



**RIO Firmware Command Reference**  
**Revision: 1752**

**04/05/18**

**©2018 Galil Motion Control**

## Table of Content

Table of Content	2
Legend	6
' Comment	7
- Subtraction Operator	8
# Label Designator	9
#AUTO Subroutine to run automatically upon power up	10
#AUTOERR Bootup Error Automatic Subroutine	11
#CMDERR Command error automatic subroutine	12
#COMINT Communication interrupt automatic subroutine	13
#ININT0 Input interrupt automatic subroutine 0	14
#ININT1 Input interrupt automatic subroutine 1	15
#ININT2 Input interrupt automatic subroutine 2	16
#ININT3 Input interrupt automatic subroutine 3	17
#MODERR Modbus timeout automatic subroutine	18
#SERERR Serial Encoder Error Automatic Subroutine	19
#TCPERR Ethernet communication error automatic subroutine	20
\$ Hexadecimal	21
% Modulo Operator	22
& Bitwise AND Operator	23
( , ) Parentheses (order of operations)	24
* Multiplication Operator	25
/ Division Operator	26
; Semicolon (Command Delimiter)	27
@ABS Absolute value	28
@ACOS Inverse cosine	29
@AN Analog Input Query	30
@AO Analog Output Query	31
@ASIN Inverse sine	32
@ATAN Inverse tangent	33
@COM Bitwise complement	34
@COS Cosine	35
@FLOT Convert Galil 4.2 to Floating Point	36
@FRAC Fractional part	37
@IN Read digital input	38
@INT Integer part	39
@OUT Read digital output	40
@REAL Convert Floating Point to Galil 4.2	41
@RND Round	42
@SIN Sine	43
@SQR Square Root	44
@TAN Tangent	45
[,] Square Brackets (Array Index Operator)	46
^L^K Lock program	47
^R^S Master Reset	48
^R^V Revision Information	49
_ Operand Overview	50
` Line Continuation Character	51
Bitwise OR Operator	52

+ Addition Operator	53
< Less than comparator	54
<= Less than or Equal to comparator	56
<> Not Equal to comparator	57
= Assignment Operator	58
= Equal to comparator	59
> Greater than comparator	60
>= Greater than or Equal to comparator	62
AA After Analog	63
AB Abort	64
AF Analog Feedback Select	65
AI After Input	66
AO Analog Output	67
AQ Analog Input Configuration	68
AT At Time	69
AZ Analog Output Select	70
BK Breakpoint	71
BN Burn	72
BP Burn Program	73
BV Burn Variables and Array	74
CB Clear Bit	75
CF Configure Unsolicited Messages Handle	76
CI Configure Communication Interrupt	77
CL Control Loop	78
CW Copyright information and Data Adjustment bit on/off	79
DA Deallocate Variables and Arrays	80
DB Deadband	81
DF Dual Feedback (DV feedback swap)	82
DH DHCP Client Enable	83
DL Download	84
DM Dimension Array	85
DQ Change Analog Output Range	86
DR Configures I O Data Record Update Rate	87
DY PWM output duty cycle	88
ED Edit	89
ELSE Else function for use with IF conditional statement	91
EN End	92
ENDIF End of IF conditional statement	93
EO Echo	94
FQ PWM output frequency	95
HS Handle Assignment Switch	97
HX Halt Execution	98
IA IP Address	99
ID Identify	100
IF IF conditional statement	102
IH Open IP Handle	103
II Input Interrupt	105
IK Block Ethernet ports	106
IL Integrator Limit	107
IN Input Variable	108
IQ Digital Input Configuration	109

JP	Jump to Program Location	110
JS	Jump to Subroutine	112
KD	Derivative Constant	113
KI	Integrator	114
KP	Proportional Constant	115
LA	List Arrays	116
LL	List Labels	117
LS	List	118
LV	List Variables	119
LZ	Omit leading zeros	120
MA	Email Server IP Address	121
MB	Modbus	122
MD	Email Destination Address	125
ME	Modbus array write enable	126
MG	Message	128
MH	Modbus Heartbeat	130
MI	Modbus Integer	131
MS	Email Source Address	132
MV	Modbus Reversal	133
MW	Modbus Wait	134
NO	No Operation	136
OB	Output Bit	137
OF	Offset	138
OP	Output Port	139
P1CD	Serial port 1 code	140
P1CH	Serial port 1 character	141
P1NM	Serial port 1 number	142
P1ST	Serial port 1 string	143
PC	Pulse Counter Enable	144
PM	PWM output enable	145
PS	Control Setpoint	146
PW	Password	147
QD	Download Array	148
QE	Query Encoder	149
QP	Query Parameters	150
QR	I O Data Record	151
QU	Upload Array	152
QZ	Return Data Record information	153
RA	Record Array	154
RC	Record	155
RD	Record Data	156
RE	Return from Error Routine	157
REM	Remark	158
RI	Return from Interrupt Routine	159
RO	Realtime Offset	160
RS	Reset	161
RT	Real Time	162
RY	Real Year Calendar Function	163
SA	Send Command	165
SB	Set Bit	166
	Description	167

Arguments	167
SE Serial Encoder	167
SI Configure the special Galil SSI feature	169
SL Single Step	170
SM Subnet Mask	171
SS Configure the special Galil BiSS feature	172
SY Serial encoder BiSS active level	174
TB Tell Status Byte	176
TC Tell Error Code	177
TE Tell Error	179
TH Tell Ethernet Handle	180
TI Tell Inputs	181
TIME Time Operand	182
TR Trace	183
TZ Tell I O Configuration	184
UL Upload	185
VF Variable Format	186
WE Write Encoder	187
WH Which Handle	188
WT Wait	189
XQ Execute Program	190
ZC User Variable, ZC	191
ZD User Variable, ZD	192
ZS Zero Subroutine Stack	193

# Legend

---

## **Burnable** **Not Burnable**

### **Description**

Commands with the "burnable" icon can be saved into memory with the BN command. If a reset is issued, the value of the command with this icon will persist if it has been burned into memory.

---

## **Scaled By TM**

### **Description**

Any command with the "scaled by TM" icon will be automatically adjusted whenever a change is made to the TM setting. Commands with this icon are dependent on the sample rate.

---

## **Trippoint**

### **Description**

A command with the "trippoint" icon will halt further program execution until the trippoint's condition is satisfied. Most trippoints cannot be issued as discrete commands, and are only valid in programs.

