set Homing Mode for axis A
:BG A;'      Home only the A-axis
```

HM applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

HV *Homing Velocity*



HVm= n
HV n,n,n,n,n,n,n,n

Usage	HVm= n	Arguments specified with a single axis mask and an assignment (=)
	HV n ...	Arguments specified with an implicit, comma-separated order
Operands	_HVm	Operand holds the value last set by the command

Description

Sets the slew speed for the FI final move to the index and all but the first stage of HM.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	0	22,000,000	256	2	Value of Homing Velocity in cnts/second	For MT settings of 1,-1,1.5 and -1.5 (Servos)
	0	6,000,000	256	2	Value of Homing Velocity in cnts/second	For MT settings of 2,-2,2.5 and -2.5 (Steppers)

Remarks

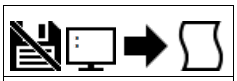
- None

Examples

```
'Galil DMC Code Example
HVA= 1000;'  set homing speed
HM A;'       home to home switch then index
BG A;'       begin motion
AM A;'       wait for motion complete
EN;'        end program
```

HV applies to DMC40x0,DMC42x0,DMC41x3,DMC18x6,DMC30010,DMC500x0

HX *Halt Execution*



HX n

Usage	HX n ...	Arguments specified with an implicit, comma-separated order
Operands	_HX0 _HX1 _HX2 _HX3 _HX4 _HX5 _HX6 _HX7	Operand has special meaning, see Remarks

Description

The HX command halts the execution of any program that is running.The parameter n specifies the thread to be halted.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	0	7	N/A	1	Thread number to halt	If n omitted, all threads are halted.

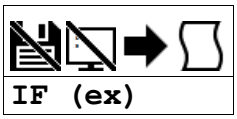
Remarks

- 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

Examples

```
'Galil DMC Code Example
XQ #a;'    Execute program #A, thread zero
XQ #b,3;' Execute program #B, thread three
HX 0;'     Halt thread zero
HX 3;'    Halt thread three
```

IF *IF conditional statement*



Usage	IF n ...	Arguments specified with an implicit, comma-separated order

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

Argument	Min	Max	Default	Resolution	Description	Notes
ex	N/A	N/A	N/A	Expression	Conditional statement for IF statement	See Remarks

Remarks

- 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
- Bit wise operators | and & can be used to evaluate multiple conditions.
- A true condition = 1 and an false condition = 0.
- Each condition must be placed in parenthesis for proper evaluation by the controller.

```
'Galil DMC Code Example
IF ((var0=1)&(var1=2));' valid IF statement

IF var0=1&var1=2;'      invalid IF statement

IF (var0=1&var1=2);'    invalid IF statement
```

Examples

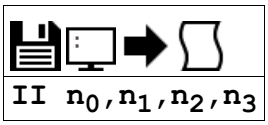
```
'Galil DMC Code Example
#a
IF (_TEA<1000);'  IF conditional statement based on a motor position
 MG "Motor is within 1000 counts of zero";' Message to be executed for true
ENDIF ;'           End of IF conditional statement
EN;'              End Program
```

```
'Galil DMC Code Example
#input
IF (@IN[1]=0);'          IF conditional statement based on input 1
 MG "Input 1 is Low";'   Message to be executed if "IF" statement is true
ENDIF ;'                 End of IF conditional statement
EN
```

```
'Galil DMC Code Example
#var
v1= @AN[1]*5;'            some calculation for variable v1
IF ((v1>25)&(@IN[4]=1));' Conditions based on V1 variable and input 4 status
 MG "Conditions met";'   Message to be executed if "IF" statement is true
ENDIF ;'                 End of IF statement
EN
```

```
'Galil DMC Code Example
REM The conditions of an if statement can be simplied with the fact that
REM a true condition = 1 and a false condition = 0.
#true
v1= 1
IF (v1)
 MG "True v1=",v1
ENDIF
#false
v1= 0
IF (v1)
 'if statement evaluates false
ELSE
 MG "False v1=",0
ENDIF
EN
```

II *Input Interrupt*



II n_0,n_1,n_2,n_3

Usage	II n ...	Arguments specified with an implicit, comma-separated order

Description

The II command enables the input interrupt function for the specified inputs.

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.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n_0	0	8	0	1	Lowest input to use for interrupt trigger	n_0=0 disables input interrupt
n_1	1	8	N/A	1	Highest input to use for interrupt trigger	n_1 must be >= n_0, If omitted n_1=n_0
n_2	1	255	N/A	1	Use bitmask as alternative selection of input interrupt triggers	If n_0 and n_1 used, n_2 is ignored, see Remarks
n_3	0	255	0	1	Bitmask specifying required input state for interrupt trigger	Default=interrupt triggers on low inputs, see Remarks

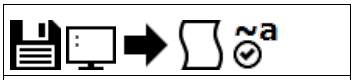
Remarks

- The argument n_2 is an integer value and represents a binary number showing the inputs selected for the input interrupt function.
  - For example, if n_2 = 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.
- This argument n_3 is an integer value and represents a binary number showing which inputs will trigger on a logic '1' and which on a logic '0'. This binary number is used to logically "AND" with the inputs which have been specified by the parameters n_1 and n_2 or the parameter n_3.
  - For example, if n_1=1 and n_2=4, the inputs 1,2,3 and 4 have been activated. If the value for n_3 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".
- The RI command is used to return from the #ININT routine.

Examples

```
'Galil DMC Code Example
#a;'                          Program A
II 1;'                        Specify interrupt on input 1
JG 5000;BG A;'                Specify jog and begin motion on A axis
#loop;JP #loop;'              Loop to keep thread zero active, only necesary on Econo (21x3/18x2)
EN;'                          End Program
#ININT;'                       Interrupt subroutine
ST A;MG "INTERRUPT";AM A;'    Stop A, print message, wait for motion to complete
AI 1;'                        Wait for input to switch states before continuing
'                              Otherwise we'll jump back in to #ININT
BG A;'                        Begin motion
RI 0;'                        Return to main program, don't re-enable trippoints
```

IL *Integrator Limit*



ILm= n
IL n,n,n,n,n,n,n,n

Usage	ILm= n	Arguments specified with a single axis mask and an assignment (=)
	IL n ...	Arguments specified with an implicit, comma-separated order
Operands	_ILm	Operand holds the value last set by the command

Description

The IL command limits the effect of the integrator gain in the filter to a certain voltage.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	-9.998	9.998	9.998	20/65,536	Value of Integrator limit in volts	n< 0 (negative value) freezes the effect of the integrator during the move

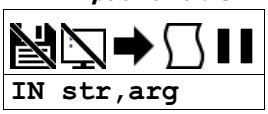
Remarks

- IL is the absolute value of the integrator limit. For example:
  - ILA= 2 limits the output of the integrator of the A-axis to the +/-2 Volt range.
  - KD and KP terms remain active in any case. The output from the KD and KP terms is not affected.
- A negative parameter will freeze the effect of the integrator during the move. For Example:
  - ILA= -3 limits the integrator output of the A axis to +/-3V but freezes the contribution of the Integrator loop during motion.
- If, at the start of the motion, the integrator output is 1.6 Volts, that level will be maintained through the move and the integrator will not accumulate during the move.
- Once the profiled move has completed (RP has reached final commanded position), the integrator loop will be enabled.

Examples

```
'Galil DMC Code Example
KI 2,3,5,8;' Integrator constants
IL 3,2,7,2;' Integrator limits
IL ?;'   Returns the A-axis limit
```

IN *Input Variable*



Usage	IN n ...	Arguments specified with an implicit, comma-separated order

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.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
str	0 chars	74 chars	N/A	String	The prompt message	See Remarks
arg	N/A	N/A	N/A	N/A	The variable where the response will be placed	

Remarks

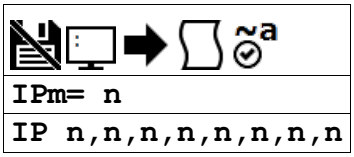
- The IN command may only be used in thread 0
- Do not include a space between the comma at the end of the input message and the variable name
- Entire command must be less than the total maximum line length. This determines the maximum length of str.
- Backlash'\' character will clear the IN command trippoint. The variable will not be overwritten in (will be last set value).
- Input Interrupts, Error Interrupts and Limit Switch Interrupts will still be active during the prompt

Examples

```
'Galil DMC Code Example
'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
BG A;'                        Begin motion
AM A;'                        Wait for motion complete
MG "MOVE DONE";'              Print Message
EN;'                          End Program
```

IN applies to DMC40x0,DMC42x0,DMC21x3,RIO,DMC18x6,DMC18x2,DMC500x0

IP *Increment Position*



```
IPm= n
IP n,n,n,n,n,n,n,n
```

Usage	IPm= n	Arguments specified with a single axis mask and an assignment (=)
	IP n ...	Arguments specified with an implicit, comma-separated order

Description

The IP command allows for a change in the command position while the motor is moving. This command does not require a BG.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
	M	N	N/A	Axis	Imaginary axis to assign value	
n	-2,147,483,648	2,147,483,647	N/A	1	Value of incremental move	

Remarks

- _IPm contains the current position of the motor
- The IP command has four effects depending on the mode of motion being executed.

IP operation based upon modes of motion

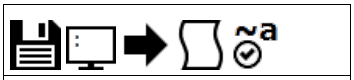
Case	Equivalent Commands	Description
Motor is standing still	IPm=n Equivalent to PRm=n;BGm	Motor will move to specified position with the predefined AC,DC,SP values.
Motor is moving toward position n	PRm=n_0; BGm;IPm=n_1 Equivalent to PRm=(n_0+n_1); BGm	Motor will move a relative move of (n_0+n_1).
Motor is in Jog Mode	JGm=n_0;BGm;IPm=n_1 Equivalent to Continuing jog from (current position + n_1)	The motor will instantly try to servo to a position which is the current instantaneous position plus the specified IP position. 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.
Motor is a slave in gearing mode	GAm= m0; GRm=n_0; IPm=n_1 Equivalent to GAm= m0; GRm=n_0; PRm=n_1; BGm	The motor will move with the predefined AC,DC,SP values superimposed on top of the existing gearing motion.

Examples

```
'Galil DMC Code Example
IP 50;'  50 counts with set acceleration and speed
#correct;' Label
AC 100000;' Set acceleration
JG 10000;BG A;' Jog at 10000 counts/sec rate
WT 1000;' Wait 1000 msec
IP 10;'   Move the motor 10 counts instantaneously
ST A;'    Stop Motion
EN
```

IP applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

IT *Independent Time Constant - Smoothing Function*



```
ITm= n
IT n,n,n,n,n,n,n,n
```

Usage	ITm= n	Arguments specified with a single axis mask and an assignment (=)
	IT n ...	Arguments specified with an implicit, comma-separated order
Operands	_ITm	Operand holds the value last set by the command

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.

The IT command also filters the individual axes during Vector Mode (VM) and Linear Interpolation Mode (LM).

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	0.004	1	1	1/256	Value of independent smoothing function	1 = no filtering, 0.004 = maximum filtering

Remarks

- The IT 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.
- Details on the IT filtering can be found in Application Note #3412
  - [http://www.galilmc.com/support/appnotes/optima/note3412.pdf]

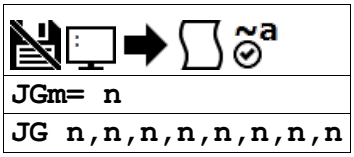
Examples

```
'Galil DMC Code Example
: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
0.8000
```

```
'Galil DMC Code Example
REM example showing increased time due to IT filtering
#move
IT 1
t= TIME;'store time reference
PR 1000
BG A;AM A
MG TIME-t;'display move time
IT 0.01
t= TIME;'store time reference
PR 1000
BG A;AM A
MG TIME-t;'display move time
EN

:'program execution output
:XQ
:
508.0000
1112.0000

:
```

IT applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

JG *Jog*



JGm= n

JG n,n,n,n,n,n,n,n

Usage	JGm= n	Arguments specified with a single axis mask and an assignment (=)
	JG n ...	Arguments specified with an implicit, comma-separated order
Operands	_JGm	Operand has special meaning, see Remarks

Description

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

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
	N	M	N/A	Axis	Virtual axis to assign value	
n	-22,000,000	22,000,000	25,000	2	Value of jog speed in cnts/second	For MT settings of 1,-1,1.5 and -1.5 (Servos)
	-6,000,000	6,000,000	25,000	2	Value of jog speed in cnts/second	For MT settings of 2,-2,2.5 and -2.5 (Steppers)
	-50,000,000	50,000,000	25,000	2	Vale of jog speed in cnts/second	ICM-42100 with AF>=5

Remarks

- When jogging, the motion controller profiles a continuous move at the commanded speed.
- To stop the motion, use the ST command.
- JG 2 is the minimum non-zero speed
- _JGm contains the absolute value of the jog speed for the specified axis.
- The JG command will set the SP register with the absolute value of the 'n' value.

Resolution

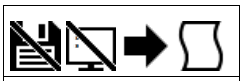
- The resolution of the JG command is dependent upon the update rate setting (TM).
  - With the default rate of TM 1000 the resolution is 2 cnts/second.
  - The equation to calculate the resolution of the JG command is:
    - resolution = 2*(1000/TM)
  - example:
    - With TM 250 the resolution of the JG command is 8 cnts/second
    - resolution = 2*(1000/250) = 8

Examples

```
'Galil DMC Code Example
#jg
JG 100,500,2000,5000
'                       Sets 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
WT 1000;'               Wait one second
JG ,,-2000;'         Change the C-axis to slew in the negative direction at -2000 counts/sec.
EN
```

JG applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

JP *Jump to Program Location*



```
JP #str,(ex)
JP n,(ex)
```

Usage	JP n ...	Arguments specified with an implicit, comma-separated order

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. A jump is taken if the specified condition is true. Multiple conditions can be used in a single jump statement.

JP can be used for relative jumps and for jump tables, see Examples.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
str	1 char	7 chars	N/A	String	Label name for jump destination	Must be a valid label in application code
n	0	see Notes	N/A	1	Line number for jump destination	Maximum is number of lines of controller program memory - 1
ex	N/A	N/A	N/A	Expression	Conditional statement/s that must evaluate true for jump to occur	If omitted, JP automatically evaluates as true

Remarks

- The logical operators that can be used in the conditional statement are:
  - < less than
  - > greater than
  - = equal to
  - <= less than or equal to
  - >= greater than or equal to
  - <> not equal to
- 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.
- Each condition must be placed in parentheses for proper evaluation by the controller.

```
'Galil DMC Code Example
REM Use of parentheses
JP #a,((var0=1)&(var1=2));' valid conditional jump
JP #a,var0=1&var1=2;'       invalid conditional jump
```

Examples

```
'Galil DMC Code Example
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,(@IN[1]=1);'  Jump to #B if input 1 = 1
JP #c;'             Jump to #C unconditionally
```

Jump Table

```
'Galil DMC Code Example
REM Example of jumping to a label plus an offset
REM #error is a subroutine that prints an error
REM  message based on the value of an error
REM  variable, ecode
#a
REM Set error code and then JS to sub
ecode = 1
JS #error
ecode = 3
JS #error
ecode = 56;' bad error code
JS #error
EN
'
'****************
'Example of a Jump table
#error
REM First check that ecode is valid
IF (ecode < 0)
 ecode = 4
ENDIF
IF (ecode > 4)
 ecode = 4
ENDIF
REM Call the helper label with an offset
JP #error_h + ecode
```

```
'CRITICAL! Do not change line
' spacing in following text
#error_h;MG "No error, zero";EN
MG "Error code 1, foo";EN
MG "Error code 2, bar";EN
MG "Error code 3, baz";EN
MG "Invalid error code";EN
REM ecode indexes the line to execute
REM  above, relative to #error_h
REM
REM Returned messages:
REM Error code 1, foo
REM Error code 3, baz
REM Invalid error code
```

Relative Jump

```
'Galil DMC Code Example
REM A loop for delaying 1000 samples (~ 1 sec)
REM sample time
MG "Relative jump"
t= TIME
REM print sampled time
MG t
REM loop until TIME increments 1000 samples
REM _XQ0-1 points back to the beginning of the line
JP _XQ0-1,(TIME < (t+1000))
REM print current time
MG TIME
REM This is NOT thread safe as
REM _XQ0 refers to thread 0 only
REM For easier readability and stability, use labels
REM wherever possible
MG "Label-based jump"
t= TIME
MG t
#wait
JP #wait, (TIME < (t+1000))
MG TIME
REM Also, where possible use trippoints
MG "Trippoint"
t= TIME
MG t
WT 1000;' see WT for units
MG TIME
EN

REM Relative jump
REM  3459.0000
REM  4459.0000
REM Label-based jump
REM  4461.0000
REM  5461.0000
REM Trippoint
REM  5463.0000
REM  6464.0000
```

JS *Jump to Subroutine*

```
JS #str(arg,arg,arg,arg,arg,arg,arg,arg), (ex)
JS n(arg,arg,arg,arg,arg,arg,arg,arg), (ex)
```

Usage	JS n ...	Arguments specified with an implicit, comma-separated order
Operands	_JS	Operand has special meaning, see Remarks

Description

Allows the program to jump to a subroutine and return back after completion. This command is often used to call reusable code.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
str	1 char	7 chars	N/A	String	Label Name for jump destination	Must be a valid label in application code
n	0	1,999	N/A	1	Line number for jump destination	May be a value or a variable, but not an evaluated statement with parenthesis
ex	N/A	N/A	N/A	N/A	Conditional statement/s that must evaluate true for jump to occur	If omitted, the jump is taken
arg	N/A	N/A	N/A	N/A	A value, variable, or array to pass to the subroutine being called	referenced from within the subroutine as ^a-^h, respectively. See Remarks for a table of valid args

Remarks

- JS can be nested, called up to 16 deep
- When used after JS is called, the _JS operand contains the returned value of the subroutine called by JS

Basic Usage

- 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. A variable holding a line number or an expression resulting in the calculation of a line number can also be used
- The line number of the calling JS command is saved and after an EN command is encountered (End of subroutine), program execution will continue with the instruction following the calling JS command.
- A jump is taken if the specified condition is true. Each condition must be placed in parenthesis for proper evaluation by the controller.
- Code flexibility/reuse. A single subroutine can be written and called many times and from various locations in code. The stack "remembers" where to return when completed. This is opposite from a "blind jump" (JP).

Conditional Syntax

Condition	Validity
JS#A,(var1=0)&(var2=1)	This conditional statement is valid
JS#A,var1=0&var2=1	This conditional statement is not valid

Passing Values on the Stack

- Parameters can be passed on the subroutine stack
- Passing parameters in a subroutine has many advantages including the following
  - Variable Scope/ Local variables. A subroutine can run with a protected variable space. Local variables exist only in the extent of the subroutine, and no external thread or stack level can access local variables. Local variables can be used for counters, indices, and other helper variables
  - Each thread has its own stack, therefore subroutines are reentrant. In other words, multiple threads can be running the same subroutine simultaneously at various stack depths.
  - Support for recursion. Although the subroutine stack is only 16 deep, recursion is possible. A stack depth of 16 is sufficient for many recursive tasks. E.G. recursing axes, handles, and thread status.
  - Parameter passing. A calling command can explicitly specify the inputs to a subroutine. The subroutine can pass one value back to the calling command. More returns are possible with pass by reference and array passing.
- Constants, Variables, and Arrays may be passed up a subroutine stack.
- Variables may be passed by value or by reference. If passed by value, a copy is made in the subroutine stack, leaving the original variable immutable. If passed by reference, the original variable's value will be changed when the subroutine writes to its local variable. This is similar, but not exactly analogous to a C pointer.
- A variable passed by reference is automatically dereferenced; the variable pointer is not exposed to the user. Following the C syntax, a by-reference pass is accomplished with the ampersand (&) in the invoking call.
  - IMPORTANT NOTE: When passing a variable by reference, do not allocate any new variables in the called subroutine.
- Arrays can be passed in the stack, though only by reference. No "&" is used when passing arrays, by-reference is assumed. To pass an array, use its name in quotations.
  - IMPORTANT NOTE: Arrays to be passed must have names that are 6 characters or less.
- The number of elements in an array is returned by reading index -1, e.g. array[-1].
- To return a value on the stack, write the value in the EN command upon ending the subroutine. The parent stack can access this value via _JS.

Examples of valid args (see examples for demo of each concept)

What to pass	arg	Example

Value	the value	JS #square(7)
Variable's value	variable name	JS #sub(var)
Variable by reference	ampersand + variable name	JS #sub(&var)
Array by reference	array name in quotes	JS#sub("array")

Examples