---

## **Valid In Program** **Not Valid In Program**

### **Description**

Commands with the "valid in program" icon can be used inside of a DMC program that is run locally on the controller. Certain commands may not be used in the program space, and can only be issued as discrete command from an external source such as a terminal.

---

## **Valid In Terminal** **Not Valid In Terminal**

### **Description**

When communicating with a controller externally, only commands which are "valid in terminal" may be sent to the controller as discrete commands. Some commands are only valid when executed in a DMC program and cannot be issued independently.

---

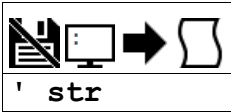
## **Valid In Motion** **Not Valid In Motion**

### **Description**

If a command is "valid in motion" then it may be executed while an axis is in motion. Some commands may not be executed while certain axes are in motion, and can only be executed when the associated axis is stopped.

---

## ' Comment



### Description

The ' allows for a user to insert in a comment on a blank line or 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

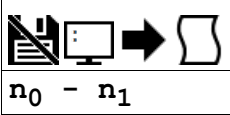
- Comments will be downloaded to controller, thus taking up program space and execution time.
  - See REM for comments that will not download to controller.

### Examples

```
'Galil DMC Code Example
REM This comment is not downloaded to controller and does not take up program
REM  space or execution time
'This comment is downloaded to controller and takes up program space
SH AB;'Comments following a command MUST be preceded by a semi-colon.
KP 10'This is NOT valid use of the '
```

' applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**- Subtraction Operator**

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

**Description**

The subtraction operator takes any two values and returns a value equal to the difference of the arguments.

**Arguments**

Argument	Min	Max	Default	Resolution	Description
<b>n0</b>	-2,147,483,648	2,147,483,647	N/A	1/65,536	Value to subtract from
<b>n1</b>	-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.
- n0 and n1 may also be variables, array elements, operands, or @ functions (e.g. @SIN[]).

**Examples**

```
'Galil DMC Code Example
'Terminal Example
:apple = 10-4
:banana = apple - 3
:MG banana - 1
2.0000
```

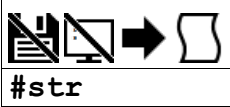
```
'Galil DMC Code Example
REM It is recommended that parenthesis be used when more than one mathematical
REM operation is combined in one command.
cherry = ((10*30)-(60/30));' evaluates as 298
date = 10*30-60/30;' evalutes as 8
EN
```

- applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)



## # Label Designator



### Description

The # denotes the name of a program label, for example, #move. Labels are often used to implement subroutines or loops. Labels are either user-defined or are reserved names, called "automatic subroutines", that automatically execute when a particular event occurs.

### Arguments

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

### Remarks

- Labels can include the characters A-Z, a-z, 1-9. Numbers can not be the first character. All other characters are invalid.
- 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.
- Gall recommends that at least the first character be lowercase for user labels to differentiate from automatic subroutines.
- Automatic subroutines are listed in the command reference starting with a # character.
- There is a maximum of 62 labels available for RIO-47xxx models with standard memory.
- There is a maximum of 126 labels available for RIO-47xxx models with expanded memory.

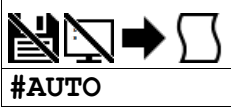
### Examples

```
'Galil DMC Code Example
REM A sample FOR loop
REM Routine will run 10 times and sum all integers 1 through 10
sum=0;' variable to hold sum of integers
i=1;' Create a counter
#for
sum=sum+i;' Add counter to sum
i=i+1;' Increment counter
JP#for,(i<=10)
EN
```

```
'Galil DMC Code Example
REM A sample Do-while loop
REM Routine will run while A axis main encoder position is under 100 counts
#while
WT10;' wait 10 mseconds
JP#while,(_TPA<100);' Loop back if position is under 100 counts
MG"Position is equal or greater than 100"
EN
```

# applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**#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
```

**#AUTO applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

## #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
- For SSI and BiSS operation
  - In the event of a serial position acquisition timeout, the following will occur:
    - The controller will reset
    - The controller servo loop will not run, TM will be set to zero
    - TC1 will return "143 TM timed out"
    - The automatic subroutine #AUTOERR will run, if present
    - The Error output will be set
  - When using serial encoders (SSI or BiSS), the #AUTOERR should follow these guidelines
    - IF\_TC=143 do not employ any trippoints in following code because the timer interrupt is suspended
    - Serial encoders can be disabled with the commands SIn=0 or SSn=0 where n is the axis indicator ABCDEFG or H
    - In order to re-enable the timer interrupt issue "TM n" where n is the servo update period in us (usually n=1000). See TM for more details

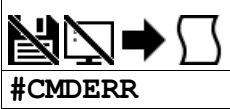
### Examples

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

```
'Galil DMC Code Example
'Use for BiSS and SSI only (-SER firmware)
'Distinguishing between a serial timeout
' condition and an EEPROM condition
#AUTOERR
IF_TC=143
REM BiSS or SSI timeout
REM No trippoints in this clause
REM Print message to DMC-4020 LCD
LU0
MG"BiSS"{L1}
MG"Timeout"{L2}
SSA=0
SSB=0
ELSE
REM Checksum error
REM trippoints ok here
REM Print message to DMC-4020 LCD
LU0
MG"EEProm:"{L1}
MG{z10.0}_RS{L2}
ENDIF
EN
```

#AUTOERR applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**#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**

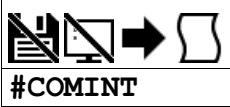
- Use EN to end the routine
- #CMDERR will only run from errors generated within embedded DMC code, not from the terminal or host
- In a single threaded application (Thread 0 only), the EN command in the #CMDERR routine will restart thread 0 where it left off.
- In a multi-threaded application, the thread 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 example shows a bug in DMC code
'#CMDERR detects the bug at runtime and
' provides debugging info
'
#mysub;'user subroutine
xx;'Accidental typo, bad command
EN
'
#CMDERR;'runs if an error occurs
MG'An error occured at line",_ED
TC1;'print type of error info
ZS;'Remove returns from the callback stack
EN;'End execution
```

**#CMDERR applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**#COMINT** *Communication interrupt automatic subroutine***Description**

Automatic subroutine to provide interrupt driven communications from the serial port. #COMINT can be configured by the CI command to run either when any character is received, or when a carriage return is received over the com port. The auxiliary port is used if equipped.

**Arguments**

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

**Remarks**

- Use EN to end the routine
- #COMINT runs in thread 0, and an application must be running in thread 0 in order for #COMINT to be enabled.
- Code running in thread zero will be interrupted by the #COMINT subroutine.
- It is important to handle the interrupt condition and return without delay. The controller will continue to receive data and update the data operands (P1CH,P2CH, etc) while in #COMINT. This can lead to missed characters, numbers, and strings if #COMINT is unnecessarily delayed.
- It is the user's responsibility to ensure the communication buffer is not filled when in this mode, otherwise the controller will report error code 5 "Input Buffer Full". The buffer on the controller is cleared when either of the two following conditions are met:
  - A carriage return is received on the communication port.
  - CI-1 is called.

**Examples**

```
'Galil DMC Code Example
CI2;'          interrupt on any character
#Loop
MG "Loop";'   print a message every second
WT 1000
JP#Loop
#COMINT
MG "COMINT=",P1CH;' print character received
CI -1;'       clear the buffer of the received character
EN1,1
```

**#COMINT applies to DMC500x0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**#ININT0** *Input interrupt automatic subroutine 0***Description**

Automatic subroutine that runs upon a state transition of digital inputs. The RIO provides an input interrupt function which causes the program to automatically execute the instructions following the #ININT0, #ININT1, #ININT2, and/or #ININT3 labels. This function is enabled using the II command. The thread for execution of each label can be chosen along with the input state conditions. Note that the specified thread needs to be executing at the time of the interrupt otherwise the automatic subroutine will be ignored.

**Arguments**

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

**Remarks**

- 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.galil.com/download/application-note/note2418.pdf>

**Examples**

```
'Galil DMC Code Example
#A;           'Program A
II 0,0,3&5&-10; 'Specify interrupt #0 on main thread when inputs 3 and 5 go high, and 10 goes low
#LOOP;JP #LOOP; 'Loop
EN;          'End Program
#ININT0;     'Interrupt subroutine number 2
MG "INTERRUPT"; 'Print Message
#CLEAR
JP#CLEAR,@IN[1]=0; 'Check for 'reset' input 1 to clear interrupt
RI;         'Return to main program
```

**#ININT0 applies to RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**#ININT1** *Input interrupt automatic subroutine 1***Description**

Automatic subroutine that runs upon a state transition of digital inputs. The RIO provides an input interrupt function which causes the program to automatically execute the instructions following the #ININT0, #ININT1, #ININT2, and/or #ININT3 labels. This function is enabled using the I command. The thread for execution of each label can be chosen along with the input state conditions. Note that the specified thread needs to be executing at the time of the interrupt otherwise the automatic subroutine will be ignored.

**Arguments**

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

**Remarks**

- 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 I 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.galil.com/download/application-note/note2418.pdf>

**Examples**

```
'Galil DMC Code Example
#A;           'Program A
II 1,0,3&5&-10; 'Specify interrupt #1 on main thread when inputs 3 and 5 go high, and 10 goes low
#LOOP;JP #LOOP; 'Loop
EN;          'End Program
#ININT1;     'Interrupt subroutine number 2
MG "INTERRUPT"; 'Print Message
#CLEAR
JP#CLEAR,@IN[1]=0; 'Check for 'reset' input 1 to clear interrupt
RI;         'Return to main program
```

**#ININT1 applies to RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**#ININT2** *Input interrupt automatic subroutine 2*

#ININT2

**Description**

Automatic subroutine that runs upon a state transition of digital inputs. The RIO provides an input interrupt function which causes the program to automatically execute the instructions following the #ININT0, #ININT1, #ININT2, and/or #ININT3 labels. This function is enabled using the I command. The thread for execution of each label can be chosen along with the input state conditions. Note that the specified thread needs to be executing at the time of the interrupt otherwise the automatic subroutine will be ignored.

**Arguments**

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

**Remarks**

- 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 I 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.galil.com/download/application-note/note2418.pdf>

**Examples**

```
'Galil DMC Code Example
#A;          'Program A
II 2,0,3&5&-10;  'Specify interrupt #2 on main thread when inputs 3 and 5 go high, and 10 goes low
#LOOP;JP #LOOP;  'Loop
EN;          'End Program
#ININT2;     'Interrupt subroutine number 2
MG "INTERRUPT";  'Print Message
#CLEAR
JP#CLEAR,@IN[1]=0;  'Check for 'reset' input 1 to clear interrupt
RI;         'Return to main program
```

**#ININT2 applies to RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)



**#ININT3** *Input interrupt automatic subroutine 3***Description**

Automatic subroutine that runs upon a state transition of digital inputs. The RIO provides an input interrupt function which causes the program to automatically execute the instructions following the #ININT0, #ININT1, #ININT2, and/or #ININT3 labels. This function is enabled using the I command. The thread for execution of each label can be chosen along with the input state conditions. Note that the specified thread needs to be executing at the time of the interrupt otherwise the automatic subroutine will be ignored.

**Arguments**

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

**Remarks**

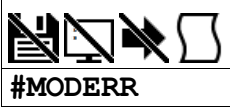
- Use RI to exit the routine
- To make an unconditional jump from #ININT, there are two methods for re-enabling the interrupt capability
  - Issue a ZS and then re-issue the command I before the JP
  - or, use a "null" routine. The "null" routine allows for the execution of the RI command before the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

**Examples**

```
'Galil DMC Code Example
#A;          'Program A
II 3,0,3&5&-10;  'Specify interrupt #3 on main thread when inputs 3 and 5 go high, and 10 goes low
#LOOP;JP #LOOP;  'Loop
EN;          'End Program
#ININT3;     'Interrupt subroutine number 2
MG "INTERRUPT";  'Print Message
#CLEAR
JP#CLEAR,@IN[1]=0;  'Check for 'reset' input 1 to clear interrupt
RI;         'Return to main program
```

**#ININT3 applies to RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**#MODERR** *Modbus timeout automatic subroutine*

#MODERR

**Description**

Automatic subroutine which can be configured with the MH command to run when the RIO has not received a Modbus command within a certain period of time. #MODERR runs in thread 0 and will automatically run even if application code is not running on thread 0 at the time of the timeout.

**Arguments**

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

**Remarks**

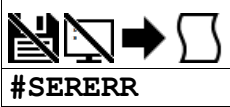
- Use RE to end the routine.