```
'Galil DMC Code Example
REM Example of pulsing an output
pulse= 0
JS #pulse,(pulse > 0);'JS not taken
WT 2000
pulse= 3
JS #pulse,(pulse > 0);'JS taken
WT 2000
pulse= 5
JS #pulse;'unconditionally take jump
EN
'
'
REM Subroutine called after
REM  setting pulse variable
#pulse
SB 1;' set bit 1
WT 500;' delay 500 ms
CB 1;' clear bit 1
WT 500;' delay 500 ms
pulse= pulse-1;' decrement pulse
JP #pulse,pulse>0;' continue till zero
EN;' return to calling JS
```

Advanced Usage Examples

```
'Galil DMC Code Example
REM Run all examples
#all
JS #val
JS #var
JS #varref
JS #array
EN
REM Example for each way to pass to a subroutine
REM ********************************************
REM Pass a Value
#val
JS #square(3)
MG _JS
EN
#square
REM Return the passed value squared
EN ,,(^a*^a)
REM ********************************************
REM Pass a variable's value
#var
val= 7
REM call the same sub above
JS #square(val)
MG _JS
EN
REM ********************************************
REM Pass a variable by reference
#varref
val= 9
JS #square2(&val)
MG val
EN
#square2
REM  change the value of the variable
^a= ^a*^a
REM don't return anything
EN
REM ********************************************
REM Pass an array by reference
#array
DM array[100]
array[42]= 11
JS #square3("array")
MG array[42]
EN
#square3
REM change the array element
^a[42]= ^a[42]*^a[42]
REM don't return anything
EN
REM ********************************************
REM Controller Response
REM :XQ#all
REM :
REM 9.0000
```

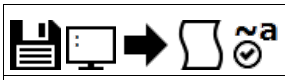
```
REM 49.0000
REM 81.0000
REM 121.0000
```

```
'Galil DMC Code Example
#add
JS #sum(1,2,3,4,5,6,7,8);'      Call subroutine, pass values
MG _JS;'                        Print return value, will print 36.0000
EN

#sum;'                          Sums values passed to it. Expects 8 numbers
EN ,,^a+^b+^c+^d+^e+^f+^g+^h;'  Return the sum
```

```
'Galil DMC Code Example
'Dimension two arrays
DM array1[10]
DM array2[100]
'Zero the contents of each array
JS #zeroary("array1", 0)
JS #zeroary("array2", 0)
EN

'Zero the contents of an array
#zeroary;'(^a array,^b starting index)
^a[^b]= 0
^b= (^b+1)
JP #zeroary,(^b < ^a[-1])
EN
```

```
'Galil DMC Code Example
REM Using dynamic destinations in a jump table
i= 1;'               Counter
#loop
 offset= #spell+i;'  Calculate offset
 JS offset;'         Jump to offset
 i= i+1;'            Increment Counter
JP #loop,i<=3;'      Loop through 3 states
EN
'
#spell;'             Subroutine containing various words
MG "One";EN;'        Prints "One" if this line is called (i=1)
MG "Two";EN;'        Prints "Two" if this line is called (i=2)
MG "Three";EN;'      Prints "Three" if this line is called (i=3)

REM Controller responds with:
REM One
REM Two
REM Three
```

KD *Derivative Constant*



KDm= n
KD n,n,n,n,n,n,n,n

Usage	KDm= n	Arguments specified with a single axis mask and an assignment (=)
	KD n ...	Arguments specified with an implicit, comma-separated order
Operands	_KDm	Operand holds the value last set by the command

Description

KD designates the derivative constant in the control filter. The derivative gain outputs a voltage based on the rate of change of the error. The filter transfer function follows:

$$D(z) = KP + KD\frac{z-1}{z} + KI\frac{z}{z-1}$$

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	0	4,095.875	64	1/8	Value of derivative term	

Remarks

- n=? will return the currently set value of KD
- m=* will set the KD value for all axes/channels

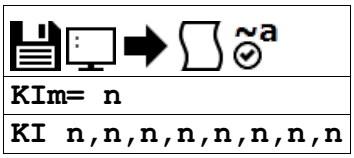
- For further details see the section "Theory of Operation" in the controller user manual.

Examples

```
'Galil DMC Code Example
:KD 12,14,16,20;' Implicit notation to set A,B,C,D axis derivative term
:KDC= 8;'         Explicit notation to set C
:KD ,,8;'         Implicit notation to set C
:KD ?,?,?,?; '    Return A,B,C,D values
 12, 14, 8, 20
:KDC= ?;'         Return C value
 8
:MG _KDA;'        Message the operand for the A axis
 12
:
```

```
'Galil DMC Code Example
REM Zeroing the PID filter allows the
REM motor command signal to be
REM used as a programmable DAC
KI*= 0;'                Zero KI
KP*= 0;'                Zero KP
KD*= 0;'                Zero KD
ER -1,-1;'              Rurn off position error limit
OF 1,2;'                Set one volt on A and two volts on B
EN
```

KI *Integrator*



KIm= n

KI n,n,n,n,n,n,n,n

Usage	KIm= n	Arguments specified with a single axis mask and an assignment (=)
	KI n ...	Arguments specified with an implicit, comma-separated order
Operands	_KIm	Operand holds the value last set by the command

Description

The KI command sets the integral gain of the control loop. The integrator term will reduce the position error at rest to zero. It fits in the control equation as follows:

$$D(z) = KP + KD\frac{z-1}{z} + KI\frac{z}{z-1}$$

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	0	255.999	0	1/1,024	Value of Integral term	

Remarks

- n=? will return the currently set value of KD
- m=* will set the KD value for all axes/channels

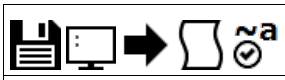
- For further details see the section "Theory of Operation" in the controller user manual.

Examples

```
'Galil DMC Code Example
:KIC= 8;'          Explicit notation to set C
:KI  ,,8;'         Implicit notation to set C
:KI ?,?,?,?; '     Return A,B,C,D values
 7, 14, 8, 20
:KIC= ?;'          Return C value
 8
:MG _KIA;'         Message the operand for the A axis
 7
:
```

```
'Galil DMC Code Example
REM Zeroing the PID filter allows the
REM motor command signal to be
REM used as a programmable DAC
KI*= 0;'               Zero KI
KP*= 0;'               Zero KP
KD*= 0;'               Zero KD
OF 1,2;'               Set one volt on A and two volts on B
EN
```

KP *Proportional Constant*



```
KPm= n
KP n,n,n,n,n,n,n,n
```

Usage	KPm= n	Arguments specified with a single axis mask and an assignment (=)
	KP n ...	Arguments specified with an implicit, comma-separated order
Operands	_KPm	Operand holds the value last set by the command

Description

KP designates the proportional constant in the controller filter. The proportianal gain outputs a control signal proportional to the amount of error. The filter transfer function follows.

$$D(z) = KP + KD\frac{z-1}{z} + KI\frac{z}{z-1}$$

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	0	1,023.875	6	1/8	Value of proportional term	

Remarks

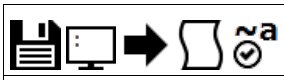
- n=? will return the currently set value of KP
- KP now has four times more resolution as prior controllers, and thus the same value as that of an Optima controller is four times less effective
- Maximum range increased from 1023.875 from 4095.875 in firmware revision 1.1a
- For further details see the section "Theory of Operation" in the controller user manual.

Examples

```
'Galil DMC Code Example
:KP 12,14,16,20;' Implicit notation to set a,b,c,d axis proportional term
:KPC= 8;'          Explicit notation to set C
:KP ,,8;'          Implicit notation to set C
:KP ?,?,?,?; '     Return A,B,C,D values
 7, 14, 8, 20
:KPC= ?;'          Return C value
 8
:MG _KPA;'         Message the operand for the A axis
 12
:
```

```
'Galil DMC Code Example
REM Zeroing the PID filter allows the
REM motor command signal to be
REM used as a programmable DAC
KI*= 0;' Zero KI
KP*= 0;' Zero KP
KD*= 0;' Zero KD
OF 1,2;' Set one volt on A and two volts on B
EN
```

KS *Step Motor Smoothing*



KSm= n

KS n,n,n,n,n,n,n,n

Usage	KSm= n	Arguments specified with a single axis mask and an assignment (=)
	KS n ...	Arguments specified with an implicit, comma-separated order
Operands	_KSm	Operand holds the value last set by the command

Description

The KS parameter sets the amount of smoothing of stepper motor pulses. Larger values of KS provide greater smoothness. KS adds a single pole low pass filter onto the output of the motion profiler.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	0.25	64	2	1/32	Value of smoothing constant	

Remarks

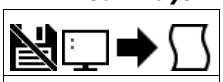
- This is most useful when operating in full or half step mode.
- KS effect on timing:
  - This parameter will increase the time to complete a motion time by 3KS sampling periods.
  - KS will cause an overall delay in the generation of output steps.

Examples

```
'Galil DMC Code Example
:KSC= 8;'          Explicit notation to set C
:KS ,,8;'          Implicit notation to set C
:KS ?,?,?,?; '     Return A,B,C,D values
 7, 14, 8, 20
:KSC= ?;'          Return C value
 8
:MG _KSA;'         Message the operand for the A axis
 7
:
```

KS applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

LA *List Arrays*



LA

Usage	LA	Command takes no arguments

Description

The LA command returns a list of all arrays in memory. The size of each array will be included next to each array name in square brackets.

Arguments

LA is an interrogation command with no parameters

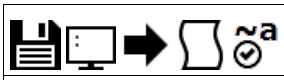
Remarks

- The listing will be in alphabetical order.

Examples

```
'Galil DMC Code Example
:DM gold[100],silver[50],plat[200];'     Dimensions arrays with given name and the number of array elements in square brakets
:LA;'                                    Commands the controller to list arrays in alphabetical order
gold[100]
plat[200]
silver[50]
:DA *[];'                                Dialocates all arrays
:LA;'                                    List arrays now returns with no arrays
:
```

LC *Low Current Stepper Mode*



LCm= n
LC n,n,n,n,n,n,n,n

Usage	LCm= n	Arguments specified with a single axis mask and an assignment (=)
	LC n ...	Arguments specified with an implicit, comma-separated order
Operands	_LCm	Operand holds the value last set by the command

Description

The LC command enables low current mode for stepper motors. Low current mode reduces the holding torque of the stepper motors while at rest.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	0	1	0	1	n= 0, Default, stepper drive providing 100% torque at rest; n= 1, 25% holding torque when motor at rest.	
	2	32,767	0	1	Specifies "n" samples after move before going to 0% holding current	

Remarks

- The MT command must be issued prior to the LC command.
- Low current mode will simply disable the motors by toggling the amplifier enable line during rest.
- Using LC will reduce current consumption, but there will be no holding torque at rest.

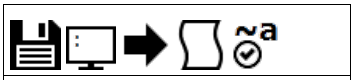
Examples

```
'Galil DMC Code Example
MTC= -2;' Specify stepper mode for the C axis
LCC= 1;' Specify low current mode for the C axis
```

```
'Galil DMC Code Example
#ex
MTA= -2;'specify stepper mode for A axis
LCA= 15;'specify motor to go to low current
' 15 samples after motion has completed
EN
```

LC applies to DMC40x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

LD *Limit Disable*



LDm= n

LD n,n,n,n,n,n,n,n

Usage	LDm= n	Arguments specified with a single axis mask and an assignment (=)
	LD n ...	Arguments specified with an implicit, comma-separated order
Operands	_LDm	Operand holds the value last set by the command

Description

Allows user to disables forward and/or reverse limit switches.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	0	4	0	1	Sets limit disable state	See table below for details

Argument	Value	Description	Notes
n	0	Both limit switches are enabled	Default
	1	Forward limit switch disabled	
	2	Reverse limit switch disabled	
	3	Both limit switches disabled	

Remarks

- n = ? will return the current setting of LD
- When this feature should be used:
  - To gain additional digital inputs if limit switches are not being utilized.
  - To prevent noise from causing the limit switchs conditions even though no limit switches are connected.
- LD does not disable software limits set by BL and FL.

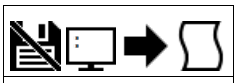
Examples

```
'Galil DMC Code Example
:LD 3,1,2;            'Implicit notation to set channel A, B, and C
:MG _LDA;'            'Message the operand for the A channel
 3.0000
:LDC= 3;'             'Explicit notation to set channel C-only
:LD*= ?;'             'Queries the value of LD for all channels
 3, 1, 3, 0
:
```

```
'Galil DMC Code Example
REM use forward limit switch as an extra I/O point
#io
LDA= 1;'disable forward limit switch
io= _LFA;'set state of limit switch to variable "io"
'Use "io" in an IF statement
IF io=1
 MG "Input On"
ELSE
 MG "Input Off"
ENDIF
EN
```

LD applies to DMC40x0,DMC42x0,DMC41x3,DMC18x6,DMC30010,DMC500x0

LE *Linear Interpolation End*



LE n

Usage	LE	Command takes no arguments
Operands	_LEm	Operand has special meaning, see Remarks

Description

The LE command indicates to the controller that the end of the sequence is coming up. This allows the controller to slow down through multiple segments, if required. LE is required to exit the linear interpolation mode gracefully (stop code, SC, 101).

Arguments

The LE command has no arguments. See the ? Remark below.

Remarks

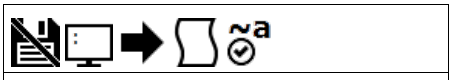
- _LEn will return the total move length in encoder counts for the selected coordinate system, where n is S or T.
- If not spcified, the LE command will apply to the last selected coordinate system, S or T.
- To select the coordinate system, use the command CA S or CA T.
- The VE command is interchangeable with the LE command.
- LE ? Returns the total vector move length in encoder counts for the current coordinate system

Examples

```
'Galil DMC Code Example
CA S;           'Specify S coordinated motion system
LM CD;          'Specify linear interpolation mode for C and D axes
LI ,,100,200;   'Specify linear distance
LE;             'Ends linear interpolation distance
BG S;           'Begin motion of the S-coodrinate system
```

LE applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

LI *Linear Interpolation Distance*



```
LIm= n <o >p
LI n,n,n,n,n,n,n,n <o >p
```

Usage	LIm= n	Arguments specified with a single axis mask and an assignment (=)
	LI n ...	Arguments specified with an implicit, comma-separated order

Description

The LI 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.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	-8,388,607	8,388,607	0	1	Assigns linear interpolation point for that axis	
o	2	22,000,000	N/A	2	Specifies the vector speed to be commanded at the beginning of the linear segment. The controller will start accelerating or decelerating at the start of the sequence to this speed.	For MT 1,-1,1.5 and -1.5.
	2	6,000,000	N/A	2	Specifies the vector speed to be commanded at the beginning of the linear segment. The controller will start accelerating or decelerating at the start of the sequence to this speed.	For MT 2,-2,2.5 and -2.5
p	2	22,000,000	N/A	2	Specifies the vector speed to be achieved at the end of the linear segment. The controller will decelerate or accelerate during the segment and will reach the specified speed at the end of the segment.	For MT 1,-1,1.5 and -1.5.
	2	6,000,000	N/A	2	Specifies the vector speed to be achieved at the end of the linear segment. The controller will decelerate or accelerate during the segment and will reach the specified speed at the end of the segment.	For MT 2,-2,2.5 and -2.5

Argument	Value	Description	Notes
o	-1	Specifies vector speed to be set by Vector Speed Variable (VV command)	See VV command

Remarks

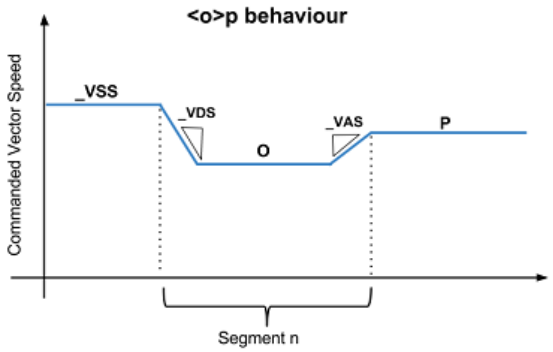
- The CA command is used to set the coordinated system (S or T) for which an LI segment is executed. The default is the S coordinate system (CAS).
- The controller always uses the axis specifications from LM, not LI, to compute the speed.
  - For example: if LM specfies that A-, B-, and C-axis are to be used in linear interpolation mode, but LI only specifies positions for B- and C-, the A-axis will still be used in calculating the overall vector speed.
  - The maximum independent speed of any axis configured as a stepper must not exceed the maximum value allowable via the SP setting.
- The slew speed, set by VS, 'o' or 'p' for linear interpolation mode, is the vector speed based on the axes specified in the LM mode. For example, if LM ABC designates linear interpolation for the A,B and C axes the speed of these axes (Va, Vb, and Vc respectively) will be computed from:

$$VS = \sqrt{V_A^2 + V_B^2 + V_C^2}$$

- The Linear End (LE) command must be given after the last LI segment in a sequence. LE tells the controller to decelerate to a stop at the last LI command.
- The BG S or BG T command should be issued before the total LI distance reaches 1,073,741,824 (2^30) encoder counts.

Linear Interpolation Mode Buffer

1. Up to 511 LI segments may be given ahead of the begin sequence (BG S or BG T) command.
2. Additional LI commands may be sent during motion when the controller sequence buffer frees additional space for new vector segments.
3. It is the responsibility of the user to keep enough LI segments in the controller's sequence buffer to ensure continuous motion.
4. _LMm (_LMS and _LMT) contains the available spaces for LI segments that can be sent to the buffer.
   1. 511 returned means the buffer is empty and 511 LI segments can be sent.
   2. A 0 returned means the buffer is full and no additional segments can be sent.
   3. See the LM command for full details.

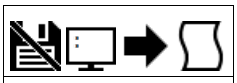


Examples

```
'Galil DMC Code Example
LM ABC;               'Specify linear interpolation mode between A-, B-, and C- axis
LI 500,,400;          'Specifies linear interpolation point, B-axis remains stagnat but is still part of the interpolation.
LI 1000,2000,3000;    'Specify  linear interpolation point
LE;                   'Last segment of sequence
BG S;                 'Begin sequence
```

LI applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

LL *List Labels*



LL

Usage	LL	Command takes no arguments

Description

The LL command returns a listing of all of the program labels in memory.

Arguments

LL is an interrogation command with no arguments

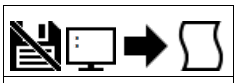
Remarks

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

Examples

```
'Galil DMC Code Example
:LL
#FIVE=5
#FOUR=4
#ONE=1
#THREE=3
#TWO=2
```

LM *Linear Interpolation Mode*



LM mm

Usage	LM mm	Argument is an axis mask
Operands	_LMm	Operand has special meaning, see Remarks

Description

The LM command specifies the linear interpolation mode and specifies the axes for linear interpolation.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
mm	A	ABCDEFGH	N/A	Multi-Axis Mask	Axes to use for linear interpolation mode	

Remarks

- Any set of axis may be used for linear interpolation.
- LI commands are used to specify the travel distances between various linear interpolation moves.
- Several LI commands may be given as long as the controller sequence buffer has room for additional segments
  - See the LI command for more information regarding the Linear Inerpolation Buffer
- The LE command specifies the end of the linear interpolation sequence.
- Once the LM command has been given, it does not need to be given again unless the VM command has been used

Operand/Queries

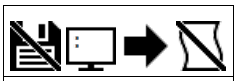
- _LMm contains the number of spaces available in the sequence buffer for the 'm' coordinate system, S or T.
- The LM command will apply to the selected coordinate system, S or T. To select the coordinate system, use the command CA S or CA T.

Examples

```
'Galil DMC Code Example
LM ABCD;                          'Specify linear interpolation mode
VS 10000;VA 100000;VD 1000000;    'Specify vector speed, acceleration and deceleration
LI 100,200,300,400;               'Specify linear distance
LI 200,300,400,500;               'Specify linear distance
LE; BG S;                         'Last vector, then begin motion
```

LM applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

LS *List*



`LS n0,n1`

Usage	LS n ...	Arguments specified with an implicit, comma-separated order

Description

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

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n0	0	1,998	0	1	Specifies the line in the program for which the listing will start	
n1	1	1,999	1,999	1	Specifies the line at which the listing will end	

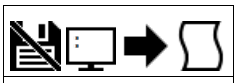
Remarks

- $n_0 < n_1$ must always be true
- If n_0 or n_1 is omitted, default values are used
- n_0 and n_1 can also specify a label, for example:
  - "LS #label,20" would print out program lines from #label to line 20.

Examples

```
'Galil DMC Code Example
:LS #a,6;        ' List program starting at #A through line 6
2 #a
3 PR 500
4 BG A
5 AM
6 WT 200
'Hint:  Remember to quit the Edit Mode  Q prior to giving the LS command. (DOS)
```

LV *List Variables*



LV

Usage	LV	Command takes no arguments

Description

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

Arguments

LV is an interrogation command with no parameters

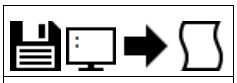
Remarks

- Use the _UL operand for total number of variables available for your controller.
  - See the UL command for more details.

Examples