**Examples**

```
'Galil DMC Code Example
#AUTO;      Auto Start
MH 100;     Set timeout at 100msec
#LOOP;     Main loop
           Loop code
JP #LOOP;  Repeat loop
#MODERR;   Modbus Timeout Automatic subroutine occurred
IHA=>-2;   Close timed out handle
RE;       Return from subroutine
```

**#MODERR applies to RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**#SERERR** *Serial Encoder Error Automatic Subroutine***Description**

Automatic subroutine that allows user code to run when there is a serial encoder fault.

This subroutine is only used with controllers equipped with hardware featuring the -BISS or -SSI encoder upgrade.

**Arguments**

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

**Remarks**

- Use the RE command to end this routine.
- #SERERR runs on thread 0
- The following are the fault conditions which will cause #SERERR to interrupt.

*Serial Encoder Faults*

BISS
Encoder timeout (bit 0 of _SE)
CRC error (bit 1 of _SE)
Error bit* (bit 2 of _SE)
Warning bit* (bit 3 of _SE)

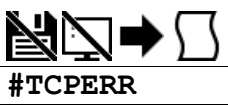
- The active level of the Error and Warning bits for BISS must be configured with SY.
- For the encoder timeout condition, TC1 will also return "140 Serial encoder missing."
- Note: The encoder manufacturer may name the Error and Warning bits differently. Consult the encoder documentation for the naming convention.
  - See the SY command to define the polarity of these bits.
- Gall defines the Warning bit as the bit directly preceding the CRC. The Error bit is defined as the bit directly preceding the Warning bit.

**Examples**

```
'Galil DMC Code Example
#SERERR
MG"SERERR"
MG_SE0      ;' output status bits
SE0,0      ;' disable encoder 0
RE
```

**#SERERR applies to SER**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**#TCPERR** *Ethernet communication error automatic subroutine***Description**

Automatic subroutine which allows execution of user code when an TCP error event occurs. #TCPERR allows the application programmer to run code (for example to reestablish the connection) when error code 123 occurs.

**Arguments**

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

**Remarks**

- Use RE to exit this subroutine.
- Error code 123 (TCP lost sync or timeout) occurs when a message is sent out a handle, and no acknowledgement is received.
  - When this occurs, the handle the message was sent out is closed.
  - #TCPERR can be used to reestablish the handle
- Code does not need to be running in thread 0 for #TCPERR to run.

**Examples**

```
'Galil DMC Code Example
#loop
MG {EA} "L"
WT1000
JP#loop

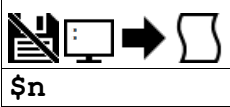
#TCPERR
MG {P1} "TCPERR. Dropped handle", _IA4
RE
```

```
'Galil DMC Code Example
'example of reestablishing connection after TCPERR
'
#main
IHE=192,168,1,30;' connect to 192,168,1,30
WT100;' wait for handle to be established
ipe=_IHE0;' save IP for reconnection use
n=0;' connection counter
#loop;' endless message loop
MG"hello"
WT1000
JP#loop
EN

#TCPERR
IHE=>-3;' make sure handle E is clear
JP#TCPERR,_IHE2<0;' wait for clear handle
IHE=ipe;' set handle with saved IP var
WT100
n=n+1;' increment counter
JP#END,n>5;' try at least 5 times
JP#TCPERR,_IHE2<>-2;'repeat if handle failed
#END
IF(n>5)
MG"failed connection"
HX0;' stop code if connection lost
ELSE
MG"Reconnected"
n = 0;' reset connection counter
ENDIF
RE
```

**#TCPERR applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@gallimc.com](mailto:documentation@gallimc.com)

**\$ Hexadecimal****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

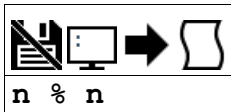
**Examples**

```
'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
```

**\$ applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2,EDD3701x**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

## % Modulo Operator



$$n \% n$$

<b>Usage</b>	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 mod of the arguments.

Mathematical operations are calculated left to right rather than multiplication and division calculations performed prior to addition and subtraction.

Example:

$1+2*3 = 9$ , not 7

It is recommended that parenthesis be used when more than one mathematical 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.
- Mathematical operations are calculated left to right rather than multiplication and division calculations performed prior to addition and subtraction.
  - Example:  $1+2*3 = 9$ , not 7
- It is recommended that parenthesis be used when more than one mathematical operation is combined in one command.
  - Example: var = ((10\*30)+(60/30));' evaluates as 302
  - var = 10\*30+60/30;' evalutes as 12

### Examples

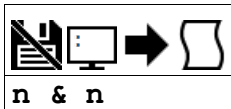
```
'Galil DMC Code Example
'Determine the day of week in n days
DM name[7];'Strings for day of week
name[0]="SUN"
name[1]="MON"
name[2]="TUE"
name[3]="WED"
name[4]="THU"
name[5]="FRI"
name[6]="SAT"
today=2;'Tuesday
days=123;'Days from now
dow=((days + today)%7);'calculate future day of week
MG"The day 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 DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,RIO47xxx,DMC18x6**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

## & Bitwise AND Operator



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

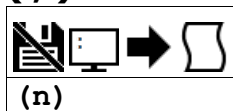
```
'Galil DMC Code Example
'Bitwise use
:var1=$F;'00001111
:var2=$F0;'1111000
:MG (var1 & var2)
0.0000
:MG var1
15.0000
:MG var2
240.0000
```

```
'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
```

**& applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**( , )** *Parentheses (order of operations)***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
```

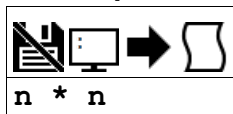
```
'Galil DMC Code Example
:var1=$1F
:var2=$F
:MG var1&var2/$10
0.9375 ($0.F000)
:MG var1&(var2/$10)
0.0000 ($0.0000)
```

**( , )** applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)



**\* Multiplication Operator**



<b>Usage</b>	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.
- Mathematical operations are calculated left to right rather than multiplication and division calculations performed prior to addition and subtraction.
  - Example: 1+2\*3 = 9;' not 7
- It is recommended that parenthesis be used when more than one mathematical operation is combined in one command.
  - Example: var = ((10\*30)+(60/30));' evaluates as 302
  - var = 10\*30+60/30;' evalutes as 12