```
'Galil DMC Code Example
:LV
APPLE = 60.0000
BOY = 25.0000
ZEBRA = 37.0000
:
```

LZ *Omit leading zeros*



LZ n

Usage	LZ n ...	Arguments specified with an implicit, comma-separated order
Operands	_LZ	Operand has special meaning, see Remarks

Description

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

Arguments

Argument	Value	Description	Notes
n	0	Does not remove leading zeros from interrogated values	
	1	Removes leading zeros from interrogated values	Default

Remarks

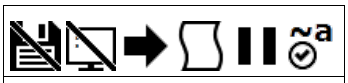
- _LZ contains the state of the LZ function. '0' is disabled and '1' is enabled.

Examples

```
'Galil DMC Code Example
:LZ 0;                  'Disable the LZ function
:var1= 10;              'Sets variable var1 to the value of 10.
:TP A;                  'Interrogate the controller for current position of A-axis
 0000021645.0000
:var1= ?;                'Request value of variable var1
 0000000010.0000
:LZ 1;                  'Enable LZ function
:TP A;                  'Interrogate the controller for current position of A-axis
 21645.0000
:var1= ?;               'Request value of variable var1
 10.0000
```

```
'Galil DMC Code Example
:LZ 0;      'Disable the LZ function
:TB;        'Tell status bits
001
:LZ 1;      'Inhibit leading zeros
:TB;        'Tell status
1
```

MC *Motion Complete*



MC mm

Usage	MC mm	Argument is an axis mask

Description

The MC command is a trippoint command that holds up execution until motion is complete on any one of a specified group of axes. The MC command, unlike the AM (after motion command) requires that both the motion profiler has completed motion AND that the motor encoder has reached the specified position before continuing execution.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
mm	A	ABCDEFGH	ABCDEFGH	Multi-Axis Mask	Axis to assign value	Any combination of the axis is valid. If no axis is specified, command applies to all axis.

Remarks

- Although many axes can be spcified, the MC command will continue execution if one of the specified axis motion is completed.

Using MC with Stepper Motors

- In the case of stepper motors, MC will monitor the number of step pulses are generated to complete the move.
- The MC command is recommended when operating with stepper motors in leiu of AM 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.

Using MC as part of the #MCTIME error routine

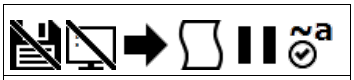
1. The command TW can be used to set an acceptable amount of time between when the motion profiler has completed and the encoder is in position; if this condition is not satisfied, a timeout error occurs.
    1. When a timeout occurs, the trippoint will clear and the stop code will be set to 99.
    2. Thread 0 of the DMC program will also jump to the special label #MCTIME, if present.
        1. See the #MCTIME automatic subroutine, TW and SC commands for more information

Examples

```
'Galil DMC Code Example
#move;                  'Label #move
TW 1000,1000;           'Set motion complete timeout to 1000 milliseconds per axis
PR 2000,4000;           'Position relative Move on A- and B-axis
BG AB;                  'Start the motion on A- and B-axis
MC AB;                  'After the move is complete on A and B axes
MG "DONE";              'Print message
EN;                     'End of Program
'
'
#MCTIME;                 'Motion Complete timeout Subroutine
 MG "Motion Timeout";   'Print failure message
 SC ;                    'Print stop codes
EN;                     'End subroutine
```

MC applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

MF *Forward Motion to Position*



MFm= n

MF n,n,n,n,n,n,n,n

Usage	MFm= n	Arguments specified with a single axis mask and an assignment (=)
	MF n ...	Arguments specified with an implicit, comma-separated order

Description

This command will hold up the execution of the following command until the specified motor moves forward and crosses the position specified.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	-2,147,483,648	2,147,483,647	N/A	1	Position required to be crossed before subsequent commands will be executed.	

Remarks

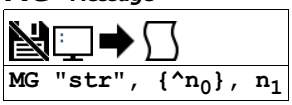
- Although multiple positions can be specified, only one of the MF conditions must be satisfied for subsequent code execution.
- MF command references absolute position.
- The MF command only requires an encoder and does not require that the axis be under servo control.
- The accuracy of the MF command is the number of counts that occur in 2*TM sec. Multiply the speed by 2*TM sec to obtain the maximum error.
  - Example with speed of 20,000 counts/second and TM of 1000 (1000 us).
    - Maximum error = 2 * 1000 E-6 seconds * 20,000 counts/second = 40 counts
- When using a stepper motor:
  - This condition is satisfied when the stepper position (as determined by the output buffer - TD) has crossed the specified Forward Motion Position.

Examples

```
'Galil DMC Code Example
#test;                  'Program Test
DP 0;                   'Define zero
JG 1000;                'Jog mode (speed of 1000 counts/sec)
BG A;                   'Begin move
MF 2000;                'After passing the position 2000
v1= _TPA;               'Assign V1 A position
MG "Position is",v1;    'Print Message
ST A;                   'Stop
EN;                     'End of Program
```

MF applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

MG *Message*



Usage	MG n ...	Arguments specified with an implicit, comma-separated order

Description

The MG command is used to send strings, operands, variables, and array values to a specified destination.

Arguments

Argument	Value	Description	Notes
str	String	A string including alphanumeric characters to be displayed	Limited to 76 characters
n_0	ASCII character in decimal	Allows users to print ASCII characters	Range of 0-255
n_1	Numeric value	Prints the numeric value specified	See Examples for valid uses of n_1.
	Variable name	Prints the numeric value stored by the variable	
	Operand	Prints the numeric value stored by the operand	
	Array element	Prints the numeric value stored by the array element	
	Mathematical expression	Prints the numeric value of the solved equation	

Remarks

- Multiple strings, variables, and ASCII characters may be used; each must be separated by a comma.
- Solicited Messages
  - From a host terminal, application code, or device, sending the MG command will return with the requested information. This is known as a solicited command, because the host sends the command and expects a response.
- Unsolicited Messages
  - From embedded DMC code, the MG command will send an unsolicited, asynchronous message from the controller to the host. This can be used to alert an operator, send instructions, or return a variable value. This is known as an unsolicited command because the host is not explicitly requesting it.
  - The CW command controls the ASCII format of all unsolicited messages.

Formatting

- Formatters can be placed after each argument in to modify how it is printed.
  - {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 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. If n is greater than the length of the string stored in the variable, null chars (0x00) will be inserted at the end of the string.
  - {N} Suppress carriage return at the end of the message.

Examples

Valid uses of n_1 argument

```
'Galil DMC Code Example
:'Values
:MG 1234.5678
 1234.5678
:'
:'Variables
:var= 12345678.9101
:MG var
 12345678.9101
:'
:'Operands
:MG @AN[1]
 0.0121
:'
:'Array Elements
:DM arr[3]
:arr[0]= 0
:arr[1]= 1
:arr[2]= 2
:MG arr[0],arr[1],arr[2]
 0.0000 1.0000 2.0000
:'
:'Mathematical Expressions
:MG 1+2
 3.0000
:MG arr[2]+var
 12345680.9101
:
```

General Use

```
'Galil DMC Code Example
:MG "Good Morning";                    'Message command displays ASCII string
Good Morning
:total= 1234.5322;                     'Assigns variable total with the value 1234.5322
:MG "The answer is...",total{F4.2};    'Will print the message and the value of variable total formatted with 4 integer digits and 2 fractional digits
The answer is... 1234.53
:MG {^13}, {^10}, {^48}, {^055};       'Specifies carriage return, line feed, and the characters 0 and 7 in ASCII decimal values

07
:MG TIME;                              'Messages the operand TIME
 261928200.0000
:variable= 10;                         'Sets the variable equal to 10
:MG variable+5;                        'Messages out variable + 5
 15.0000
:MG _TI0;                              'Messages the value stored in the operand _TI0
 255.0000
```

MO *Motor Off*

MO mm

Usage	MO mm	Argument is an axis mask
Operands	_MOm	Operand has special meaning, see Remarks

Description

The MO command turns off the motor command line and toggles the amplifier enable signal.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
mm	A	ABCDEFGH	ABCDEFGH	Multi-Axis Mask	Specifies axis to turn off	

Remarks

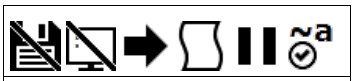
- The controller will continue to monitor the motor position
  - See the TP command for more details
- To turn the motor back on use the SH (Servo Here) command.
- The MO command is useful for positioning the motors by hand.

Examples

```
'Galil DMC Code Example
MO ;          'Turns off all motors
MO A;        'Turns off the A motor.
MO B;        'Turns off the B motor.
MO CA;       'Turns off the C and A motors.
SH ;          'Turns all motors on
axis= _MOA; 'Sets variable axis equal to the A-axis servo status
```

MO applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

MR *Reverse Motion to Position*



MRm= n

MR n,n,n,n,n,n,n,n

Usage	MRm= n	Arguments specified with a single axis mask and an assignment (=)
	MR n ...	Arguments specified with an implicit, comma-separated order

Description

This command will hold up the execution of subsequent code specified motor moves backward and crosses the position specified.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	-2,147,483,648	2,147,483,647	N/A	1	Value of position that must be crossed in the reverse direction	

Remarks

- MR command references absolute position.
- Although multiple positions can be specified, only one of the MR conditions must be satisfied for subsequent code execution.
- The MR command only requires an encoder and does not require that the axis be under servo control.
- The accuracy of the MR command is the number of counts that occur in 2*TM usec. Multiply the speed by 2*TM usec to obtain the maximum error.
  - Example with speed of 20,000 counts/second and TM of 1000 (1000 us).
    - Maximum error = 2 * 1000 E-6 seconds * 20,000 counts/second = 40 counts
- When using a stepper motor, this condition is satisfied when the stepper position (as determined by the output buffer - TD) has crossed the specified reverse motion position.

Examples

```
'Galil DMC Code Example
#test;'                  Program Test
 DP 0;'                   Define zero
 JG -1000;'              Jog mode (speed of 1000 counts/sec)
 BG A;'                  Begin move
 MR -3000;'              After passing the position -3000
 v1= _TPA;'               Assign V1 A position
 MG "Position is", v1;' Print Message
 ST ;'                    Stop
EN;'                     End of Program
```

MR applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

MT *Motor Type*

MTm= n

MT n,n,n,n,n,n,n,n

Usage	MTm= n	Arguments specified with a single axis mask and an assignment (=)
	MT n ...	Arguments specified with an implicit, comma-separated order
Operands	_MTm	Operand holds the value last set by the command

Description

The MT command selects the type of the motor and the polarity of the drive signal. Motor types include standard servomotors, which require a voltage in the range of +/- 10 Volts, and step motors, which require pulse and direction signals. The polarity reversal inverts the analog signals for servomotors, or inverts logic level of the pulse train for step motors.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	

Argument	Value	Description	Notes
n	1	Servo motor	Default
	-1	Servo motor with reversed polarity	Setting invalid for Galil sine drives
	1.5	PWM/Sign servo drive	
	-1.5	PWM/Sign servo drive with reversed polarity	
	2	Step motor with active low step pulses	
	-2	Step motor with active high step pulses	Valid setting for all Galil SDM stepper drives
	2.5	Step motor with reversed direction and active low step pulses	
	-2.5	Step motor with reversed direction and active high step pulses	Valid setting for all Galil SDM stepper drives

Remarks

- n = ? will return the value of the motor type for the specified axis.
- For step and direction modes (n=2,-2,2.5,-2.5), the auxiliary encoder input for the axis is no longer available.

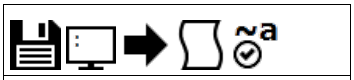
Examples

```
'Galil DMC Code Example
MT 1,-1,2,2;   'Configure A as servo, B as reverse servo, C and D as steppers
MT ?,?;        'Interrogate motor type for A- and B-axis
```

Error Number	Description	Cause
6	Number out of range	Argument value is not valid
135	Motor must be in MO	Axis must be in motor off before chaning MT
183	Not valid when EtherCAT network is up	MT cannot be set when EtherCAT is running (EU1)

MT applies to DMC500x0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,DMC18x6,DMC18x2

NB *Notch Bandwidth*



NBm= n
NB n,n,n,n,n,n,n,n

Usage	NBm= n	Arguments specified with a single axis mask and an assignment (=)
	NB n ...	Arguments specified with an implicit, comma-separated order
Operands	_NBm	Operand holds the value last set by the command

Description

The NB command sets real part of the notch poles. In other words, the NB controls the range of frequencies that will be attenuated.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	0	62.5	0.5	1/2	Value of the notch bandwidth in Hz	Max value dependent upon TM setting, see Remarks

Remarks

- _NBm contains the value of the notch bandwidth for the specified axis.
- NB also determines the ratio of NB/NZ which controls the attenuation, or depth, of the notch. See NZ for more details.
- See the NF command for recommendations on choosing NZ, NB, and NF values.

- See Application note #2431 for additional information on setting the NF, NB and NZ commands
  - http://www.galilmc.com/support/appnotes/optima/note2431.pdf

Maximum Range

- The maximum n argument is specified in Hz and is calculated by the equation below:

$$\frac{1}{(16 \times TM \times 10^{-6})}$$

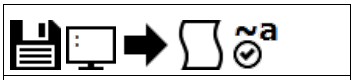
  - where TM is specified in microseconds.
  - The default TM is 1000, therefore default maximum NB value = 1/(16x1000E-6) = 62.5 Hz

Examples

```
'Galil DMC Code Example
NBA= 10;          'Sets the real part of the notch pole to 10/2 Hz
notch = _NBA;     'Sets the variable "notch" equal to the notch bandwidth value for the A axis
```

NB applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

NF *Notch Frequency*



NFm= n
NF n,n,n,n,n,n,n,n

Usage	NFm= n	Arguments specified with a single axis mask and an assignment (=)
	NF n ...	Arguments specified with an implicit, comma-separated order
Operands	_NFm	Operand holds the value last set by the command

Description

The NF command sets the frequency of the notch filter, which is placed in series with the PID compensation.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	0	250	0	1	Sets the frequency of the notch filter	Max value dependent upon TM setting, see Remarks

Remarks

- _NFm contains the value of notch filter for the specified axis.
- n = ? Returns the value of the Notch filter for the specified axis.
- n = 0 disables the notch.

- See Application note #2431 for additonal information on setting the NF, NB and NZ commands
  - http://www.galilmc.com/support/appnotes/optima/note2431.pdf

Chosing NF, NB, and NZ

1. A simple way for attaining NF, NB, and NZ parameters is to follow these simple rules:
   1. Estimate the resonance frequency (GalilTools Scope with cursors or Galil's FAS software)
   2. Set NF equal to the resonance frequency
   3. Set NB = 1/2NF
   4. Set NZ between 0 and 5
2. The ratio of NB/NF is extremly important. See the NB command for more details.

Maximum Range

- The maximum n argument is specified in Hz and is calculated by the equation below:

$$\frac{1\times10^{6}}{(4\times TM)}$$

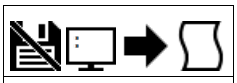
- Where TM is in microseconds.
  - Default TM is 1000, therefore default maximum value = 1E6/(4*1000) = 250 Hz

Examples

```
'Galil DMC Code Example
NF , 20;' Sets the notch frequency of B axis to 20 Hz
```

NF applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

NO *No Operation*



```
NO str
```

Operands	_NO	Operand has special meaning, see Remarks

Description

The NO command performs no action in a sequence and can be used as a comment in a program.

Arguments

Argument	Value	Description	Notes
str	String	A no action sequence used to document a program	Comments are limited to the maximum row size in a program. This will vary by controller.

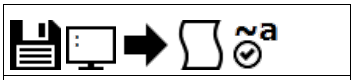
Remarks

- _NO returns a bit mask 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

Examples

```
'Galil DMC Code Example
#a;                    'Program A
NO;                    'No Operation
NO This Program;       'No Operation
NO This Program ;      'No Operation
NO Does Absolutely;    'No Operation
NO Nothing;            'No Operation
EN;                    'End of Program
```

NZ *Notch Zero*



NZm= n
NZ n,n,n,n,n,n,n,n

Usage	NZm= n	Arguments specified with a single axis mask and an assignment (=)
	NZ n ...	Arguments specified with an implicit, comma-separated order
Operands	_NZm	Operand holds the value last set by the command

Description

The NZ command sets the real part of the notch zero. In other words, the NB/NZ ratio controls the amount of attenuation, or depth, of the notch filter.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	0.5	62.5	0	0.5	Value of Notch Frequency in Hz	Max value dependent upon TM setting, see Remarks

Remarks

- See the NF command for recommendations on chosing NZ, NB, and NF values.
- The maximum n argument is determiend by the following equation

$$\frac{1}{(16 \times TM \times 10^{-6})}$$

- Where TM is in microseconds, the default TM is 1000.
- See Application note #2431 for additonal information on setting the NF, NB and NZ commands
  - http://www.galilmc.com/support/appnotes/optima/note2431.pdf

The NB/NZ Ratio

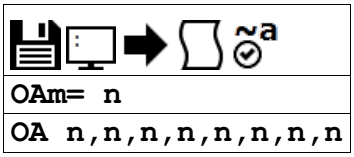
- The ratio, NB/NZ controls the amount of attenuation, or depth of the notch.
  - The larger the ratio of NB/NZ, the larger the attenuation, and vice versa.
- If NB/NZ > 1 the signal will amplifiy the output signal causing a resonance.
- NB = NZ essentially eliminates the notch

Examples

```
'Galil DMC Code Example
NZA = 10;' Sets the real part of the notch pole to 10/2 Hz
```

NZ applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

OA *Off on encoder failure*



OAm= n
OA n,n,n,n,n,n,n,n

Usage	OAm= n	Arguments specified with a single axis mask and an assignment (=)
	OA n ...	Arguments specified with an implicit, comma-separated order
Operands	_OAm	Operand holds the value last set by the command

Description

The OA command turns on or off encoder failure detection. The controller can detect a failure on either or both channels of the encoder. This is accomplished by checking on whether motion of less than 4 counts is detected whenever the torque exceeds a preset level (OV) for a specified time (OT).

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	0	1	0	1	Status of encoder failure detection	1 = enabled, 0 = disabled

Remarks

- The OA command works like the OE command: if OA is set to 1 and an encoder failure occurs, the axis goes into the motor off (MO) state and the stop code (SC) is set to 12 if detected during motion.
- The encoder failure detection will shut the motor off regardless of profiling status, but the stop code is not updated unless the axis is executing a profiled move at the time of the detection of the encoder failure.
- If included in the application program and OA is set to 1, #POSERR will run when an encoder failure is detected for the axis.
  - Note that for this function to work properly it is recommended to have a non-zero value for KI.

Examples

```
'Galil DMC Code Example
OAA= 1;' enable A axis encoder error detection
MG _OAA;'query OA value for A axis
```

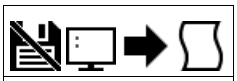
```
'Galil DMC Code Example
OA ,1;'  enable B axis encoder error detection
```

```
'Galil DMC Code Example
#setup
'setup the encoder error detection
OTA= 10;'  Set time to 10 milliseconds
OVA= 5;'   Set voltage to 5
OAA= 1;'   Enable encoder detection feature
EN
```

```
'Galil DMC Code Example
REM #POSERR example for checking to see if encoder failure occured
REM This procedure is needed because the stop code will only update if
REM the profilier is running at the time the encoder failure is detected.
#POSERR
~a= 0
#loop
IF _MO~a=1
 IF ((_TE~a<_ER~a)&(_OE~a)&(_OA~a))
  MG "possible encoder failure on ",~a{Z1.0}," axis"
 ENDIF
ENDIF
~a= ~a+1
JP #loop,~a<_BV
AI 1;'              wait for input 1 to go high
SH ;'               enable all axes
RE
```

OA applies to DMC40x0,DMC42x0,DMC41x3,DMC18x6,DMC30010,DMC500x0

OB *Output Bit*



OB n,ex

Usage	OB n ...	Arguments specified with an implicit, comma-separated order

Description

The OB command allows variable control of an output bit based on logical expressions. The OB n, logical expression command defines output bit i as either 0 or 1 depending on the result from the logical expression.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	1	80	0	1	Output bit specified	Outputs 9-16 only valid on 5-8 axis controller. See Remarks.
ex	N/A	N/A	N/A	Expression	Expression that defines status of output	If ex is true/non-zero, set output to 1. If ex is false/zero, set output to 0

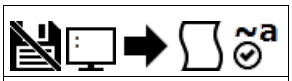
Remarks

- An expression is any valid logical expression, variable or array element.
- Any non-zero value of the expression results in a one set to the output bit.
- Extended IO must be configured as outputs by the CO command for proper operation with the OB command.

Examples