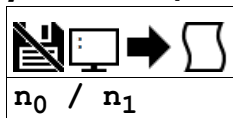
**Examples**

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

\* applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

## / Division Operator



<b>Usage</b>	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
<b>n0</b>	-2,147,483,648	2,147,483,647	N/A	1/65,536	Numerator of divide operation	
<b>n1</b>	-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.
- Mathematical operations are calculated left to right rather than multiplication and division calculations performed prior to addition and subtraction.
  - Example: 1+2\*3 = 9;' not 7
- It is recommended that parenthesis be used when more than one mathematical 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
```

/ applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**;** *Semicolon (Command Delimiter)***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.
- The semicolon operator is used put comments on the same line as the command (STX ;'stop), or to send multiple commands to the controller in the same string.

**Examples**

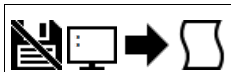
```
'Galil DMC Code Example
SB1;WT500;CB1;' multiple commands separated by semicolons with a comment
```

```
'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
```

; applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,DMC18x6,DMC18x2,RIO47xxx,EDD3701x

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@gallimc.com](mailto:documentation@gallimc.com)

**@ABS** *Absolute value***@ABS** [n]

<b>Usage</b>	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**

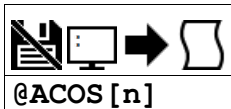
- @ABS[] is an operand, not a command. It can only be used as an argument to other commands and operators

**Examples**

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

**@ABS applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**@ACOS** *Inverse cosine*

<b>Usage</b>	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**

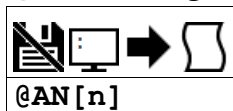
- @ACOS[] is an operand, not a command. It can only be used as an argument to other commands and operators
- @ACOS[] is also referred to as the inverse cosine function

**Examples**

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

**@ACOS applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**@AN** Analog Input Query

<b>Usage</b>	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	0	7	N/A	1	Analog input to query	
n	1,000	3,999	N/A	1	Read Modbus slave analog input	For RIO models with Standard Memory (refer to user manual). See Remarks for usage
n	1,000	5,999	N/A	1	Read Modbus slave analog input	For RIO models with Expanded Memory. See Remarks for usage

**Remarks**

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

**Using @AN with a Modbus Slave**

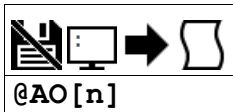
- RIO as Modbus Slave
- 3rd Party Modbus Slave Device
- n is the I/O number calculated using the following equations:
- $n = (\text{HandleNum} * 1000) + (\text{Bitnum} - 1)$ 
  - HandleNum is the handle specifier from A to H.
    - Handle must be assigned to port 502 for Modbus comms (See IH)
  - BitNum is the I/O point in the module from 1 to 8

**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 DMC500x0, DMC52xx0, DMC40x0, DMC42x0, DMC41x3, DMC30010, DMC21x3, RIO47xxx, RIO574x0, DMC18x6**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**@AO** *Analog Output Query*

@AO [n]

<b>Usage</b>	variable = @AO[value]	Performs a function on a value or evaluated statement and returns a value
--------------	-----------------------	---

**Description**

The @AO[n] operator is used to query the value of an Analog Output.

On the RIO-4712x, use the DQ command to specify the voltage output range. Note - The RIO-472xx does not have any analog outputs by default.

**Arguments**

Argument	Min	Max	Default	Resolution	Description	Notes
n	0	7	N/A	1	Analog output to query	See Remarks

**Remarks**

- @AO[] is an operand, not a command. It can only be used as an argument to other commands and operators
- On the RIO-4712x, use the DQ command to specify the voltage output range.
- The RIO-472xx does not have any analog outputs by default.

**Examples**

```
'Galil DMC Code Example
:MG@AO[1];'      Displays status of Analog output 1
2.5000
:Temp=@AO[1];'  Sets variable Temp to the value of Analog output 1
```

**@AO applies to DMC52xx0,DMC30010,RIO47xxx,RIO574x0**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@gallimc.com](mailto:documentation@gallimc.com)

**@ASIN** *Inverse sine***@ASIN**[n]

<b>Usage</b>	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

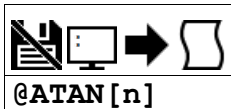
**Examples**

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

**@ASIN applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)



**@ATAN** *Inverse tangent*

<b>Usage</b>	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**

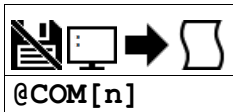
- @ATAN[] is an operand, not a command. It can only be used as an argument to other commands and operators
- @ATAN[] is also referred to as the inverse tangent function

**Examples**

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

**@ATAN applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**@COM** *Bitwise complement*

@COM[n]

<b>Usage</b>	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
```

```
'Galil DMC Code Example
'toggle output 1
OB 1,@COM[@OUT[1]] & 1;' read current state of output 1, take the bitwise complement, mask out bits.
```

**@COM applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@gallimc.com](mailto:documentation@gallimc.com)

**@COS** *Cosine*

@COS [n]

<b>Usage</b>	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
```

**@COS applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**@FLOT** *Convert Galil 4.2 to Floating Point***@FLOT** [n]

<b>Usage</b>	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://babbarge.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
```

**@FLOT applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**@FRAC** *Fractional part***@FRAC** [n]

<b>Usage</b>	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**

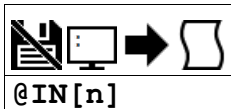
- The sign of the number input to the operation will be maintained in the fractional output.
- @FRAC[] is an operand, not a command. It can only be used as an argument to other commands and operators

**Examples**

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

**@FRAC applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**@IN** Read digital input

<b>Usage</b>	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	0	15	N/A	1	Digital input to query	RIO-471xx and RIO-472xx
	0	23	N/A	1	Digital input to query	RIO-473xx
	24	47	N/A	1	Digital input to query	RIO-473xx with -24ExIn option
n	1,000	3,999	N/A	1	Read Modbus slave bit	For RIO-47xx0. See Remarks
	1,000	5,999	N/A	1	Read Modbus slave bit	For RIO-47xx2 and RIO-473xx. See Remarks

**Remarks**

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

**Using @IN with a Modbus Slave**

- $n = (\text{SlaveAddress} * 10000) + (\text{HandleNum} * 1000) + ((\text{Module}-1) * 4) + (\text{Bitnum}-1)$ 
  - Slave Address is used when the ModBus device has slave devices connected to it and specified as Addresses 0 to 255. Please note that the use of slave devices for modbus are very rare and this number will usually be 0.
  - HandleNum is the handle specifier where A is 1, B is 2 and so on.
  - Module is the position of the module in the rack from 1 to 16.
  - BitNum is the I/O point in the module from 1 to 4