```
'Galil DMC Code Example
OB 1, pos;'           If pos<>0, Bit 1 is high.
'                     If pos=0, Bit 1 is low
OB 2, @IN[1]&@IN[2];' If Input 1 and Input 2 are both high, then
'                     Output 2 is set high
OB 3, count[1];' If the element 1 in the array is zero, clear bit 3
OB n, count[1];' If element 1 in the array is zero, clear bit n
```

OC *Output Compare*



OCm = n_0, n_1

Usage	OCm= n	Arguments specified with a single axis mask and an assignment (=)
Operands	_OC	Operand has special meaning, see Remarks

Description

The OC command sets up the Output Compare feature, also known as Pulse on Position. The controller has a special digital output which can be configured to pulse on a specified absolute encoder position, and optionally on a delta encoder change after that. These operations are known as one-shot and circular compare, respectively.

Each set of 4 axes, ABCD and EFGH, has one digital output which can be configured to this mode of operation

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to enable output compare	Axes A-D share one output compare, axes E-H share a second output compare output
n_0	-2,147,483,648	2,147,483,647	N/A	1	Absolute encoder position of first pulse	n_0 must be within 65535 counts of current position
n_1	-65,536	65,535	N/A	1	Incremental encoder distance between pulses	0 indicates single-shot pulse in positive direction, -65536 indicates single shot when moving in the negative direction

Remarks

- For controllers with 5-8 axes, two output compares are available. One for the A-D axes, the other for the E-H axes

One shot Compare Mode:

- The output compare signal will go low, and stay low at a specified absolute encoder position.
- This is done by specifying n_1 as 0 for positive motion, and -65536 for negative motion

Circular Compare Mode:

- After the absolute position of the first pulse (n_0), the cicular compare can be configured to pulse low at a relative distance thereafter (n_1).
- This is done by specifying n_1 to a non-zero delta position (range of -65535 to 65535)
    - OCA = 0 will disable the Circular Compare function on axes A-D.
    - OCE = 0 will disable the Circular Compare function on axes E-H.

Limitations

- The Output Compare function is only valid with incremental encoders.
    - The Output Compare function is not valid with SIN/COS (AF settings of 5-12), standard analog (AF setting of 1), BiSS or SSI feedback (SS or SI commands).
- The OC function cannot work for an axis configured as a stepper.
- The auxiliary encoder of the corresponding axis cannot be used when in this mode.
    - Dual loop mode (which uses the aux encoder input) will not operate when the OC command is enabled.
- The OC function requires that the main encoder and auxiliary encoders be configured exactly the same (see the command, CE). For example: CE 0, CE 5, CE 10, CE 15.
- OC only requires an encoder, and is independent of axis tuning, and motion profiling.

Operand Usage

- _OC contains the state of the OC function.
    - _OC = 0 : OC function has been enabled but not generated any pulses.
    - _OC = 1: OC function not enabled or has generated the first output pulse.
- On a 5-8 axis controller, _OC is a logical AND of axes A-D and E-H.

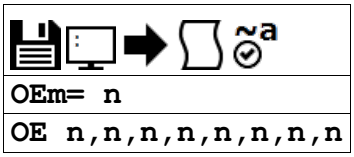
Examples

```
'Galil DMC Code Example
OCA= 300,100;' Select A encoder as position sensor.
REM First pulse at 300. Following pulses at 400, 500, 600 ...
```

```
'Galil DMC Code Example
REM Output compare can be used to create raster scans.
REM By using cicular compare on one axis, followed by an index move on a perpindicular axis
REM raster patterns are easily made.
REM The following image shows a rastered "dot matrix" type image easily created
REM with output compare and a laser on a two dimensional stage.
```

OC applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

OE *Off-on-Error*



OEm= n
OE n,n,n,n,n,n,n,n

Usage	OEm= n	Arguments specified with a single axis mask and an assignment (=)
	OE n ...	Arguments specified with an implicit, comma-separated order
Operands	_OEm	Operand holds the value last set by the command

Description

The OE command sets the Off On Error function for the controller. The OE command causes the controller to shut off the motor command if a position error exceeds the limit specified by the ER command, an abort occurs from either the abort input or on AB command, or an amplifier error occurs based on the description of the TA command.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	0	0	0	0	Disables the Off On Error Function	Default
	1	1	0	0	Motor shut off by position error, amplifier error or abort input	
	2	2	0	0	Motor shut off by hardware limit switch	
	3	3	0	0	Motor shut off by position error, amplifier error, abort input or by hardware limit switch	

Remarks

- For any value of OE <> 0, the axis will be shut off due to amplifier faults on any amplifier axis. See the TA command for conditions of an amplifier fault.
- BR1 must be enabled when internal brushless servo amplifiers are installed but the axis is driven with an external amplifier. BR1 disables hall error checking when OE <> 0
    - Examples of brushless servo amps that require this consideration include the AMP-43040 (-D3040) or the AMP-20540
- Motion Behavior:
    - If an error or axis-specific abort is detected, and the motion was executing an independent move, only that axis will be shut off.
    - If the motion is a part of coordinated mode of the types GM, VM, LM or CM, all participating axes will be stopped.

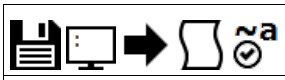
Examples

```
'Galil DMC Code Example
: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
:MG _OEA;' Query A axis OE setting
 1.0000
```

```
'Galil DMC Code Example
#main
'code to enable the OE command for all error conditions
'and setup the corresponding automatic subroutines
'to display relevent data
'no loop for abort input, as that stops code operation
OE 3,3,3,3
SH ABCD
JG*= 1000;' all jog at 1000
BG ABCD
#loop
'endless loop
WT 1000
JP #loop
EN

#AMPERR
MG "amplifier fault"
MG _TA0,_TA1,_TA2,_TA3
EN

#POSERR
MG "position error fault"
MG _TEA,_TEB,_TEC,_TED
EN

#LIMSWI
MG "limit switch fault"
MG _TSA,_TSB,_TSC,_TSD
EN
```

OE applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

OF *Offset*



OFm= n
OF n,n,n,n,n,n,n,n

Usage	OFm= n	Arguments specified with a single axis mask and an assignment (=)
	OF n ...	Arguments specified with an implicit, comma-separated order
Operands	_OFm	Operand holds the value last set by the command

Description

The OF command sets a bias voltage in the command output or returns a previously set value.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	-9.9982	9.9982	0	20/65,536	Offset voltage applied to MCMD	

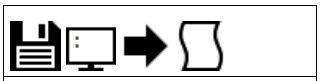
Remarks

- This can be used to counteract gravity or an offset in an amplifier.

Examples

```
'Galil DMC Code Example
:OF 1,-2,3,5;' Set A-axis offset to 1, the B-axis offset to -2, the C-axis to 3, and the D-axis to 5
:OF -3;'  Set A-axis offset to -3  Leave other axes unchanged
:OF ,0;'  Set B-axis offset to 0  Leave other axes unchanged
:OF ?,?,?,?;' Return offsets
-3.0000,0.0000,3.0000,5.0000
:OF ?;'  Return A offset
-3.0000
:OF ,?;' Return B offset
0.0000
```

OP *Output Port*



OP n_0,n_1,n_2,n_3,n_4

Usage	OP n ...	Arguments specified with an implicit, comma-separated order
Operands	_OP0 _OP1 _OP2 _OP3 _OP4	Operand holds the value last set by the command

Description

The OP command sets the output ports of the controller in a bank using bitmasks. Arguments to the OP command are bit patterns (decimal or hex) to set entire banks (bytes) of digital outputs. Use SB, CB or OB to set bits individually.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n_0	0	65,535	0	1	Decimal representation: General Outputs 1-16	On a 1-4 axis controller, max is 255 ($FF) for outputs 1-8 only.
n_1	0	65,535	0	1	Decimal representation: Extended Output (Bank 2,3)	DB-14064 required for Extended IO support
n_2	0	65,535	0	1	Decimal representation: Extended Output (Bank 4,5)	
n_3	0	65,535	0	1	Decimal representation: Extended Output (Bank 6,7)	
n_4	0	65,535	0	1	Decimal representation: Extended Output (Bank 8,9)	

Remarks

- Bit patterns for extended I/O banks (where available) configured as inputs have no affect on the IO status.

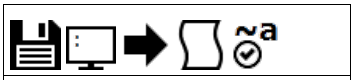
Output Mapping Examples

Example	Command Issued (Hex version)	Bits Set	Bits Cleared
1-4 axis Set all outputs	OP255 (OP$FF)	1-8	-
5-8 axis Set all outputs	OP65535 (OP$FFFF)	1-16	-
Clear all outputs	OP0 (OP$0000)	-	1-16
Alternating on/off	OP43690 (OP$AAAA)	2,4,6,8,10,12,14,16	1,3,5,7,9,11,13,15
Set High Byte	OP65280 (OP$FF00)	9-16	1-8
Set Low Byte	OP255 (OP$00FF)	1-8	9-16

Examples

```
'Galil DMC Code Example
OP 0;'   Clear Output Port -- all bits
OP $85;' Set outputs 1,3,8 and clear the others
MG _OP0;' Returns the parameter "n0"
```

OT *Off on encoder failure time*



OTm= n
OT n,n,n,n,n,n,n,n

Usage	OTm= n	Arguments specified with a single axis mask and an assignment (=)
	OT n ...	Arguments specified with an implicit, comma-separated order
Operands	_OTm	Operand holds the value last set by the command

Description

The OT command sets the timeout time for the encoder failure routine. The command sets the time in samples that the encoder failure will wait for motion after the OV threshold has been exceeded. The controller can detect a failure on either or both channels of the encoder.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	1	32,000	30	1	Number of samples for error detection	

Remarks

- Encoder error detection is based on whether motion of at least 4 counts is detected whenever the torque exceeds a preset level (OV) for a specified time (OT).
  - Note that for this function to work properly it is necessary to have a non-zero value for KI.
- See the OA command for more details on this error detection mode

Examples

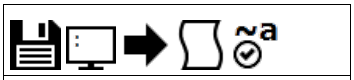
```
'Galil DMC Code Example
OTD= 400;'   Set D axis encoder error timeout to 400 samples
OT 100,200;' Set A axis to 100 and B axis to 200 sample timeouts
```

```
'Galil DMC Code Example
#setup
OTA= 10;'  Set time to 10 milliseconds
OVA= 5;'   Set voltage to 5
OAA= 1;'   Enable encoder detection feature
EN
```

```
'Galil DMC Code Example
REM #POSERR example for checking to see if encoder failure occured
REM This procedure is needed because the stop code will only update if
REM the profilier is running at the time the encoder failure is detected.
#POSERR
~a= 0
#loop
IF _MO~a=1
 IF ((_TE~a<_ER~a)&(_OE~a)&(_OA~a))
  MG "possible encoder failure on ",~a{Z1.0}," axis"
 ENDIF
ENDIF
~a= ~a+1
JP #loop,~a<_BV
AI 1;'              wait for input 1 to go high
SH ;'               enable all axes
RE
```

OT applies to DMC40x0,DMC42x0,DMC41x3,DMC18x6,DMC30010,DMC500x0

OV *Off on encoder failure voltage*



OVm= n
OV n,n,n,n,n,n,n,n

Usage	OVm= n	Arguments specified with a single axis mask and an assignment (=)
	OV n ...	Arguments specified with an implicit, comma-separated order
Operands	_OVm	Operand holds the value last set by the command

Description

The OV command sets the threshold voltage for detecting an encoder failure. The controller can detect a failure on either or both channels of the encoder.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	0	9.9982	0.9438	20/65,536	Torque voltage to trigger encoder error detection	

Remarks

- Encoder error detection is accomplished by checking on whether motion of at least 4 counts is detected whenever the torque exceeds a preset level (OV) for a specified time (OT).
  - Note that for this function to work properly it is recommended to have a non-zero value for KI.
- The value of OV should be high enough to guarantee that the motor would overcome any static friction in the system. If it is too low, there will be false triggering of the error condition.
- The OV value may not be higher than the TL value.
- See the OA command for more details on this error detection mode

Examples

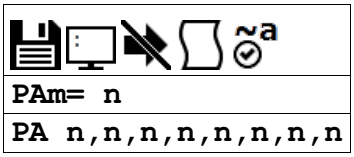
```
'Galil DMC Code Example
OVB= 1.2;' Set B axis encoder detection torque value to 1.2V
OV 0.54;'  Set A axis encoder detection torque value to 0.54V
```

```
'Galil DMC Code Example
#setup
'setup the encoder error detection
OTA= 10;'  Set time to 10 milliseconds
OVA= 5;'   Set voltage to 5
OAA= 1;'   Enable encoder detection feature
EN
```

```
'Galil DMC Code Example
REM #POSERR example for checking to see if encoder failure occured
REM This procedure is needed because the stop code will only update if
REM the profilier is running at the time the encoder failure is detected.
#POSERR
~a= 0
#loop
IF _MO~a=1
 IF ((_TE~a<_ER~a)&(_OE~a)&(_OA~a))
  MG "possible encoder failure on ",~a{Z1.0}," axis"
 ENDIF
ENDIF
~a= ~a+1
JP #loop,~a<_BV
AI 1;'              wait for input 1 to go high
SH ;'               enable all axes
RE
```

OV applies to DMC40x0,DMC42x0,DMC41x3,DMC18x6,DMC30010,DMC500x0

PA *Position Absolute*



PAm= n

PA n,n,n,n,n,n,n,n

Usage	PAm= n	Arguments specified with a single axis mask and an assignment (=)
	PA n ...	Arguments specified with an implicit, comma-separated order
Operands	_PAm	Operand has special meaning, see Remarks

Description

The PA command sets the end target of the Position Absolute Mode of Motion.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
	M	N	N/A	Axis	Virtual axis to assign value	
n	-2,147,483,648	2,147,483,647	0	1	Absolute position target for independant move	n=? returns the commanded position at which motion last stopped

Remarks

- The position is referenced to the absolute zero position, defined as position 0.
- By default a new PA command may not be issued before the previous PA command has finished executing. This operation may be changed by running in Position Tracking Mode - See the PT command for more information.

Operand Usage

- _PAm contains the last commanded position at which motion stopped.

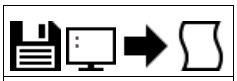
Examples

```
'Galil DMC Code Example
: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 ;'                    Execute Motion
:PA ?,?,?,?;'            Returns the current commanded position after motion has completed
400, -600, 500, 200
: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.
```

```
'Galil DMC Code Example
DP 10000;'    set current position to 10000
PA 3000;'     move to absolute position 3000, which is a -7000 count move
BG A;'        begin -7000 count move
EN
```

PA applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

PF *Position Format*



PF $n_0.n_1$

Usage	PF n ...	Arguments specified with an implicit, comma-separated order
Operands	_PF	Operand holds the value last set by the command

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 decimal 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.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n_0	-8	10	10	1	Number of places displayed preceding the decimal point	Negative numbers force data to display in hexadecimal format
n_1	0	4	0	1	Number of places displayed after the decimal point	

Remarks

- If PF is minus, the format will be hexadecimal and a dollar sign will precede the characters. Hex numbers are displayed as 2's complement with the first bit used to signify the sign.
- If a number exceeds the format, the number will be displayed as the maximum possible positive or negative number (i.e. 999.99, -999, $8000 or $7FF).
- The PF command formats the values returned from the following commands:

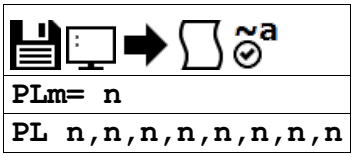
BL?	IP ?	TD
DE ?	LE ?	TE
DP ?	PA ?	TN
EM ?	PR ?	TP
FL ?	RL	VE
GP	RP	

Examples

```
'Galil DMC Code Example
:DP 21;'  Set position of A axis for example
:TP A;'   Tell position of A in default format
 21
:PF 5.2;' Change format to 5 digits of integers and 2 of decimal
:TP A
 21.00
:PF -5.2;' Change format to hexadecimal
:TP A
$00015.00
```

PF applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

PL *Pole*



PLm= n

PL n,n,n,n,n,n,n,n

Usage	PLm= n	Arguments specified with a single axis mask and an assignment (=)
	PL n ...	Arguments specified with an implicit, comma-separated order
Operands	_PLm	Operand holds the value last set by the command

Description

The PL command adds a low-pass filter in series with the PID compensation.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	0	0.9999	0	2/65,536	Value used to generate pole filter crossover frequency	See Remarks for the equation used. n = 0 disables the Pole filter

Remarks

- The digital transfer function of the filter is (1 - n) / (Z - n) and the equivalent continuous filter is A/(S+A) where A is the filter cutoff frequency: A=(1/T) ln (1 / n) rad/sec and T is the sample time.

Calculated Pole

- To convert from the desired crossover (-3 dB) frequency in Hertz to the value given to PL, use the following formula

$$n = e^{-T \bullet f_c \bullet 2\pi}$$

- where
  - n is the argument given to PL (less than 1)
  - T is the controller's servo loop sample time in seconds (TM divided by 1,000,000)
  - Fc is the crossover frequency in Hertz
- Example: Fc=36Hz TM=1000 n=e^(-0.001*36*2*pi) =0.8
- The following shows several example crossover frequencies achieved with various values of PL

n	Fc (Hz)
0	Infinite (off)
0.2	256
0.4	145
0.6	81
0.8	36
0.999	0

Examples

```
'Galil DMC Code Example
'Set A-axis Pole to 0.95, B-axis to 0.9, C-axis to 0.8, D-axis pole to 0.822
:PL .95,.9,.8,.822
Query all Pole values
:PL ?,?,?,?
0.9527,0.8997,0.7994,0.8244
Return A Pole only
:PL ?
0.9527
```

PL applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

PR *Position Relative*

PRm= n

PR n,n,n,n,n,n,n,n

Usage	PRm= n	Arguments specified with a single axis mask and an assignment (=)
	PR n ...	Arguments specified with an implicit, comma-separated order
Operands	_PRm	Operand holds the value last set by the command

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

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
	M	N	N/A	Axis	Virtual axis to assign value	
n	-2,147,483,648	2,147,483,647	N/A	1	Incremental distance for independent move	n = ? returns the current incremental distance specified

Remarks

- _PRm contains the current incremental distance for the specified axis.

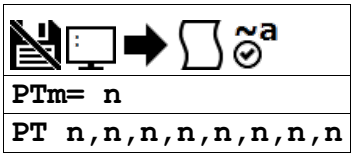
Examples

```
'Galil DMC Code Example
: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.
```

```
'Galil DMC Code Example
'using PA/PR, you can query PR for the incremental distance
:DP 10000
:PA 8000
:PR ?
-2000
```

PR applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

PT *Position Tracking*



PTm= n

PT n,n,n,n,n,n,n,n

Usage	PTm= n	Arguments specified with a single axis mask and an assignment (=)
	PT n ...	Arguments specified with an implicit, comma-separated order
Operands	_PTm	Operand holds the value last set by the command

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 that begin motion immediately without requiring a BG command. The absolute position may be specified such that the axis will begin motion, continue in the same direction, reverse directions, or decelerate to a stop

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	0	1	0	1	Setting for position tracking mode of motion	n = 1 enables PT mode, n = 0 disables PT mode

Remarks

- The PA command is used to give the controller an absolute position target. Motion commands other than PA are not supported in this mode.
- The motion profile is trapezoidal with the parameters controlled by acceleration, deceleration, and speed (AD, DC, SP).
- When in the PT mode the ST command will exit the mode.
- The AM and MC trip points are not valid in this mode.
  - MF and MR are recommended with this mode as they allow the user to specify both the absolute position, and the direction. The AP trip point may also be used.
- Position Tracking is not valid on virtual axes

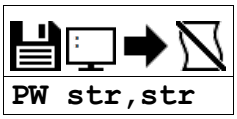
Examples

```
'Galil DMC Code Example
DPA= 0;'   Start position at absolute zero
PTA= 1;'   Start PT mode on A axis
PA 1000;'  Move to position 1000, motion starts right away
MF 500;'   Wait till position 500 reached
PA -1000;' Reverse direction to move to position -1000
EN
```

```
'Galil DMC Code Example
#a
 PT 1,1,1,1;'        Enable the position tracking mode for axes A, B, C, and D
'                    NOTE: The BG command is not used to start the PT mode.
#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.
'
 PA v1,v2,v3,v4;'    Command ABCD axes to move to absolute positions.  Motion
'                    begins when the command is processed.  BG is not used
'                    to begin motion in this mode.  In this example, it is
'                    assumed that the user is updating the variable at a
'                    specified rate.  The controller will update the new
'                    target position every 10 milliseconds (WT10).
 WT 10;'             Wait 10 milliseconds
JP #loop;'           Repeat by jumping back to label LOOP
```

PT applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

PW *Password*



Usage	PW n ...	Arguments specified with an implicit, comma-separated order

Description

The PW command sets the password used to lock the controller. Locking the controller prevents interrogation of the controller program space.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
str	0 chars	8 chars	""	String	String to be used for password	Both parameters must match for the PW command to succeed.

Remarks

- The password can only be changed when the controller is in the unlocked state. See the ^L^K for more details.
- The password is burnable but cannot be interrogated. If you forget the password and the controller is locked you must master reset the controller to gain access.
- Quotes are not used to frame the password string. If quotes are used, they are part of the password.

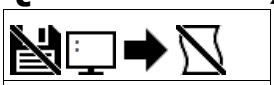
Examples

```
'Galil DMC Code Example
:PW apple,orange
?
:TC 1
138 Passwords not identical
:PW apple,apple
:^L^K apple,1
```

```
'Galil DMC Code Example
:PW test,test;'            Set password to "test"
:^L^K test,1;'            Lock the program
:ED;'                     Attempt to edit program
?
:TC 1
106 Privilege violation
```

PW applies to DMC40x0,DMC42x0,DMC41x3,RIO,DMC18x6,DMC30010,DMC500x0

QD *Download Array*



QD str[],n_0,n_1

Usage	QD n ...	Arguments specified with an implicit, comma-separated order

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.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
str	1 char	7 chars	N/A	String	Name of array to receive data via download.	
n_0	0	see Notes	0	1	Index of the first array element.	Value cannot exceed size of array - 2
n_1	1	see Notes	see Notes	1	Index of the last array element.	Value cannot exceed size of array - 1. Defaults to size of array - 1.

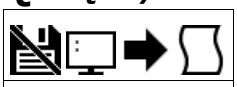
Remarks

- Array name must be a valid, dimensioned array name followed by empty [] brackets.
- The array elements can be separated by a comma (,) or by CR/LF.
- The downloaded array is terminated by a \ character.
- QD is not supported in the Galiltools terminal
  - It is recommended to use the array download functions available through the GalilTools software and drivers rather than directly using the QD command.

Examples

```
'Galil DMC Code Example
:'From a character-buffered terminal such as Telnet or Hyperterm
:DM array[3]
:QD array[]
1,2,3\:LA
array[3]
:array[0]= ?
 1.0000
:array[1]= ?
 2.0000
:array[2]= ?
 3.0000
:
```

QP *Query Parameters*



QP

Usage	QP	Command takes no arguments

Description

Returns memory information for the controller.

Arguments

QP takes no arguments.

Remarks

- Each row of the QP response describes a parameter characteristic of the controller.
- The data is provided in a comma separated list starting with a fixed ID string.
- QP is only valid on firmware Rev 1.1a and later.

QP response row descriptions

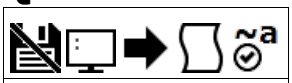
Row ID	Field 1	Field 2	Field 3	Field 4	Description of the row
"PR"	characters per line	number of lines	flash=1, ram=0	N/A	Determines the dimensions of the program and the runtime location of the program.
"VA"	number of array elements	number of arrays	number of variables	number of labels	Determines the dimensions of the variables and arrays.

Examples

```
'Galil DMC Code Example
:QP;' only valid on firmware Rev 1.1a and later
PR, 80, 4000, 0
VA, 24000, 30, 510, 510
```

QP applies to DMC500x0,DMC40x0,DMC42x0,DMC41x3,DMC30010,RIO,DMC18x6

QR *I O Data Record*



QR mm

QR

Usage	QR mm	Argument is an axis mask

Description

The QR command causes the controller to return a record of information regarding controller status.

This status information includes 4 bytes of header information and specific blocks of information as specified by the command arguments. The details of the status information is described in Chapter 4 of the user's manual.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
mm	A	ABCDEFGHSTI	ABCDEFGHSTI	Multi-Axis Mask	Axes/Coordinated/IO data specified to display in the data record	If no argument entered, mm = "ABCDEFGHSTI"

Argument	Value	Description	Notes
mm	A-H	Output axes A-H data record block	
	S	Output coordinated axis S data block	
	T	Output coordinated axis T data block	
	I	Output General IO data block	

Remarks

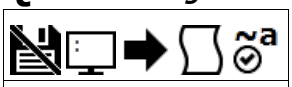
- The data returned by the QR command is in binary format and is unreadable in programs such as Galiltools.
  - The Galiltools API has specialized commands to parse the data record packet. See the Galiltools User Manual for more details.

Examples

```
'Galil DMC Code Example
QR A;'   Return the data record with A axis block only
QR BI;'  Return the data record with B axis block and IO block
QR ST;'  Return the data record with S and T coordinated axis blocks
QR ;'    Return the data record for all axes, including IO and S and T axis blocks
```

QR applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,RIO,DMC18x6,DMC18x2,DMC30010,DMC500x0

QS *Error Magnitude*



QS mm

Usage	QS mm	Argument is an axis mask
Operands	_QSm	Operand has special meaning, see Remarks

Description

The QS command reports the magnitude of error, in drive step counts, for axes in Stepper Position Maintenance mode. A step count is directly proportional to the micro-stepping resolution of the stepper drive.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
mm	A	ABCDEFGH	ABCDEFGH	Multi-Axis Mask	Axes to query for step motor error magnitude	Default value used if mm is undefined.
m	A	H	N/A	Axis	Single Axis to query for error magnitude	

Remarks

- The result of QS is modularized so that result is never greater than 1/2 the revolution of the stepper motor.
  - Largest possible QS result = 0.5*YA*YB
- If present in embedded code, command execution will jump to #POSERR when QS is equal to 3 full motor steps (_YAm * 3)
- QSm=? will return the current error for axis m

Operand Usage

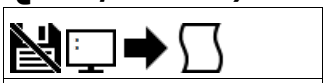
- _QSm contains the error magnitude in drive step counts for the specified axis.

Examples

```
'Galil DMC Code Example
'For an SDM-20620 microstepping drive, query the error of B axis:
:QS B
 253
:' Above shows 253 step counts of error.
:' The SDM-20620 resolution is 64 microsteps per full motor step
:' nearly four full motor steps of error.
Query the value of all axes:
:QS
 0,253,0,0,0,0,0,0
:' Response shows all axes error values
```

QS applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

QU *Upload Array*



QU str[],n_0,n_1,n_2

Usage	QU n ...	Arguments specified with an implicit, comma-separated order

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.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
str	1 char	7 chars	N/A	String	Name of array to be uploaded	
n_0	0	see Notes	0	1	Index of first array element	Value cannot exceed size of array - 2
n_1	1	see Notes	see Notes	1	Index of last array element	Defaults to last element of array. Value cannot exceed size of array - 1
n_2	0	1	0	1	Selects character delimiter between array elements	n_2 = 0 selects CR delimiting. n_2 = 1 select comma delimiting.

Remarks

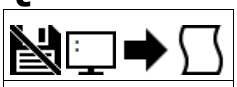
- Array name must be a valid, dimensioned array name followed by empty [] brackets.
- The uploaded array will be followed by a <control>Z as an end of text marker.
- The GalilTools array upload functions can be used to upload array data in .csv format.

Examples

```
'Galil DMC Code Example
DM test[10];'     Dimension a 10 element sized array
QU test[],0,1,1;' Upload first 2 elements
QU test[],8,9,1;' Upload last 2 elements (size-2 and size-1 used for n1,n2)
EN
```

```
'Galil DMC Code Example
:DM array[5];'                          Dimension Array
:QU array[],0,4,1;'                     Upload Array
0.0000, 0.0000, 0.0000, 0.0000, 0.0000
:array[0]= 9;'                           Set value
:array[1]= 1
:QU array[],0,4,1
9.0000, 1.0000, 0.0000, 0.0000, 0.0000
:array[0]= ?;'                           Alternative method to return just one array value
 9.0000
```

QZ *Return Data Record information*



QZ

Usage	QZ	Command takes no arguments

Description

The QZ command is an interrogation command that returns information regarding the data record. The controller's response to this command will be the return of 4 integers separated by commas.

Arguments

QZ is an interrogation command with no parameters.

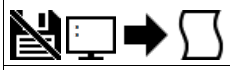
Remarks

- The four fields returned by QZ represent the following:
  1. First field returns the number of axes.
  2. Second field returns the number of bytes to be transferred for general status
  3. Third field returns the number of bytes to be transferred for coordinated move status
  4. Fourth field returns the number of bytes to be transferred for axis specific information

Examples

```
'Galil DMC Code Example
:QZ;'              standard DMC-4143 example response
 4, 52, 26, 36
```

RA *Record Array*



RA str[ ],str[ ],str[ ],str[ ],str[ ],str[ ],str[ ],str[ ]

Usage	RA n ...	Arguments specified with an implicit, comma-separated order

Description

The RA command selects the user arrays to be populated by the Record Array function. The data to be captured is specified by the RD command and time interval by the RC command.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
str	1 char	7 chars	N/A	String	Valid array name to use in record array function	The arrays listed correspond to the source list defined by the RD command. See Remarks

Remarks

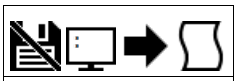
- The array name str must be followed by the [] brackets. Those brackets must be empty.
- The array name str must be a valid array defined by the DM command and reported by LA.

Examples

```
'Galil DMC Code Example
'  try to start record array without defining array[]
:RA array[]
?
:TC 1
82 Undefined array
:DM array[100]
:RA array[]
```

```
'Galil DMC Code Example
#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

'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.

'The GalilTools Realtime scope can often be used as an alternative to record array.
```

RC *Record*



RC n_0, n_1

Usage	RC n ...	Arguments specified with an implicit, comma-separated order
Operands	_RC	Operand has special meaning, see Remarks

Description

The RC command begins recording for the Automatic Record Array Mode. RC 0 stops recording. The record array mode loads source data specified by the RD command into the arrays defined by the RA command. The address for the array element for the next recording can be interrogated with _RD.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n_0	0	8	0	1	Specify the record array time interval as 2^n samples.	n_0 = 0 stops recording.
n_1	see Notes	see Notes	0	1	Specify the number of records to perform	n_1 has special rules for the maximum setting. See Remarks.

Remarks

- Firmware Note: Do not allocate or deallocate arrays (DM,DA) while the Automatic Record Array Mode is running.
- GalilTools Note: Do not download arrays from GalilTools, or call the arrayDownload() or arrayDownloadFile() functions while automatic record array mode is running.

- n_0 = non zero number automatically starts record mode.
- n_0 = ? returns status of recording. '1' if recording, '0' if not recording.

Second Parameter Rules

- n_1 specifies the last array element to use for record mode.
- If arrays specified by RA have different sizes, the smallest array size is the maximum value for n_1
- If n_1 = 0 or not specified, the maximum value is used.
- A negative value for n_1 specifies circular (continuous) record over array addresses 0 to (n_1-1).
  - The absolute value of the minimum n_1 allowed = maximum n_1 allowed

Operand Usage

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

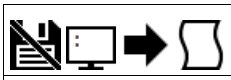
Setting up the record array mode

1. Dimension an array/arrays for storing data. Make sure you dimension the array with the number of elements required to capture data for your application.
2. Set the RA command with the arrays to be used for recording
3. Set the RD command with the data sources to be applied to the arrays. The order of your arrays entered into RA will match the order of data sources set by RD
4. Set the RC command to get the desired time between records and enable the recording.
5. Monitor the _RC operand for a 0 to indicate recording is done.
6. View the data in your embedded code, or extract the data using Galiltools software and the Upload array function.

Examples

```
'Galil DMC Code Example
#record;'              Record label
DM torque[1000];'      Define Array
RA torque[];'          Specify Array to record data
RD _TTA;'              Specify Data Type
RC 2;'                 Begin recording and set 4 msec between records
JG 1000;BG ;'           Begin motion
#a;JP #a,_RC=1;'       Loop until done
MG "DONE RECORDING";' Print message
EN;'                   End program
```

RD *Record Data*



```
RD arg, arg, arg, arg, arg, arg, arg, arg
```

Usage	RD n ...	Arguments specified with an implicit, comma-separated order
Operands	_RD	Operand has special meaning, see Remarks

Description

The RD command specifies the data type to be captured for the Record Array (RA) mode. The data defined in this command is stored in arrays defined by the RA command at the time interval specified with the RC command.

Arguments

Valid arguments for RD command

Argument	Value	Description	Notes
arg	TIME	Time in servo samples	Value as read by the TIME command
	_AFm	Analog input digital value	Data range is -32768 to 32767. The analog inputs are limited to those which correspond to an axis on the controller. **Syntax Note:** Unlike the operand _AFm, the symbol _AFm in the context of RD records the ADC value, not the AF setting.
	_DEm	2nd encoder position	
	_TPm	Encoder position	
	_TEm	Position error	
	_RPm	Commanded position	_RPm and _SHm capture the same data
	_SHm	Commanded position	_RPm and _SHm capture the same data
	_RLm	Latched position	
	_TI	Input status	
	_OP	Output status	
	_TSm	Switches	Only bits 0-4 valid
	_SCm	Stop code	
	_TTm	Torque command	The values recorded for torque are in the range of +/- 32767 where 0 is 0 torque, -32767 is -10 volt command output, and +32767 is +10 volt.
	_TVm	Filtered velocity	This value will be 64 times greater than TV command
	_TDm	Stepper position	

Remarks

- Arguments listed as _XXm are valid when m is a valid axis mask
- The order of args specified in RD corresponds with the array order specified in the RA command.
- the operand _RD contains the address for the next array element for recording.
- When recording _AFm, the returned value is signed. This means that when AQ is used to set unipolar inputs, values on the upper half of the voltage range are sign extended. Anding the value with $0000FFFF will return the expected unsigned value.

Examples

```
'Galil DMC Code Example
DM errora[50],errorb[50];'      Define arrays
RA errora[],errorb[];'          Specify arrays to be recorded
RD _TEA,_TEB;'                  Specify data source
RC 1;'                          Begin recording, period is once every other servo sample
JG 1000;BG ;'                    Begin motion

'The GalilTools Realtime scope can often be used as an alternative to record array.
```

RE *Return from Error Routine*

RE n

Usage	RE n ...	Arguments specified with an implicit, comma-separated order

Description

The RE command is used to end subroutines in application code. An RE at the end of these routines causes a return to the main program. Specific automatic error subroutines require the use of the RE command to end the code correctly.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	0	1	0	1	Determines state of interrupted trippoint when returning from an automatic subroutine.	n = 1 restores the interrupted trippoint. n = 0 clears the trippoint

Remarks

- The RE command is used to end the following error automatic subroutines.

Automatic Subroutines Used	Notes
#LIMSWI	
#POSERR	
#SERERR	Only when equipped with serial encoder firmware support

- Care should be taken to ensure the error conditions are cleared when finishing the subroutine to avoid immediate re-entering of the error routine.
- To avoid returning to the main program on an interrupt, use the ZS command to zero the subroutine stack, then use JP to return to the desired location in code.
- RE 1 restores the trippoint that was interrupted by an automatic subroutine (like WT)
    - A motion trippoint like MF or MR requires the axis to be actively profiling in order to be restored with the RE 1 command.

Examples

```
'Galil DMC Code Example
REM dummy loop
#a
JP #a
EN

#POSERR;'       Begin Error Handling Subroutine
MG "ERROR";'    Print message
SB 1;'          Set output bit 1
RE;'            Return to main program and clear trippoint
```

REM *Remark*

```
REM str
```

Description

REM is used for comment lines. 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.

NO (or ') should be used instead of REM for commenting in application code unless speed or program space is an issue.

Arguments

Argument	Value	Description	Notes
str	String	Comment to be removed from code prior to download	This comment is not limited by the character limit of the controller, as it is never downloaded

Remarks

- REM differs from NO (or ') in the following ways:
  - 1. NO (or ') comments are downloaded to the controller and REM comments aren't
  - 2. NO (or ') 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 (or ') comments are recovered. Thus the uploaded program is less readable with REM.
  - 4. NO (or ') 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 (or ') can be used to place comments to the right of code (after a semicolon) on the same line

Special Strings

- REM DISABLE COMPRESSION
  - Inserting this line into the beginning of your application code disables Galiltools download compression utility. This is not a controller function.

Examples

```
'Galil DMC Code Example
REM This comment will be stripped when downloaded to the controller
'This comment will be downloaded and takes some execution time
PRA= 1000 ;'this comment is to the right of the code
```

RI *Return from Interrupt Routine*

RI n

Usage	RI n ...	Arguments specified with an implicit, comma-separated order

Description

The RI command is used to end the input interrupt subroutine.

The input interrupt subroutine begins with the label #ININT. An RI at the end of this routine causes a return to the main program.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	0	1	0	1	Determines state of interrupted trippoint when returning from an automatic subroutine.	n = 0 clears the trippoint. n = 1 restores the interrupted trippoint.

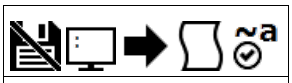
Remarks

- 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.
  - http://www.galilmc.com/support/appnotes/optima/note2418.pdf
- If the program sequencer was interrupted while waiting for a trippoint, such as WT, RI 1 restores the trippoint on the return to the program. RI 0 clears the trippoint.
- A motion trippoint like MF or MR requires the axis to be actively profiling in order to be restored with the RI1 command.
- The RI command re-enables input interrupts.

Examples

```
'Galil DMC Code Example
#a;II 1;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 *Report Latched Position*



RL mm

Usage	RL mm	Argument is an axis mask
Operands	_RLm	Operand has special meaning, see Remarks

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 the appropriate input must be activated. Each axis uses a specific general input for the latch input; see the AL command for information on latch inputs.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
mm	A	ABCDEFGH	ABCDEFGH	Multi-Axis Mask	Axes to query for latched position	

Remarks

- The armed state of the latch can be configured using the CN command.
- The Latch Function works with the main or auxiliary encoder.

Capturing Stepper Position using the Latch

- When working with a stepper motor without an encoder, the latch can be used to capture the stepper position. Follow the steps below to achieve this.

1. Place a wire from the controller Step (PWM) output into the main encoder input, channel A+.
2. Connect the Direction (sign) output into the channel B+ input.
3. Configure the main encoder for Step/Direction using the CE command.
4. The latch will now capture the stepper position based on the pulses generated by the controller.

Operand Usage

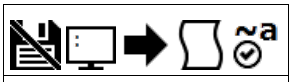
- _RLm contains the latched position of the specified axis.

Examples

```
'Galil DMC Code Example
:JG ,5000;' Set up to jog the B-axis
:BG B;'     Begin jog
:AL B;'     Arm the B latch, assume that after about 2 seconds, input goes low
:RL B;'     Report the latch
10000
```

RL applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

RP *Reference Position*



RP mm

Usage	RP mm	Argument is an axis mask
Operands	_RPm	Operand has special meaning, see Remarks

Description

The RP command returns the commanded reference position of the motor(s). RP command is useful when operating step motors since it provides the commanded position in steps when operating in stepper mode.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
mm	A	ABCDEFGH	ABCDEFGH	Multi-Axis Mask	Axes to report commanded position	
	M	N	N/A	Multi-Axis Mask	Virtual axes to report commanded position	

Remarks

- The relationship between RP, TP and TE: TEA equals the difference between the reference position, RPA, and the actual position, TPA.
  - TE = RP - TP
- _RPm contains the commanded reference position for the specified axis.

Examples

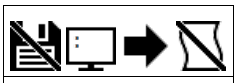
```
'Galil DMC Code Example
'Assume that A axis is commanded to be at the position 200
'The returned units are in quadrature counts.
:PF 7;' Position format of 7
:RP
200
:RP A
200 Return the A motor reference position
:PF -6.0;' Change to hex format
:RP
$0000C8
:position = _RPA;' Assign the variable, position, the value of RPA
```

```
'Galil DMC Code Example
'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
:RP ;'   Return A,B,C,D reference positions
200,-10,0,-110
:RP A
200 Return the A motor reference position
:RP B
-10 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
```

```
'Galil DMC Code Example
:GA N;'     make A axis slave to N imaginary axis
:GR -1;'    1:-1 gearing
:SPN= 10000
:PRN= 10000
:BG N;'     Begin motion
:RP N;'     Get master position
 10000
:RP A;'     Get slave commanded position
-10000
```

RP applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

RS *Reset*



RS n

Usage	RS n ...	Arguments specified with an implicit, comma-separated order
Operands	_RS	Operand has special meaning, see Remarks

Description

The RS command resets the state of the processor to its power-on condition. The previously saved state of the hardware, along with parameter values and saved program, are restored.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	-1	0	0	1	Set behavior of RS command	n = 0 peforms normal reset. n = -1 performs soft master reset. See Remarks.

Remarks

- A soft master reset performed by issuing RS -1 restores factory default settings without erasing the EEPROM. To restore saved EEPROM settings use RS with no arguments, or RS 0.