**Examples**

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

@IN applies to DMC500x0, DMC52xx0, DMC40x0, DMC42x0, DMC41x3, DMC30010, DMC21x3, RIO47xxx, RIO574x0, DMC18x6, DMC18x2

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**@INT** *Integer part***@INT** [n]

<b>Usage</b>	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**

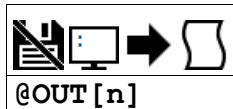
```
'Galil DMC Code Example
:MG @INT[1.2]
1.0000
:MG @INT[-2.4]
-2.0000
```

```
'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
```

**@INT applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@gallimc.com](mailto:documentation@gallimc.com)

**@OUT** *Read digital output*

@OUT [n]

<b>Usage</b>	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
<b>n</b>	0	15	N/A	1	Digital output to query	RIO-471xx and RIO-472xx
	0	23	N/A	1	Digital output to query	RIO-473xx
	24	47	N/A	1	Digital output to query	RIO-473xx with -24ExOut option
<b>n</b>	1,000	3,999	N/A	1	Query Modbus slave bit	For RIO-47xx0. See Remarks
	1,000	5,999	N/A	1	Query Modbus slave bit	For RIO-47xx2 and RIO-473xx. See Remarks

**Remarks**

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

**Using @OUT with a Modbus Slave**

- $n = (\text{SlaveAddress} * 10000) + (\text{HandleNum} * 1000) + ((\text{Module}-1) * 4) + (\text{Bitnum}-1)$ 
  - Slave Address is used when the ModBus device has slave devices connected to it and specified as Addresses 0 to 255. Please note that the use of slave devices for modbus are very rare and this number will usually be 0.
  - HandleNum is the handle specifier where A is 1, B is 2 and so on.
  - Module is the position of the module in the rack from 1 to 16.
  - BitNum is the I/O point in the module from 1 to 4

**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
```

**@OUT applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,RIO574x0,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)



**@REAL** *Convert Floating Point to Galil 4.2***@REAL** [n]

<b>Usage</b>	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://babbarge.cs.qc.cuny.edu/IEEE-754/index.shtml>

**Examples**

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

**@REAL applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**@RND** *Round*

@RND [n]

<b>Usage</b>	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
```

**@RND applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@gallimc.com](mailto:documentation@gallimc.com)

## @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
```

@SIN applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**@SQR** *Square Root*

@SQR [n]

<b>Usage</b>	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
```

**@SQR applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**@TAN** *Tangent*

@TAN [ n ]

<b>Usage</b>	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**

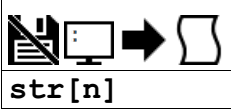
- @TAN[] is an operand, not a command. It can only be used as an argument to other commands and operators

**Examples**

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

**@TAN applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

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

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

They are also used to designate the argument to a function, such as @ABS[n].

**Arguments**

Argument	Min	Max	Default	Resolution	Description	Notes
<b>str</b>	1 char	8 chars	N/A	String	Name of array to access	Must be a valid dimensioned array name.
<b>n</b>	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**

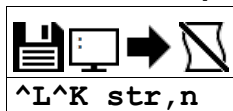
- None

**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
```

**[,]** applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**^L^K** *Lock program*

^L^K str,n

<b>Usage</b>	^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
<b>str</b>	0 char	8 chars	""	String	Controller password string	Password assigned with the PW command.
<b>n</b>	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.

*ASCII Values*

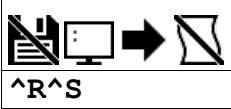
Char	Dec	Hex
^L	12	0C
^K	11	0B

**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 DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,RIO47xx,DMC18x6**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**^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**

- Sending a ^R^S over an Ethernet connection will cause the IP address to be cleared from the controller and will result in a timeout.

*ASCII Values*

Char	Dec	Hex
^R	18	12
^S	19	13

**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
:KP10
:KP?
10.00
:BN
:RS

:KP?
10.00
:^R^S

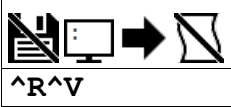
:KP?
6.00
```

**^R^S applies to**

**DMC500x0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2,RIO574x0,DMC52xx0,EDD3701x**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)



**^R^V** *Revision Information*

<b>Usage</b>	^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.

*ASCII Values*

Char	Dec	Hex
^R	18	12
^V	22	16

**Examples**

```
'Galil DMC Code Example
:^R^V
RIO47100 Rev 1.0e1
```

**^R^V applies to**

**DMC500x0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2,RIO574x0,DMC52xx0,EDD3701x**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@gallimc.com](mailto:documentation@gallimc.com)

## Operand Overview



### Operand Usage

Operands allow motion or status parameters of the controller to be incorporated into programmable variables and expressions. Most DMC commands have an equivalent operand - which are designated by adding an underscore (\_) prior to the DMC command. An operand typically contains the value of the command associated with it, for instance `_TPA` contains the current position of axis A. Below is an example of proper and improper usage for an operand.

### Example Usage

'Galil DMC Code Example

'Correct usage

```
MG _TI0; ' Message the Input Bank 0 state
```

```
err = _TC; ' Save the current error code to a variable, err.
```

'Incorrect usage

```
_TI0; ' Sending this to the controller will result in an error, as operands are not valid commands on their own.
```

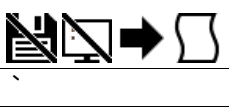
### Special Operands

The majority of DMC operands return information directly related to their command. However, there are a few operands which provide access to internal variables that are not accessible by standard DMC commands. Below is a list of special operands which contain information not stored in a typical DMC command.

**For more details on the content of these operands, see their associated command page.**

Special Operand	Description
<code>_BN</code>	Contains the controller's serial number
<code>_DA</code>	Contains the number of array space left in the controller's memory.
<code>_DL</code>	Contains the number of label space left in the controller's memory.
<code>_DM</code>	Contains the number of array space left in the controller's memory.
<code>_ED0</code>	Contains the line number where an error last occurred.
<code>_ED1</code>	Contains the thread where an error last occurred.
<code>_ED4</code>	Contains the thread ID of the thread evaluating the operand.
<code>_HXn</code>	Contains the running status of thread 'n'.
<code>_NO</code>	Contains a bitmask of the running threads.
<code>_RS</code>	Contains a bitmask of checksum errors.
<code>_UL</code>	Contains the number of variables left in the controller's memory.
<code>_XQn</code>	Contains the current line number of thread 'n'.
<code>TIME</code>	Contains the current value of the controller's free running clock.