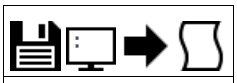
Operand Usage

- _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
  - Bit 2 For program checksum error
  - Bit 1 For parameter checksum error
  - Bit 0 For variable checksum error
- At startup the controller operating system verifies the firmware sector. If there is a checksum error shown by _RS in firmware, it is not loaded and the controller will boot to monitor mode.
  - The #AUTOERR automatic subroutine will run if this error occurs and the subroutine is located in the program space.

Examples

```
'Galil DMC Code Example
:RS;'     Reset the hardware

:RS -1;'   Perform a soft master reset

:
```

SB *Set Bit*



SB n

Usage	SB n ...	Arguments specified with an implicit, comma-separated order

Description

The SB command sets a particular digital output. The SB and CB (Clear Bit) instructions can be used to control the state of output lines.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	1	16	N/A	1	General output bit to be set	Range 1-8 for 1-4 axis controllers
n	17	80	N/A	1	Extended I/O output bit to be set	Requires DB-14064 and I/O must be configured for outputs, see CO command

Remarks

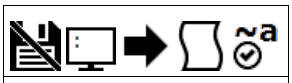
- The state of the output can be read with the @OUT command

Examples

```
'Galil DMC Code Example
#main
SB 5;'      Set digital output 5
SB 1;'      Set digital output 1
CB 5;'      Clear digital output 5
CB 1;'      Clear digital output 1
EN
```

For detailed information on connecting to a Modbus slave, see:
http://www.galilmc.com/techtalk/io-control/setting-up-and-rio-as-extended-io-for-a-controller/

SC *Stop Code*



SC mm

Usage	SC mm	Argument is an axis mask
Operands	_SCm	Operand has special meaning, see Remarks

Description

The Stop Code command returns a number indicating why a motor has stopped.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
mm	A	ABCDEFGH	N/A	Multi-Axis Mask	Axis to query stop code	Omitting argument shows stop code for all axes

Remarks

- When SC is issued, the controller reponds with a number for the axis queried. The number is interpreted as follows:

Stop Code Table

Stop Code Number	Meaning
0	Motors are running, independent mode
1	Motors decelerating or stopped at commanded independent position
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)
6	Stopped by Abort input
7	Stopped by Abort command (AB)
8	Decelerating or stopped by Off on Error (OE1)
9	Stopped after finding edge (FE)
10	Stopped after homing (HM) or Find Index (FI)
11	Stopped by selective abort input
12	Decelerating or stopped by encoder failure (OA1) (For controllers supporting OA/OV/OT)
15	Amplifier Fault (For controllers with internal drives)
16	Stepper position maintenance error
30	Running in PVT mode
31	PVT mode completed normally
32	PVT mode exited because buffer is empty
50	Contour Running
51	Contour Stopped
60	ECAM Running
61	ECAM Stopped
70	Stopped due to EtherCAT communication failure
71	Stopped due to EtherCAT drive fault
99	MC timeout
100	Vector Sequence running
101	Vector Sequence stopped

- _SCm contains the value of the stop code for the specified axis.

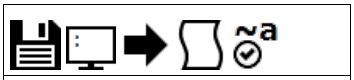
Examples

```
'Galil DMC Code Example
tom = _SCA;'      Assign the Stop Code of A axis to variable tom
```

```
'Galil DMC Code Example
:JG 10000
:BG A
:SC A
 0              //Axis is running in independent mode
:ST A
:SC A
 4              //Axis is stopped by ST command
:
```

SC applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

SD *Switch Deceleration*



SDm= n
SD n,n,n,n,n,n,n,n

Usage	SDm= n	Arguments specified with a single axis mask and an assignment (=)
	SD n ...	Arguments specified with an implicit, comma-separated order
Operands	_SDm	Operand holds the value last set by the command

Description

The Limit Switch Deceleration command (SD) sets the linear deceleration rate of the motors when a limit switch has been reached.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	1,024	1,073,740,800	256,000	1,024	Value of switch deceleration	Resolution changes with TM, see Remarks

Remarks

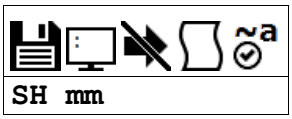
- The resolution of the SD command is dependent upon the update rate setting (TM). With the default rate of TM 1000 the resolution is 1024 cnts/second^2. The equation to calculate the resolution of the AC command is:
  - Resolution = 1024*(1000/TM)^2
  - Example:
    - With TM 500 the minimum AC setting and resolution is 4096 cnts/second^2
    - resolution = 1024*(1000/500)^2 = 4096
- The SD command may be changed during the move in JG move, but not in PR or PA move.

Examples

```
'Galil DMC Code Example
#main
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
EN
```

SD applies to DMC40x0,DMC42x0,DMC41x3,DMC18x6,DMC30010,DMC500x0

SH *Servo Here*



Usage	SH mm	Argument is an axis mask

Description

The SH commands tells the controller to use the current motor position as the command position and to enable servo control here.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
mm	A	ABCDEFGH	ABCDEFGH	Multi-Axis Mask	Axes to enable	

Remarks

- The SH command changes the coordinate system.
  - Therefore, all position commands given prior to SH, must be repeated. Otherwise, the controller produces incorrect motion.
- This command can be useful when the position of a motor has been manually adjusted following a motor off (MO) command.

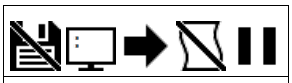
Examples

```
'Galil DMC Code Example
SH ;' Servo A,B,C,D motors
SH A;' Only servo the A motor, the B,C and D motors remain in its previous state.
SH B;' Servo the B motor, leave the A,C and D motors unchanged
SH C;' Servo the C motor, leave the A,B and D motors unchanged
SH D;' Servo the D motor, leave the A,B and C motors unchanged
```

```
'Galil DMC Code Example
'show how issuing SH clears position error
'by resetting the coordinate system
:MO A;'  disable the A axis
:TE A;'  check error on A axis
-12435   large error due to manual motion
:TP A;'  Check position
 12435
:SH A;'  enable A axis, doing so clears the error
:TE A;'  check error again
 0
:TP A;'  confirm position hasn't changed
 12435
```

SH applies to DMC500x0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,DMC18x6,DMC18x2

SL *Single Step*



SL n

Usage	SL n ...	Arguments specified with an implicit, comma-separated order

Description

The SL command is used to single-step through a program for debugging purposes. SL can be used after execution has paused at a breakpoint (BK). The argument n allows user to specify the number of lines to execute before pausing again.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	1	255	1	1	Number of lines to execute before pausing	If n is omitted, default value used.

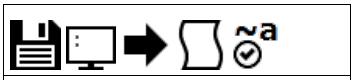
Remarks

- The BK command resumes normal program execution.

Examples

```
'Galil DMC Code Example
:BK 3;' Pause at line 3 (the 4th line) in thread 0
:BK 5;' Continue to line 5
:SL;' Execute the next line
:SL 3;' Execute the next 3 lines
:BK;' Resume normal execution
```

SP *Speed*



SPm= n

SP n,n,n,n,n,n,n,n

Usage	SPm= n	Arguments specified with a single axis mask and an assignment (=)
	SP n ...	Arguments specified with an implicit, comma-separated order
Operands	_SPm	Operand holds the value last set by the command

Description

The SP command sets the slew speed of any or all axes for independent moves.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
	M	N	N/A	Axis	Virtual axis to assign value	
n	0	22,000,000	25,000	2	Value of jog speed in cnts/second	For MT settings of 1,-1,1.5 and -1.5 (Servos) - See Remarks for Resolution details
	0	6,000,000	25,000	2	Value of jog speed in cnts/second	For MT settings of 2,-2,2.5 and -2.5 (Steppers) - See Remarks for Resolution details
	0	50,000,000	25,000	2	Vale of jog speed in cnts/second	ICM-42100 with AF>=5 - See Remarks for Resolution details

Remarks

- Negative values will be interpreted as the absolute value

Resolution

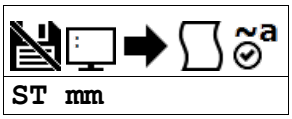
- The resolution of the SP command is dependent upon the update rate setting (TM).
  - With the default rate of TM 1000 the resolution is 2 cnts/second.
  - The equation to calculate the resolution of the SP command is:
    - resolution = 2*(1000/TM)
  - example:
    - With TM 250 the resolution of the SP command is 8 cnts/second
    - resolution = 2*(1000/250) = 8

Examples

```
'Galil DMC Code Example
PR 2000,3000,4000,5000;'          Specify a,b,c,d parameter
SP 5000,6000,7000,8000;'   Specify a,b,c,d speeds
BG ;'                              Begin motion of all axes
AM C;'                            After C motion is complete
'
'
'For vector moves, use the vector speed command (VS) to change the speed.
'SP is not a "mode" of motion like JOG (JG).
'Note:  2 is the minimum non-zero speed.
```

SP applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

ST *Stop*



Usage	ST mm	Argument is an axis mask

Description

The ST command stops motion on the specified axis. Motors will come to a decelerated stop.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
mm	A	ABCDEFGHMNST	ABCDEFGH	Multi-Axis Mask	Axes to command to stop motion	

Remarks

- If ST is sent from the host without an axis specification, program execution will stop in addition to motion.

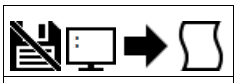
Examples

```
'Galil DMC Code Example
ST A;' Stop A-axis motion
ST S;' Stop coordinate plane S
ST ABCD;' Stop A,B,C,D motion
ST SCD;' Stop coordinate plane S, as well as axes C and D
ST ;' Stop motion on all axes including any virtual axes and coordinate planes
'Use the after motion complete command, AM, to wait for motion to be stopped.
```

```
'Galil DMC Code Example
:ST A;'       Stop motion on the A axis
:SC A;'       Query A axis status
 4            Indicates stopped by ST command
:MG _NO;'     Check if code is running
 1            Thread 0 running
:ST ;'        General stop
:MG _NO;'     check code again
 0            Thread 0 stopped
```

ST applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

TB *Tell Status Byte*



TB

Usage	TB	Command takes no arguments
Operands	_TB	Operand has special meaning, see Remarks

Description

The TB command returns status information from the controller as a decimal number. Each bit of the status byte denotes an active condition when the bit is set (high):

Arguments

The following table describes the specific conditions reported with each bit of the TB report.

Tell Status Byte Response Bit Description

Bit #	Status
Bit 7	Executing application program
Bit 6	Data Record active
Bit 5	Contouring
Bit 4	Executing error or limit switch routine
Bit 3	Input Interrupt enabled
Bit 2	Executing input interrupt routine
Bit 1	DPRAM Polling active
Bit 0	Echo on

Remarks

- _TB Contains the status byte reported by the TB command

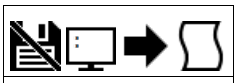
Examples

```
'Galil DMC Code Example
:TB
65'       Data Record Active and Echo is on (2^6 + 2^0 = 64 + 1 = 65)
```

```
'Galil DMC Code Example
:TB
33'       Contouring on and Echo is on (2^5 + 2^0 = 32 + 1 = 33)
```

```
'Galil DMC Code Example
:TB;'       Tell status information
129'      Executing program and echo on (2^7 + 2^0 = 128 + 1 = 129)
```

TC *Tell Error Code*



TC n

Usage	TC n ...	Arguments specified with an implicit, comma-separated order
Operands	_TC	Operand has special meaning, see Remarks

Description

The TC command reports programming or command errors detected by the controller. 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 or when the response to a command is a question mark.

Arguments

Argument	Value	Description	Notes
n	0	Return the numerical code only	Default
	1	Return the numerical code and human-readable message	

TC Error Code List

Tell Code Number	Description	Notes
1	Unrecognized command	
2	Command only valid from program	
3	Command not valid in program	
4	Operand error	
5	Input buffer full	
6	Number out of range	
7	Command not valid while running	
8	Command not valid while not running	
9	Variable error	
10	Empty program line or undefined label	
11	Invalid label or line number	
12	Subroutine more than 16 deep	
13	JG only valid when running in jog mode	
14	EEPROM check sum error	
15	EEPROM write error	
16	IP incorrect sign during position move or IP given during forced deceleration	
17	ED, BN and DL not valid while program running	
18	Command not valid when contouring	
19	Application strand already executing	
20	Begin not valid with motor off	
21	Begin not valid while running	
22	Begin not possible due to Limit Switch	
24	Begin not valid because no sequence defined	
25	Variable not given in IN command	
28	S operand not valid	
29	Not valid during coordinated move	
30	Sequenct Segment Too Short	
31	Total move distance in a sequence > 2 billion	
32	Segment buffer full	
33	VP or CR commands cannot be mixed with LI commands	
39	No time specified	
41	Contouring record range error	
42	Contour data being sent too slowly	
46	Gear axis both master and follower	
50	Not enough fields	
51	Question mark not valid	
52	Missing " or string too long	
53	Error in {}	
54	Question mark part of string	
55	Missing [or []	
56	Array index invalid or out of range	
57	Bad function or array	
58	Bad command response	i.e._GNX
59	Mismatched parentheses	

60	Download error - line too long or too many lines	
61	Duplicate or bad label	
62	Too many labels	
63	IF statement without ENDIF	
65	IN command must have a comma	
66	Array space full	
67	Too many arrays or variables	
71	IN only valid in thread #0	
80	Record mode already running	
81	No array or source specified	
82	Undefined Array	
83	Not a valid number	
84	Too many elements	
90	Only A B C D valid operand	
97	Bad Binary Command Format	
98	Binary Commands not valid in application program	
99	Bad binary command number	
100	Not valid when running ECAM	
101	Improper index into ET	
102	No master axis defined for ECAM	
103	Master axis modulus greater than 256 EP value	
104	Not valid when axis performing ECAM	
105	EB1 command must be given first	
106	Privilege Violation	
110	No hall effect sensors detected	
111	Must be made brushless by BA command	
112	BZ command timeout	
113	No movement in BZ command	
114	BZ command runaway	
118	Controller has GL1600 not GL1800	
119	Not valid for axis configured as stepper	
120	Bad Ethernet transmit	not valid for PCI
121	Bad Ethernet packet received	not valid for PCI
123	TCP lost sync	not valid for PCI
124	Ethernet handle already in use	not valid for PCI
125	No ARP response from IP address	not valid for PCI
126	Closed Ethernet handle	not valid for PCI
127	Illegal Modbus function code	not valid for PCI
128	IP address not valid	not valid for PCI
130	Remote IO command error	not valid for PCI
131	Serial Port Timeout	not valid for PCI, See Remarks
132	Analog inputs not present	
133	Command not valid when locked / Handle must be UDP	not valid for PCI
134	All motors must be in MO for this command	
135	Motor must be in MO	
136	Invalid Password	
137	Invalid lock setting	
138	Passwords not identical	
140	Serial encoder missing	Valid for BiSS support
141	Feature not supported	
143	TM timed out	Valid on SER firmware (SSI and BiSS)
160	BX failure	
161	Sine amp axis not initialized	
163	IA command not valid when DHCP mode enabled	
164	Exceeded maximum sequence length, BGS or BGT is required	
166	Unable to set analog output	30000 Hardware, see AO
180	Duplicate EtherCAT station ID or cable position	DMC500x0 firmware only
181	EtherCAT initialization Timeout	DMC500x0 firmware only
182	EtherCAT drive not present	DMC500x0 firmware only
183	Not valid when EtherCAT network is up	DMC500x0 firmware only
184	EtherCAT lost sync	DMC500x0 firmware only
185	Drive not currently in fault state	DMC500x0 firmware only
186	Not available on EtherCAT	DMC500x0 firmware only
187	Invalid EtherCAT configuration	DMC500x0 firmware only

188	Not valid when EtherCAT network is down	DMC500x0 firmware only
189	Not valid with this motor type	DMC500x0 firmware only

Remarks

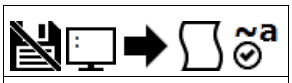
- TC command accepts ? as a query. This is equivalent to TC or TC 0
- After TC has been read, the error code is set to zero.
- _TC contains the value of the error code. Use of the operand does not clear the error code.

Examples

```
'Galil DMC Code Example
:GF32;'   Bad command
?
:TC 1;'    Tell error code
1       Unrecognized command
:
```

TC applies to DMC500x0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO,DMC18x6,DMC18x2

TD *Tell Dual Encoder*



TD mm

Usage	TD mm	Argument is an axis mask
Operands	_TDm	Operand has special meaning, see Remarks

Description

The TD command returns the current position of the dual (auxiliary) encoder input. When operating with stepper motors, the TD command returns the number of counts that have been output by the controller.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
mm	A	ABCDEFGH	ABCDEFGH	Multi-Axis Mask	Axes to report dual (auxiliary) encoder position.	

Remarks

- Auxiliary encoders are not available for a stepper axis or for the axis where output compare is used.

Operand Usage

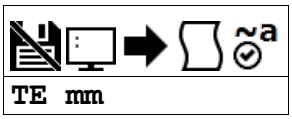
- _TDm reports the dual encoder position for the specified axis.

Examples

```
'Galil DMC Code Example
:TD ;' Return A,B,C,D Dual encoders
 200, -10, 0, -110
:TD A;' Return the A motor Dual encoder
 200
:dual= _TDA;' Assign the variable, DUAL, the value of TDA
```

TD applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

TE *Tell Error*



Usage	TE mm	Argument is an axis mask
Operands	_TEm	Operand has special meaning, see Remarks

Description

The TE command returns the current error in the control loop.

The command returns the position error of the motor(s), which is the difference between commanded (RP) and actual (TP) position.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
mm	A	ABCDEFGH	ABCDEFGH	Multi-Axis Mask	Axes to report position error	

Remarks

- Under normal operating conditions with servo control, the position error should be small. The position error is typically largest during acceleration and deceleration.
- The Tell Error command is not valid for step motors since they operate open-loop.

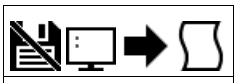
Operand Usage

- _TEm contains the current position error value for the specified axis.

Examples

```
'Galil DMC Code Example
:TE ;' Return all position errors
 5, -2, 0, 6
:TE A;' Return the A motor position error
 5
:TE B;' Return the B motor position error
 -2
:error = _TEA;' Sets the variable, Error, with the A-axis position error
```

TI *Tell Inputs*



```
TI n
```

Usage	TI n ...	Arguments specified with an implicit, comma-separated order
Operands	_TI0 _TI1 _TI2 _TI3 _TI4 _TI5 _TI6 _TI7 _TI8 _TI9 _TI10 _TI11	Operand has special meaning, see Remarks

Description

The TI command returns the state of the inputs in banks of 8 bits, or 1 byte. 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

Argument	Value	Description	Notes
n	0	Report status of Inputs 1-8	Default
	1	Report status of Inputs 9-16	Only valid for 5-8 axis controllers
	2	Report status of Inputs 17-24	Must have extended IO configured as inputs for valid values. See Remarks
	3	Report status of Inputs 25-32	Must have extended IO configured as inputs for valid values. See Remarks
	4	Report status of Inputs 31-40	Must have extended IO configured as inputs for valid values. See Remarks
	5	Report status of Inputs 41-48	Must have extended IO configured as inputs for valid values. See Remarks
	6	Report status of Inputs 49-56	Must have extended IO configured as inputs for valid values. See Remarks
	7	Report status of Inputs 57-64	Must have extended IO configured as inputs for valid values. See Remarks
	8	Report status of Inputs 65-72	Must have extended IO configured as inputs for valid values. See Remarks
	9	Report status of Inputs 73-80	Must have extended IO configured as inputs for valid values. See Remarks
	10	Report status of Inputs 81-88	Auxiliary encoder inputs. See Remarks
	11	Report status of Inputs 89-96	Auxiliary encoder inputs. Only valid for 5-8 axis controllers. See Remarks

Remarks

- For n = 2 to n = 9, the DB-14064 hardware is required for extended IO support.
- Extended IO blocks must be configured as inputs with the CO command before using TI
- For n = 10 and n = 11, the auxiliary encoder channels A and B can be used as additional IO. Only 2 * the number of axes worth of inputs are available.
  - See the User manual for more details.

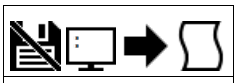
Operand Usage

- _TIn contains the status byte of the input block specified by 'n'.
  - Note that the operand can be masked to return only specified bit information - see section on Bit-wise operations.

Examples

```
'Galil DMC Code Example
:TI 1;'              Tell input state on bank 1
8                   Bit 3 is high, others low
:TI 0
0                   All inputs on bank 0 low
:input= _TI1;'      Sets the variable, Input, with the TI1 value
:input= ?
8.0000
```

TIME *Time Operand*



Usage	variable= TIME	Holds a value
Operands	TIME	Operand has special meaning, see Remarks

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. The clock is reset to 0 with a standard reset or a master reset.

Arguments

TIME is an operand and has no parameters

Remarks

- The keyword, TIME, does not require an underscore (_) as with the other operands.
- TIME will increment up to +2,147,483,647 before rolling over to -2,147,483,648 and continuing to count up.
    - TIME rollover occurs after ~24-25 days of on-time at TM 1000 with no reset.
- TM 1000 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.

Examples

```
'Galil DMC Code Example
MG TIME;'  Display the value of the internal clock
t1= TIME;'  Sets the variable t1 to the TIME value
```

TK *Peak Torque Limit*

TKm= n

TK n,n,n,n,n,n,n,n

Usage	TKm= n	Arguments specified with a single axis mask and an assignment (=)
	TK n ...	Arguments specified with an implicit, comma-separated order
Operands	_TKm	Operand holds the value last set by the command

Description

The TK command sets the peak torque limit on the motor command output. This command works with the TL command which sets the continuous torque limit. When the average torque is below TL, the motor command signal can go up to the TK (Peak Torque) limit for a short amount of time.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	0	9.9982	0	20/65,536	Value of peak torque limit	n = 0 disables the peak torque limit

Remarks

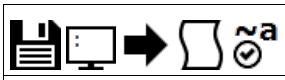
- TK provides the absolute value of the peak torque limit for -/+ torque outputs
- Peak torque can be achieved for approximately 1000 samples upon initial command from 0V torque.
- If TK is set lower than TL, then TL is the maximum command output under all circumstances.

Examples

```
'Galil DMC Code Example
TLA= 7;'   Limit A-axis to a 7 volt average torque output
TKA= 9.99;'  Limit A-axis to a 9.99 volt peak torque output
```

TK applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

TL *Torque Limit*



TLm= n

TL n,n,n,n,n,n,n,n

Usage	TLm= n	Arguments specified with a single axis mask and an assignment (=)
	TL n ...	Arguments specified with an implicit, comma-separated order
Operands	_TLm	Operand holds the value last set by the command

Description

The TL command sets the limit on the motor command output. This limit is designed to prevent over current to motors with lower current rating than the drive.

TL works along with the TK (Peak torque) command to control output current to the motor.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	0	9.9982	9.9982	20/65,536	Value of torque limit	

Remarks

- TL sets the absolute torque maximum for negative and positive torque
  - For example, TL of 5 limits the motor command output to 5 volts maximum and -5 volts minimum

Examples

```
'Galil DMC Code Example
:TL 1,5,9,7.5;' Limit A-axis to 1 volt. Limit B-axis to 5 volts. Limit C-axis to 9 volts. Limit D-axis to 7.5 volts.
:TL ?,?,?,?;' Return limits
1.0000,5.0000,9.0000,7.5000
:TL ?;'  Return A-axis limit
1.0000
```

TL applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

TM *Update Time*

TM n

Usage	TM n ...	Arguments specified with an implicit, comma-separated order
Operands	_TM	Operand holds the value last set by the command

Description

The TM command sets the sampling period of the control loop. The units of this command are microseconds. A negative number turns off the servo loop.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	62.5	5,000	1,000	31.25	Set the sample time in usecs	The minimum value varies based on axis count and firmware usage. See Remarks

Remarks

- TM 1000 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.
- If a higher sampling frequency is required, please contact Galil.
- The minimum allowed TM setting for the controller is listed in the tables below:

- The following commands are automatically scaled to adjust for changes in sample time.
  - AC
  - AS
  - AT
  - DC
  - FA
  - FV
  - HV
  - JG
  - KP
  - NB
  - NF
  - NZ
  - PL
  - SD
  - SP
  - VA
  - VD
  - VS
  - WT
- The following commands are NOT automatically scaled to adjust for changes in sample time
  - BW
  - DR
  - DT
  - IT
  - KD
  - KI
  - TIME
  - TK
  - TV
  - TW
- For more infomation see:
  - [http://www.galilmc.com/techtalk/motion-controllers/time-based-commands-on-accelera-controllers/]

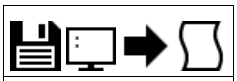
Axis Count	Minimum TM
1-2	62.5
3-4	125
5-6	156.25
7-8	187.5

Examples

```
'Galil DMC Code Example
TM -1000;' Turn off internal clock
TM 2000;' Set sample rate to 2000 msec
TM 1000;' Return to default sample rate
```

TM applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

TN *Vector Tangent*



TN n_0, n_1

Usage	TN n ...	Arguments specified with an implicit, comma-separated order
Operands	_TNm	Operand has special meaning, see Remarks

Description

The TN command describes the tangent axis to the coordinated motion path. n_0 is the scale factor in counts/degree of the tangent axis. n_1 is the absolute position of the tangent axis where the tangent axis is aligned with zero degrees in the coordinated motion plane. The tangent function is useful for cutting applications where a cutting tool must remain tangent to the part.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n_0	-127	127	0	0.004	Scale factor in counts/degree of the tangent axis	
n_1	-8,388,608	8,388,607	0	1	Absolute position of tangent axis where the tangent angle is 0	

Remarks

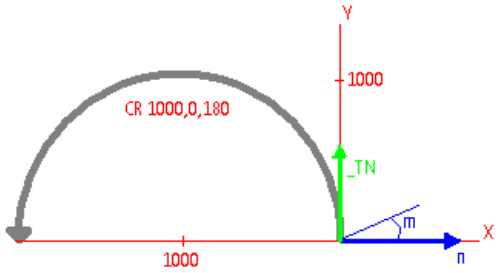
- When operating with stepper motors, n_0 is the scale factor in steps / degree
- The tangent axis is specified with the VMm0m1m2 command where m2 is the tangent axis.
  - For example, VMABD specifies the D axis as the tangent axis

Operand Usage

- _TNm (where m = S or T) contains the first position value for the tangent axis in the specified vector plane. This allows the user to correctly position the tangent axis before the motion begins.
  - _TNm will change based upon the vector path described in the VM declaration. See the example below.
  - n_0 = ? also reports this value

Examples

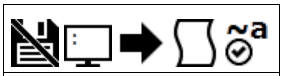
Use a 2D table with a tangent cutting blade to cut a half circle. Ensure that the blade is oriented before turning on the saw. The saw is activated with output 1.




```
'Galil DMC Code Example
#example
VM ABC;'              Z axis is tangent
VS= 500;'              Set vector speed
m= 1000/360;'          Z axis encoder is 1000 counts per full revolution
n= 0;'                 When TPZ=0, blade is oriented to cut along X axis
TN m,n;'              Set these tangent characteristics
CR 1000,0,180;'       Profile a circle with radius 1000 counts,
'                      starting at 0 degrees
'                      and spanning 180 degrees
VE;'                  End the vector path
MG _TNS;'               Print the calculated initial tangent entry point (250)
PA= _TNS;'              Profile a move to orient the Z axis to begin
BG C;'                 Move the blade into place
AM C;'                 Wait until the blade motion is done
SB 1;'                 Turn on the saw
WT 1000;'              Wait for saw to spin up
BG S;'                 Begin vector motion, saw will stay tangent
AM S;'                 Wait for the cut to complete
CB 0;'                 Turn off the saw
MG "ALL DONE";'       Print a message
EN
```

TN applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC500x0

TP *Tell Position*



TP mm

Usage	TP mm	Argument is an axis mask
Operands	_TPm	Operand has special meaning, see Remarks

Description

The TP command returns the current position of the motor.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
mm	A	ABCDEFGH	ABCDEFGH	Multi-Axis Mask	Axes to report motor position	

Remarks

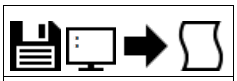
- _TPm contains the current position value for the specified axis.
- Omitting mm returns the position of all axes

Examples

```
'Galil DMC Code Example
'Assume the A-axis is at the position 200 (decimal), the B-axis is at the position -10 (decimal)
'the C-axis is at position 0, and the D-axis is at -110 (decimal).  The returned parameter units are in quadrature counts.
:PF 7;' Position format of 7
:TP ;' Return A,B,C,D positions
 200, -10, 0, -110
:TP A;' Return the A motor position
 200
:TP B;' 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
```

TP applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

TR *Trace*



TR n_0, n_1

Usage	TR n ...	Arguments specified with an implicit, comma-separated order

Description

The TR command causes each instruction in a program to be sent out the communications port prior to execution. The trace command is useful in debugging programs.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n_0	0	1	0	1	Set status of trace function	n_0 = 0 or null disables Trace. n_0 = 1 enables trace.
n_1	0	255	255	1	Set threads to trace by bitmask	See Remarks

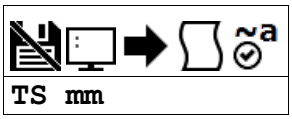
Remarks

- n_1 sets a 1-byte bitmask which determines which threads will run. Bit n set corresponds to thread n traced.
  - For example, setting bit 2 and 3 sets TR to trace threads 2 and 3. (2^2 + 2^3 = 4 + 8 = 12. TR 1,12 is issued)
- Omitting n_1 sets it to the default maximum value to enable trace on all threads.

Examples

```
'Galil DMC Code Example
:'Turn on trace during a program execution
:LS
0 MGTIME
1 WT1000
2 JP0
3
:XQ
:
18003461.0000
18004461.0000
18005461.0000

:TR 1
:
2 JP0
0 MGTIME
18006461.0000
1 WT1000
2 JP0
0 MGTIME
18007461.0000
1 WT1000

:TR 0
:
18008461.0000
18009461.0000

:ST
:
```

TS *Tell Switches*



Usage	TS mm	Argument is an axis mask
Operands	_TSm	Operand has special meaning, see Remarks

Description

The TS command returns information including axis-specific IO status, error conditions, motor condition and state. The value returned by this command is decimal and represents an 8 bit value (decimal value ranges from 0 to 255).

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
mm	A	ABCDEFGH	ABCDEFGH	Multi-Axis Mask	Axes to report axis switches	

Remarks

- Each bit of the TS response represents the following status information when the bit is set (1).

Bit #	Status
Bit 7	Axis in motion
Bit 6	Position error exceeds error limit
Bit 5	Motor off
Bit 4	Reserved (0)
Bit 3	Forward Limit switch inactive
Bit 2	Reverse Limit switch inactive
Bit 1	Home switch status
Bit 0	Position Latch has occurred

- For active high or active low configuration (CN command), the limit switch bits are '1' when the switch is inactive and '0' when active.

Operand Usage

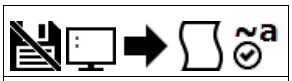
- _TSm contains the current status of the switches for the specified axis.

Examples

```
'Galil DMC Code Example
:v1= _TSB;' Assigns value of TSB to the variable V1
:v1= ?;' Interrogate value of variable V1
15 (returned value) Decimal value corresponding to bit pattern 00001111
Y axis not in motion (bit 7 - has a value of 0)
Y axis error limit not exceeded (bit 6 has a value of 0)
Y axis motor is on (bit 5 has a value of 0)
Y axis forward limit is inactive (bit 3 has a value of 1)
Y axis reverse limit is inactive (bit 2 has a value of 1)
Y axis home switch is high (bit 1 has a value of 1)
Y axis latch is not armed (bit 0 has a value of 1)
```

TS applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

TT *Tell Torque*



TT mm

Usage	TT mm	Argument is an axis mask
Operands	_TTm	Operand has special meaning, see Remarks

Description

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

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
mm	A	ABCDEFGH	ABCDEFGH	Multi-Axis Mask	Axes to report output torque command	

Remarks

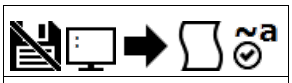
- Torque output is limited by the value set for the TL command.
- _TTm contains the value of the torque for the specified axis.

Examples

```
'Galil DMC Code Example
:v1= _TTA;' Assigns value of TTA to variable, v1
:TT A;'   Report torque on A
-0.2843
```

TT applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

TV *Tell Velocity*



TV mm

Usage	TV mm	Argument is an axis mask
Operands	_TVm	Operand has special meaning, see Remarks

Description

The TV command returns the actual velocity of the axes in units of encoder count/s. The value returned includes the sign bit for direction.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
mm	A	ABCDEFGH	ABCDEFGH	Multi-Axis Mask	Axes to report velocity	

Remarks

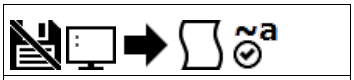
- The TV command is computed using a special averaging filter (over approximately 0.25 sec for TM1000). Therefore, TV will return average velocity, not instantaneous velocity.
- _TVm contains the value of the velocity for the specified axis.

Examples

```
'Galil DMC Code Example
:vela= _TVA;'       Assigns value of A-axis velocity to the variable VELA
:TV A;'             Returns the A-axis velocity
 3420
```

TV applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

TW *Timeout for MC trippoint*



TWm= n
TW n,n,n,n,n,n,n,n

Usage	TWm= n	Arguments specified with a single axis mask and an assignment (=)
	TW n ...	Arguments specified with an implicit, comma-separated order
Operands	_TWm	Operand holds the value last set by the command

Description

The TW command sets the timeout time for the MC trippoint. The TW command sets the timeout in msec to declare an error if the MC command is active and the motor is not at or beyond the actual position within n msec after the completion of the motion profile. If a timeout occurs, then the MC trippoint will clear and the stopcode will be set to 99. A running program will jump to the special label #MCTIME, if located in the application code.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	-1	32,767	32,766	1	Set the timeout in msec for the MC command	n = -1 disables the timeout

Remarks

- The EN command should be used to return from the #MCTIME subroutine.

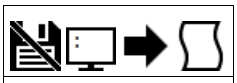
Examples

```
'Galil DMC Code Example
TWA= 1000;' set timeout time for MC to 1000 for A axis
var= _TWA;' set value of TW for A axis to variable, var
```

```
'Galil DMC Code Example
TWA= 5000;'     set MC timeout to 5 seconds
PRA= 10000;'    set move length
BG A
MC A
MG "Move done";' message when move completes
EN
'
#MCTIME
'code when motor doesn't reach final pos in time
MG "Move didn't finish"
MG "Longer than ",_TWA," msecs"
ST A
AM A
MO A;'            shut off axis
EN
```

TW applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

UI *User Interrupt*



UI n

Usage	UI n ...	Arguments specified with an implicit, comma-separated order

Description

The UI command allows user-defined interrupts to be created. UI can generate 16 different status bytes, $F0 to $FF (240-255), corresponding to UI0 to UI15.

UI immediately causes a PC interrupt with the selected status byte, whose meaning is user-defined (unlike EI). When the UI command (e.g. UI5) is executed , a particular status byte value (e.g. $F5 or 245) is delivered to the host PC along with the hardware interrupt.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	0	15	0	1	Set the status byte for the interrupt	

Status Byte (dec)	Condition		Status Byte (dec)	Condition
$F0 (240)	UI0 was executed		$F8 (248)	UI8 was executed
$F1 (241)	UI1 was executed		$F9 (249)	UI9 was executed
$F2 (242)	UI2 was executed		$FA (250)	UI10 was executed
$F3 (243)	UI3 was executed		$FB (251)	UI11 was executed
$F4 (244)	UI4 was executed		$FC (252)	UI12 was executed
$F5 (245)	UI5 was executed		$FD (253)	UI13 was executed
$F6 (246)	UI6 was executed		$FE (254)	UI14 was executed
$F7 (247)	UI7 was executed		$FF (255)	UI15 was executed

Remarks

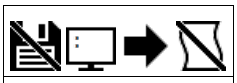
- See Chapter 4 of the User Manual for details.

Examples

```
'Galil DMC Code Example
JG 5000;' Jog at 5000 counts/s
BG A;' Begin motion
AS A;' Wait for at speed
UI 1;' Cause an interrupt with status byte $F1 (241)
'The program above interrupts the host PC with status byte $F1 (241)
'when the motor has reach its target speed of 5000 counts/s
```

UI applies to DMC40x0,DMC42x0,DMC41x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

UL *Upload*



UL

Usage	UL	Command takes no arguments
Operands	_UL	Operand has special meaning, see Remarks

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 or a '\' as an end of text marker.

Arguments

UL is a command with no parameters

Remarks

- In the Galil software, the UL command is not necessary because the UL command is handled by the graphical interface (Upload Program).
- In a terminal utility such as HyperTerminal or Telnet, the UL command will bring the uploaded program to screen.
- From there, the user can copy it and save it to a file.

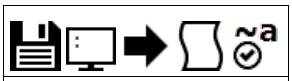
Operand Usage

- When used as an operand, _UL gives the number of available variables.

Examples

```
'Galil DMC Code Example
:UL;' Begin upload
#A Line 0
NO This is an Example Line 1
NO Program Line 2
EN Line 3
{cntrl}Z Terminator
:
```

VA *Vector Acceleration*



VAm= n
VA n,n

Usage	VAm= n	Arguments specified with a single axis mask and an assignment (=)
	VA n ...	Arguments specified with an implicit, comma-separated order
Operands	_VAm	Operand holds the value last set by the command

Description

The VA command sets the acceleration rate of the vector in a coordinated motion sequence.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	S	T	S	Axis	Coordinate plane to be specified	
n	1,024	1,073,740,800	256,000	1,024	Vector acceleration for the coordinate system	

Remarks

- _VAm contains the value of the vector acceleration for the specified coordinate system

Examples

```
'Galil DMC Code Example
:VA 1024;' Set vector acceleration to 1024 counts/sec2
:VA ?;'   Return vector acceleration
1024
:VA 20000;' Set vector acceleration
:VA ?;'          Return vector acceleration
19456
:accel= _VAS;' Assign variable, accel, the value of VA
```

VA applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

VD *Vector Deceleration*

VDm= n
VD n,n

Usage	VDm= n	Arguments specified with a single axis mask and an assignment (=)
	VD n ...	Arguments specified with an implicit, comma-separated order
Operands	_VDm	Operand has special meaning, see Remarks

Description

The VD command sets the deceleration rate of the vector in a coordinated motion sequence.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	S	T	S	Axis	Coordinate plane to be specified	
n	1,024	1,073,740,800	256,000	1,024	Vector deceleration for the coordinate system	

Remarks

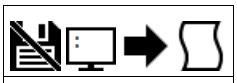
- _VDm contains the value of the vector deceleration for the specified coordinate system.

Examples

```
'Galil DMC Code Example
#vector;'       Vector Program Label
VM AB;'           Specify plane of motion
VA 1000000;'    Vector Acceleration
VD 5000000;'    Vector Deceleration
VS 2000;'       Vector Speed
VP 1000,2000;' Vector Position
VE;'            End Vector
BG S;'          Begin Sequence
AM S;'          Wait for Vector sequence to complete
EN;'            End Program
```

VD applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

VE *Vector Sequence End*



VE n

Usage	VE n ...	Arguments specified with an implicit, comma-separated order
Operands	_VEm	Operand has special meaning, see Remarks

Description

The VE command indicates to the controller that the end of the vector is coming up. This allows the controller to slow down through multiple segments, if required. VE is required to exit the vector mode gracefully (stop code, SC, 101).

Arguments

Argument	Value	Description	Notes
n	0	Specify the end of a vector segment	Also occurs when n = 'null'
	?	Returns the length of the vector in counts	

Remarks

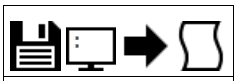
- The VE command will apply to the selected coordinate system, S or T. To select the coordinate system, use the command CAS or CAT.
- _VEm contains the length of the vector in counts for the specified coordinate system, S or T

Examples

```
'Galil DMC Code Example
#vector;'      Vector Program Label
VM AB;'        Specify plane of motion
VA 1000000;'   Vector Acceleration
VD 5000000;'   Vector Deceleration
VS 2000;'      Vector Speed
VP 1000,2000;' Vector Position
VE;'           End Vector
BG S;'         Begin Sequence
AM S;'         Wait for Vector sequence to complete
EN;'           End Program
```

VE applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

VF *Variable Format*



VF $n_0.n_1$

Usage	VF n ...	Arguments specified with an implicit, comma-separated order
Operands	_VF	Operand has special meaning, see Remarks

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

Argument	Min	Max	Default	Resolution	Description	Notes
n_0	-8	10	10	1	Specify the number of digits displayed before the decimal point.	A negative value specifies hexadecimal format. See Remarks
n_1	0	4	4	1	Specify the number of digits displayed after the decimal point.	

Remarks

- A negative n_0 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.
- A positive n_0 specifies standard decimal format.
- A ? is only valid for querying n_0. When queried, the value reported will be the value of the format for variables and arrrays specified by n_0 and n_1
  - eg. VF 10,4 would respond to VF ? with 10.4
- _VF contains the value of the format for variables and arrays
- If the number of digits set by n_0 is insufficient for representing the integer portion of a variable, the returned value will be the greatest number representable by $n_0.n_1$. For example, if *var=123*, and `VF` is 2.4, `var=?` will return `99.9999`.