## Line Continuation Character



### Description

The ``` (ASCII 96) character allows a command in an application program to extend beyond the maximum line length of 40 characters. This is especially useful for code compression, long MG statements, or multiple conditions in an IF,JP or JS statement. A continued line can be a maximum of 80 characters total.

### Arguments

See Examples for usage.

### Remarks

- When multiple lines are joined using the line continuation character the first line number is the line number used for any errors.
  - For example, if lines 5,6,7 are joined and a syntax error occurs on your 7th line the controller will actually report a problem on line 5.
- The lines following the Line Continuation Character (```) will not be displayed in the trace output (TR1).

### Examples

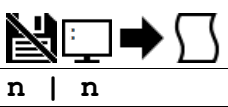
```
'Galil DMC Code Example
#A
a=123`
456;'not displayed with TR1 output
EN
```

```
'Galil DMC Code Example
IF((var100=1000)&(var101=50));MG"GO";EL`
SE;MG"STOP";ENDIF
```

``` applies to DMC30010,RIO47xxx

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

## | Bitwise OR Operator



|              |                              |                                                                  |
|--------------|------------------------------|------------------------------------------------------------------|
| <b>Usage</b> | 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)
```

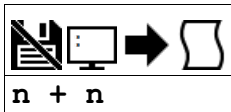
```
'Galil DMC Code Example
'Conditional Use
var1=$F;'00001111
var2=$F0;'1111000
IF (var1 = $F) | (var2 = $F1)
  MG "True"
ELSE
  MG "False"
ENDIF
EN

REM Returned: True
```

| applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

## + Addition Operator



|              |                              |                                                                  |
|--------------|------------------------------|------------------------------------------------------------------|
| <b>Usage</b> | 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

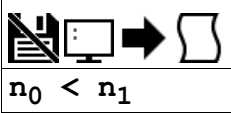
- This is a binary operator (takes two arguments and returns one value). The result of this operation is a value, which is not valid on its own. It must be coupled with a command. See examples below.
- Mathematical operations are calculated left to right rather than multiplication and division calculations performed prior to addition and subtraction.
  - Example: 1+2\*3 = 9;' not 7
- It is recommended that parenthesis be used when more than one mathematical 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
```

**+ applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**< Less than comparator** $n_0 < n_1$ 

|              |                              |                                                                  |
|--------------|------------------------------|------------------------------------------------------------------|
| <b>Usage</b> | 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 |
|----------------------|----------------|---------------|---------|------------|---------------|-------|
| <b>n<sub>0</sub></b> | -2,147,483,648 | 2,147,483,647 | N/A     | 1/65,536   | Value to test |       |
| <b>n<sub>1</sub></b> | -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 comparison is false, it will evaluate to 0.0000.
- Evaluation occurs left to right. Use parenthesis for operator precedence.

**Examples**

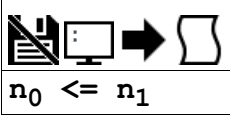
```
'Galil DMC Code Example
:bool= (1<2)
:MG bool
  1.0000
:bool= (1<0)
:MG bool
  0.0000
```

```
'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
```

< applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**<=** *Less than or Equal to comparator*



|              |                               |                                                                  |
|--------------|-------------------------------|------------------------------------------------------------------|
| <b>Usage</b> | 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 |
|-----------|----------------|---------------|---------|------------|---------------|-------|
| <b>n0</b> | -2,147,483,648 | 2,147,483,647 | N/A     | 1/65,536   | Value to test |       |
| <b>n1</b> | -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  $n_0 \leq n_1$ , the expression will evaluate to 1.0000. If the comparison is false, it will evaluate to 0.0000.
- Evaluation occurs left to right. Use parenthesis for operator precedence.

**Examples**

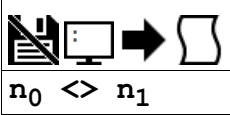
```
'Galil DMC Code Example
:bool= (1 <= 2)
:MG bool
1.0000
:bool= (2 <= 2)
:MG bool
1.0000
:bool= (3 <= 2)
:MG bool
0.0000
```

```
'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
```

**<=** applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: documentation@galilmc.com



**<> Not Equal to comparator**

|              |                               |                                                                  |
|--------------|-------------------------------|------------------------------------------------------------------|
| <b>Usage</b> | 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 |
|-----------|----------------|---------------|---------|------------|---------------|-------|
| <b>n0</b> | -2,147,483,648 | 2,147,483,647 | N/A     | 1/65,536   | Value to test |       |
| <b>n1</b> | -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 comparison is false, it will evaluate to 0.0000.
- Evaluation occurs left to right. Use parenthesis for operator precedence.

**Examples**

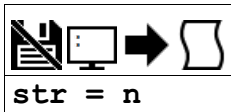
```
'Galil DMC Code Example
:bool= (1 <> 2)
:MG bool
1.0000
:bool= (2 <> 2)
:MG bool
0.0000
```

```
'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, (_TIO <> 170)
EN
```

<> applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

## = Assignment Operator



### Description

The = operator is the assignment operator for the controller. The assignment operator is used for two reasons:

- (1) to define and initialize a variable ( $x = 0$ ) before it is used
- (2) to assign a new value to a variable ( $x = 5$ )

### Arguments

| Argument | Min                | Max           | Default   | Resolution | Description                           | Notes                                                              |
|----------|--------------------|---------------|-----------|------------|---------------------------------------|--------------------------------------------------------------------|
| str      | 1 char             | 8 chars       | N/A       | String     | Variable name to access               |                                                                    |
| n        | -<br>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

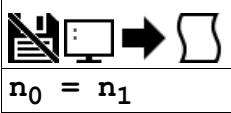
- None

### 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
```

= applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**= Equal to comparator**

|              |                              |                                                                  |
|--------------|------------------------------|------------------------------------------------------------------|
| <b>Usage</b> | 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 |
|-----------|----------------|---------------|---------|------------|---------------|-------|
| <b>n0</b> | -2,147,483,648 | 2,147,483,647 | N/A     | 1/65,536   | Value to test |       |
| <b>n1</b> | -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 comparison is false, it will evaluate to 0.0000.
- Evaluation occurs left to right. Use parenthesis for operator precedence.