Examples

```
'Galil DMC Code Example
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
```

```
'Galil DMC Code Example
:VF 8,4;'  set vf to 8 digits of integers and 4 digits of fraction
:VF ?;'    query the value of VF
8.4
:MG _VF;'  query again
8.4
```

VM *Vector Mode*

Usage	VM mm	Argument is an axis mask
Operands	_VMm	Operand has special meaning, see Remarks

Description

The VM command enables the coordinated motion mode and specifies the plane of motion. This mode may be specified for motion on any set of two axes, including a combination of real and virtual axes for single-axis operation. The motion is specified by the instructions VP and CR, which specify linear and circular segments.

Up to 511 segments may be given before the Begin Sequence (BGS or BGT) command. The number of available segments is queriable via the _LMm operand.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m0	A	H	A	Axis	First axis specified for vector motion	
m1	A	H	B	Axis	Second axis specified for vector motion	
	M	N	B	Axis	Virtual axis specified for vector mode	Used when performing vector mode for a single real axis
m2	A	H	N/A	Axis	Tangent axis specified for vector mode.	m2 = null if tangent mode is not desired.
	M	N	N/A	Axis	Virtual axis specified for vector mode.	Used to disable the tangent function if already enabled. Otherwise, use m2 = null.

Remarks

- Specifying one axis for vector mode is useful for obtaining sinusoidal motion on 1 axis using the CR command.
- The Vector End (VE) command must be given after the last segment. This allows the controller to properly decelerate.
- 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 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.

- The VM command will apply to the selected coordinate system, S or T. To select the coordinate system, use the command CAS or CAT.
- _VMm contains instantaneous commanded vector velocity for the specified coordinate system, S or T.

Enabling Vector Mode

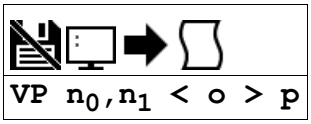
1. Specify the desired coordinate system to use with the CA command. S is default.
2. Specify the vector plane to be used with the VMm0m1 command. If using tangent axis include that as the m2 parameter
    1. EG. for a AB vector plane with the D axis used as a tangent axis, issue VM ABD
    2. If only the vector plane is desired for the above example, then issue VM AB
3. Specify vector speed with VS, vector acceleration with VA, and vector deceleration with VD
4. Specify vector segments with the VP command, or circular segments with the CR command
5. When finished with the sequence of moves, issue VE
6. Issue BGS to begin motion for the S coordinate system
7. You can now wait for motion to complete, issue additional segments as buffer space is cleared, or start a new move on the T coordinate plane by specifying CAT and starting from step 2.

Examples

```
'Galil DMC Code Example
#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
BG S;'           Begin motion sequence
AM S;'           Wait for vector motion to complete
EN;'             End Program
```

```
'Galil DMC Code Example
#a;'             Program Label
VM AN;'          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
BG S;'           Begin motion sequence
AM S;'           Wait for vector motion to complete
EN;'             End Program
```

VP *Vector Position*



Usage	VP n ...	Arguments specified with an implicit, comma-separated order
Operands	_VPm	Operand has special meaning, see Remarks

Description

The VP command defines a vector move segment for the VM mode of motion. The VP command defines the target coordinates of a straight line segment in a 2 axis motion sequence. 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 in linear interpolation mode, use the LI command.

Arguments

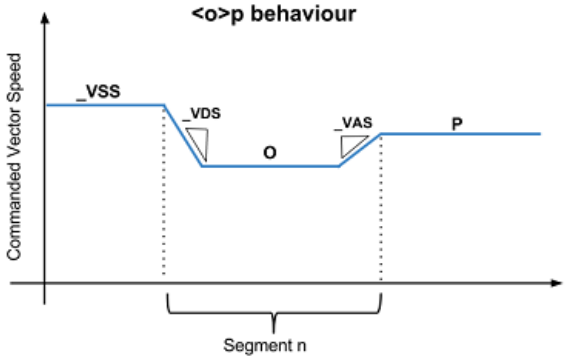
Argument	Min	Max	Default	Resolution	Description	Notes
n0	-2,147,483,648	2,147,483,647	0	1	Specify the target position for the first vector axis	See Remarks
n1	-2,147,483,648	2,147,483,647	0	1	Specify the target postion for the second vector axis	See Remarks
o	2	22,000,000	N/A	2	Specifies the vector speed to be commanded at the beginning of the linear segment. The controller will start accelerating or decelerating at the start of the sequence to this speed.	For MT 1,-1,1.5, and -1.5.
	2	6,000,000	N/A	2	Specifies the vector speed to be commanded at the beginning of the linear segment. The controller will start accelerating or decelerating at the start of the sequence to this speed.	For MT 2,-2,2.5, and -2.5.
p	2	22,000,000	N/A	2	Specifies the vector speed to be achieved at the end of the linear segment. The controller will decelerate or accelerate during the segment and will reach the specified speed at the end of the segment.	For MT 1,-1,1.5, and -1.5.
	2	6,000,000	N/A	2	Specifies the vector speed to be achieved at the end of the linear segment. The controller will decelerate or accelerate during the segment and will reach the specified speed at the end of the segment.	For MT 2,-2,2.5, and -2.5.

Argument	Value	Description	Notes
o	-1	Specifies vector speed to be set by Vector Speed Variable (VV command)	See VV command

Remarks

- 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.
- Vector moves are defined as absolute positions from the origin of the sequence.
- The length of each vector segment must be limited to 8,388,607.

- The VM command will apply to the selected coordinate system, S or T. To select the coordinate system, use the command CAS or CAT.
- _VPm where m = axis designator A,B,C,D,E,F,G or H and 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 of _VPm as an operand is valid in the linear mode, LM, and in the Vector mode, VM.



Examples

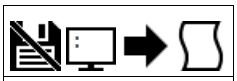
```
'Galil DMC Code Example
#a;'           Program Label
VM AB;'        Specify motion plane
VP 1000,2000;' Specify vector position 1000,2000
```

```
VP 2000,4000;' Specify vector position 2000,4000
CR 1000,0,360;'Specify arc
VE;'           Vector end
BG S;'          Begin motion sequence
AM S;'          Wait for vector motion to complete
EN;'           End Program
```

```
'Galil DMC Code Example
REM VP n,m  p
REM 'o' and 'p' are in counts/sample rather than counts/second as the VS command.
REM This means that when TM <> 1000, commanded speed for VS will be different than
REM values for 'o' and 'p'
REM To get counts/second for 'o' and 'p', divide them by a ratio of 1000/_TM
REM
REM #vs and #vsop result in the same profile
#vs
TM 250
VM AN
VS 100000
VA 2560000
VD 2560000
VP 20000,20000
VE
BG S
AM S
EN
'
#vsop
TM 250
VM AN
n= 1000/_TM
'VS 100000
VA 2560000
VD 2560000
VP 20000,20000<(100000/n)
VE
BG S
AM S
EN
```

VP applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

VR *Vector Speed Ratio*



VR n_0, n_1

Usage	VR n ...	Arguments specified with an implicit, comma-separated order
Operands	_VRm	Operand holds the value last set by the command

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 previous vector speed commands.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n_0	0	10	1	1/65,536	Vector ratio specified for the S coordinate plane	
n_1	0	10	1	1/65,536	Vector ratio specified for the T coordinate plane	

Remarks

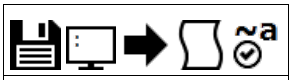
- VR doesn't ratio acceleration or deceleration, but the change in speed is accomplished by accelerating or decelerating at the rate specified by VA and VD.
- VR is useful for feedrate override, particularly when specifying the speed of individual segments using the operator '<' and '>'.
- _VRm contains the vector speed ratio of the specified coordinate system - where m = S or T.
- _VRS contains the vector speed ratio of the specified coordinate system

Examples

```
'Galil DMC Code Example
#a;'             Vector Program
VM AB;'          Vector Mode
VP 1000,2000;'   Vector Position
CR 1000,0,360;'  Specify Arc
VE;'             End Sequence
VS 2000;'        Vector Speed
BG S;'           Begin Sequence
AM S;'           After Motion
JP #a;'           Repeat Move
#speed;'         Speed Override
VR (@AN[1]*.1);' Read analog input compute ratio
vr= _VRS;'        Store vector ratio in variable 'vr'
JP #speed;'      Loop
XQ #a,0
XQ #speed,1;'    Execute task 0 and 1 simultaneously
```

VR applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

VS *Vector Speed*



VSm= n

VS n,n

Usage	VSm= n	Arguments specified with a single axis mask and an assignment (=)
	VS n ...	Arguments specified with an implicit, comma-separated order
Operands	_VSm	Operand has special meaning, see Remarks

Description

The VS command specifies the speed of the vector in a coordinated motion sequence in either the LM or VM modes. This speed is in place when individual segment speeds for VP, LI and CR are not specified.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	S	T	S	Axis	Coordinate plane to be specified	
n	2	22,000,000	25,000	2	Vector speed applied to the coordinate system	

Remarks

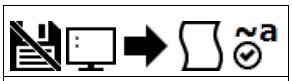
- Vector speed can be attached to individual vector segments using the operators '<' and '>'. For more information, see description of VP, CR, and LI commands. The VV command allows for variables to be specified during vector segments.
- Vector Speed can be calculated by taking the square root of the sum of the squared values of speed for each axis specified for vector or linear interpolated motion.
- _VSm contains the vector speed of the specified coordinate system

Examples

```
'Galil DMC Code Example
:VS 2000;'    Define vector speed of S coordinate system
:VS ?;'       Return vector speed of S coordinate system
2000
:
```

VS applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

VV *Vector Speed Variable*



VVm= n
VV n,n

Usage	VVm= n	Arguments specified with a single axis mask and an assignment (=)
	VV n ...	Arguments specified with an implicit, comma-separated order
Operands	_VVm	Operand has special meaning, see Remarks

Description

The VV command sets the speed of the vector variable in a coordinated motion sequence in either the LM or VM modes. The VV command is used to set the "o" vector speed argument for segments that exist in the vector buffer for LI, CR and VP commands. 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.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	S	T	S	Axis	Coordinate plane to assign value	
n	0	22,000,000	0	2	Variable vector speed	For MT 1,-1,1.5 and -1.5
	0	3,000,000	0	2	Variable vector speed	For MT 2,-2,2.5 and -2.5

Remarks

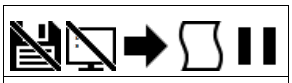
- VV command is useful when vector segments exist in the buffer that use the "<" and ">" speed indicators for specific segment and corner speed control and the host needs to be able to dynamically change the nominal return operating speed.
- _VVm contains the vector speed variable of the specified coordinate system

Examples

```
'Galil DMC Code Example
:VVS= 20000;'          Define vector speed variable to 20000 for the S coordinate system
:VP 1000,2000<-1>100;'  Define vector speed variable for specific segment.
:VVS= ?
20000
:
```

VV applies to DMC40x0,DMC42x0,DMC41x3,DMC18x6,DMC30010,DMC500x0

WT *Wait*



WT n

Usage	WT n ...	Arguments specified with an implicit, comma-separated order

Description

The WT command is a trippoint used to time events.When this command is executed, the controller will wait for the amout of time specified before executing the next command.

The amount of time in the WT command is specified in milliseconds

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	2	2,147,483,646	N/A	2	Number of milliseconds to hold execution of code.	

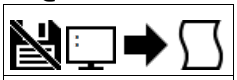
Remarks

- None

Examples

```
'Galil DMC Code Example
'10 seconds after a move is complete, turn on a relay for 2 seconds
#a;'          Program A
PR 50000;'    Position relative move
BG A;'         Begin the move
AM A;'         After the move is over
WT 10000;'    Wait 10 seconds
SB 1;'        Turn on relay (set output 1)
WT 2000;'     Wait 2 seconds
CB 1;'         Turn off relay (clear output 1)
EN;'          End Program
```

XQ *Execute Program*



XQ #str,n_1
XQ n_0,n_1

Usage	XQ n ...	Arguments specified with an implicit, comma-separated order
Operands	_XQ0 _XQ1 _XQ2 _XQ3 _XQ4 _XQ5 _XQ6 _XQ7	Operand has special meaning, see Remarks

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 8 programs may be executed simultaneously to perform multitasking.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
str	1 char	7 chars	See Notes	String	Label to begin code execution	If ommitted, start from line 0(n_0=0)
n_0	0	3,999	0	1	Line number to begin code execution	Firmware Rev 1.1a and later
n_0	0	1,999	0	1	number to begin code execution	
n_1	0	7	0	1	Thread number to execute code	

Remarks

- _XQn contains the current line number of execution for thread n, and -1 if thread t is not running.
- If using ED to add code, you must exit ED mode before executing code.

Examples

```
'Galil DMC Code Example
XQ #apple,0;'  Start execution at label apple, thread zero
XQ #data,2;'   Start execution at label data, thread two
XQ ;'          Start execution at line 0
```

YA *Step Drive Resolution*

YAm= n

YA n,n,n,n,n,n,n,n

Usage	YAm= n	Arguments specified with a single axis mask and an assignment (=)
	YA n ...	Arguments specified with an implicit, comma-separated order
Operands	_YAm	Operand holds the value last set by the command

Description

Specifies the microstepping resolution of the step drive for Stepper Position Maintenance (SPM) mode in microsteps per full motor step. Consult your drive documentation to determine its microstepping setting. See the table below for internal Galil stepper drives.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	0	9,999	2	1	Drive resolution in step counts/motor step for SPM mode	

Remarks

- None

Examples

```
'Galil DMC Code Example
'Set the step drive resolution for a 1/64 Microstepping Drive:
:YA 64,64,64,64
:'Query the D axis value
:MG _YAD;'Response shows D axis step drive resolution
 64.0000
::
```

```
'Galil DMC Code Example
'Set the step drive resolution for a 1/256 Microstepping Drive:
:YA 256
:'Query the A axis value
:MG _YAA;' Response shows A axis step drive resolution
 256.0000
::
```

YA applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

YB *Step Motor Resolution*

YBm= n

YB n,n,n,n,n,n,n,n

Usage	YBm= n	Arguments specified with a single axis mask and an assignment (=)
	YB n ...	Arguments specified with an implicit, comma-separated order
Operands	_YBm	Operand holds the value last set by the command

Description

The YB command specifies the resolution of the step motor, in full steps per full revolution, for Stepper Position Maintenance (SPM) mode.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	0	9,999	200	1	Motor resolution in full steps/revolution	

Remarks

- This command is only required if using SPM mode with stepper motors with an attached encoder.
- A 1.8 degree step motor is 200 steps/revolution.

Examples

```
'Galil DMC Code Example
'Set the step motor resolution of the A axis for a 1.8 degree step motor:
:YBA= 200
:'Query the A axis value
:YBA= ?
 200 Response shows A axis step motor resolution
```

YB applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

YC *Encoder Resolution*

YCm= n

YC n,n,n,n,n,n,n,n

Usage	YCm= n	Arguments specified with a single axis mask and an assignment (=)
	YC n ...	Arguments specified with an implicit, comma-separated order
Operands	_YCm	Operand holds the value last set by the command

Description

The YC command specifies the resolution of the encoder, in counts per revolution, for Stepper Position Maintenance (SPM) mode.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	0	32,766	2,000	1	Encoder resolution in counts/revolution	

Remarks

- This command is only required if using SPM mode with stepper motors with an attached encoder.

Examples

```
'Galil DMC Code Example
'Set the encoder resolution of the A axis
:YCA= 2000
:'Query the A axis value
:YCA= ?
 2000
:'Response shows A axis encoder resolution
```

YC applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

YR *Error Correction*

YRm= n
YR n,n,n,n,n,n,n,n

Usage	YRm= n	Arguments specified with a single axis mask and an assignment (=)
	YR n ...	Arguments specified with an implicit, comma-separated order

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

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	-2,147,483,648	2,147,483,647	0	1	Number of step pulses to increment position by	

Remarks

- Users will typically use the value of QS to increment motor by the number of step pulses of error.
  - EG. YRm = _QSm increments the specified axis by the error magnitude.
- The sign of YR depends on the polarity of the position encoder
  - If the encoder increments when the stepper moves forward (increasing TD), the correction is YRm=_QSm. This is typical.
  - If the encoder decrements when the stepper moves forward, the correction is YRm= -_QSm. See CE to invert the polarity of the position encoder, if desired.

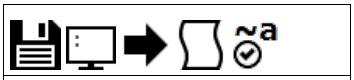
Examples

```
'Galil DMC Code Example
'Query the error of the B axis:
:QS B
 253
:'This shows 253 step counts of error
:'Correct for the error:
:YRB= _QSB;' The motor moves _QS step counts to correct for the error
:'and YS is set back to 1
```

```
'Galil DMC Code Example
'Query the error of the A axis:
:QS A
 253
:' This shows 253 step counts of error
:'Correct for the error:
:YRA= _QSA;' The motor moves _QS step counts to correct for the error
'and YS is set back to 1
```

YR applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

YS *Stepper Position Maintenance Mode Enable, Status*



YSm= n

YS n,n,n,n,n,n,n,n

Usage	YSm= n	Arguments specified with a single axis mask and an assignment (=)
	YS n ...	Arguments specified with an implicit, comma-separated order
Operands	_YSm	Operand has special meaning, see Remarks

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

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	0	1	0	1	Setting of the SPM mode	n = 0 disables SPM mode, n = 1 Enables SPM mode. See Remarks

Remarks

- Both YSm = ? and _YSm contain the value of n. n is 1 when SPM mode is enabled and no error has occurred. If a position error has occured, n becomes 2.
  - If n = 2, this indicates a position error condition defined as more than 3 full motor steps of position error.
  - Issuing an n = 1 will clear the error

Position Error Limit

Microstep Setting (YA)	Error (QS) Limit
1	3
2	6
16	48
64	192
256	768

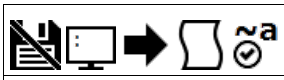
Examples

```
'Galil DMC Code Example
'Enable the mode:
:YSH= 1
:'Query the value:
:YS*= ?
 0,0,0,0,0,0,0,1 Response shows H axis is enabled
```

```
'Galil DMC Code Example
'Enable the mode:
:YSA= 1
:'Query the value:
:YSA= ?
 1 Response shows A axis is enabled
```

YS applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,DMC18x6,DMC18x2,DMC30010,DMC500x0

ZA *User Data Record Variables*



ZAm= n
ZA n,n,n,n,n,n,n,n

Usage	ZAm= n	Arguments specified with a single axis mask and an assignment (=)
	ZA n ...	Arguments specified with an implicit, comma-separated order
Operands	_ZAm	Operand holds the value last set by the command

Description

ZA sets the user variables in the data record. The 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

Argument	Min	Max	Default	Resolution	Description	Notes
m	A	H	N/A	Axis	Axis to assign value	
n	-2,147,483,648	2,147,483,647	0	1	Value of user variable for data record	

Remarks

- n is an integer and can be a number, controller operand, variable, mathematical function, or string.
- Only 4 bytes are available for n. Fractional values are not stored or sent via the data record

Examples

```
'Galil DMC Code Example
#thread
ZAA= myvar;'   constantly update ZA with variable myVar
WT 10
JP #thread;'    run in an infinite loop
```

ZA applies to DMC40x0,DMC42x0,DMC41x3,DMC18x6,DMC30010,DMC500x0

ZS *Zero Subroutine Stack*

ZS n

Usage	ZS n ...	Arguments specified with an implicit, comma-separated order
Operands	_ZS0 _ZS1 _ZS2 _ZS3 _ZS4 _ZS5 _ZS6 _ZS7	Operand has special meaning, see Remarks

Description

The ZS command is used to clear the stack when finishing or leaving a subroutine. This command is used to avoid returning from an interrupt (either input or error). This turns the jump to subroutine into a jump. The status of the stack can be interrogated with the operand _ZS, see Remarks.

Arguments

Argument	Min	Max	Default	Resolution	Description	Notes
n	0	1	0	1	Sets zero stack operation	n = 0 clears the entire stack. n = 1 clears one level of the stack.

Remarks

- Do not use RI (Return from Interrupt) when using ZS.
  - To re-enable interrupts, you must use II command again.

Operand Usage

- _ZSn contains the stack level for the specified thread where n = 0 to 7.
  - The response, an integer between zero and sixteen, indicates zero for beginning condition and sixteen for the deepest value.

Examples

```
'Galil DMC Code Example
#a;'           Main Program
II 1;'          Input Interrupt on 1
#b;JP #b;EN ;' Loop
#ININT;'       Input Interrupt
MG "INTERRUPT";'Print message
s= _ZS0;'        Interrogate stack
s= ?;'          Print stack
ZS;'           Zero stack
s= _ZS0;'        Interrogate stack
s= ?;'          Print stack
EN;'           End
```