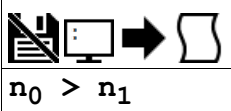
**Examples**

```
'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
```

= applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**> Greater than comparator** $n_0 > n_1$ 

|              |                              |                                                                  |
|--------------|------------------------------|------------------------------------------------------------------|
| <b>Usage</b> | 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 |
|----------------------|----------------|---------------|---------|------------|---------------|-------|
| <b>n<sub>0</sub></b> | -2,147,483,648 | 2,147,483,647 | N/A     | 1/65,536   | Value to test |       |
| <b>n<sub>1</sub></b> | -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 comparison is false, it will evaluate to 0.0000.
- Evaluation occurs left to right. Use parenthesis for operator precedence.

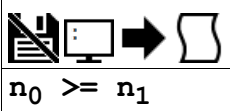
**Examples**

```
'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
```

> applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**>= Greater than or Equal to comparator**

$$n_0 \geq n_1$$

|              |                               |                                                                  |
|--------------|-------------------------------|------------------------------------------------------------------|
| <b>Usage</b> | 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 |
|-----------|----------------|---------------|---------|------------|---------------|-------|
| <b>n0</b> | -2,147,483,648 | 2,147,483,647 | N/A     | 1/65,536   | Value to test |       |
| <b>n1</b> | -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  $n_0 \geq n_1$ , the expression will evaluate to 1.0000. If the comparison is false, it will evaluate to 0.0000.
- Evaluation occurs left to right. Use parenthesis for operator precedence.

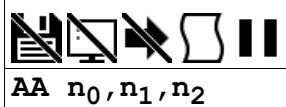
**Examples**

```
'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
```

**>= applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**AA** *After Analog*AA  $n_0, n_1, n_2$ 

|              |          |                                                             |
|--------------|----------|-------------------------------------------------------------|
| <b>Usage</b> | AA n ... | Arguments specified with an implicit, comma-separated order |
|--------------|----------|-------------------------------------------------------------|

**Description**

AA is a trippoint that halts program execution until a voltage on a particular analog input has been reached.

**Arguments**

| Argument | Min | Max | Default | Resolution | Description                   | Notes                                                    |
|----------|-----|-----|---------|------------|-------------------------------|----------------------------------------------------------|
| $n_0$    | 0   | 7   | N/A     | 1          | Selects analog input          |                                                          |
| $n_1$    | 0   | 5   | 6       |            | Selects the trippoint voltage | Range dependent on hardware and AQ setting. See Remarks. |

| Argument | Value | Description                                           | Notes |
|----------|-------|-------------------------------------------------------|-------|
| $n_2$    | 0     | Trippoint to clear when voltage is greater than $n_1$ |       |
|          | 1     | Trippoint to clear when voltage is less than $n_1$    |       |

**Remarks**

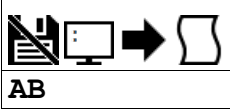
- The range of  $n_1$  is dependant on RIO hardware:
  - 0-5V for RIO-47x0x
  - Range dependent on AQ command for RIO-47x2x
    - See the AQ command for more info

**Examples**

```
'Galil DMC Code Example
#A
'wait for analog input 3 to go above 2.5V
AA 3,2.5,0
MG "Analog input 3 reached 2.5V"
EN
```

**AA applies to RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**AB** *Abort*

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

**Description**

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

The AB command aborts any application code running in any thread on the RIO.

**Arguments**

AB is a command with no arguments.

**Remarks**

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

**Examples**

```
'Galil DMC Code Example
AB;'
Abort application program
```

**AB applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,DMC18x6,DMC18x2,RIO47xxx,EDD3701x**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)



## AF Analog Feedback Select

|                              |
|------------------------------|
|                              |
| AFm= n                       |
| AF n, n, n, n, n, n, n, n, n |

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

### Description

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

The AF command is used to select which analog input channel will be used as the source for feedback in each process control loop.

### Arguments

| Argument | Min | Max | Default | Resolution | Description                                   | Notes                                |
|----------|-----|-----|---------|------------|-----------------------------------------------|--------------------------------------|
| <b>m</b> | A   | F   | N/A     | Channel    | Channel to configure                          | RIO-47xx2 and RIO-473xx              |
|          | A   | B   | N/A     | Channel    | Channel to configure                          | RIO-47xx0                            |
| <b>n</b> | -1  | 7   | -1      | 1          | Selects the analog input channel for feedback | n=-1 to disable process control loop |

### 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      |

- For RIO-47x0x, the analog voltage range is hardset at 0-5V with 12bit resolution.
- For RIO-47x2x, the analog voltage range is set using the AQ command. AQ must be set prior to setting AF.
- The analog feedback is decoded by a 12-bit A/D converter. An upgrade option is available for 16-bits.
- See the User manual for more information on the process control loop mode.

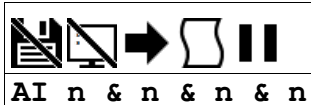
### Examples

```
'Galil DMC Code Example
AF 0;      Use input 0 as feedback
AZ 0;      Use output 0 as control
KP 1;
KD 10;
KI 0.5;
PS 2.5;
```

**AF applies to DMC500x0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2,EDD3701x**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

## AI After Input



AI n &amp; n &amp; n &amp; 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        | 0   | 15  | N/A     | 1          | Input to use for trippoint | +n = High trigger. -n = Low trigger. Up to 4 input conditions may be AND'd together. |
|          | 0   | 23  | N/A     | 1          | Input to use for trippoint | Applies to RIO-473xx.                                                                |

### 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

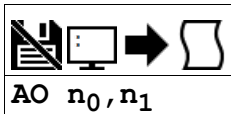
```
'Galil DMC Code Example
#A: ' Begin Program
AI 7&15&-1&-12;' wait until inputs 7 & 15 are high, and inputs 1 & 12 are low
MG "DONE";' Send message 'DONE' when conditions are satisfied
EN;' End Program
```

```
'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
WT10;' 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.
```

**AI applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

## AO Analog Output



AO  $n_0, n_1$

|              |          |                                                             |
|--------------|----------|-------------------------------------------------------------|
| <b>Usage</b> | AO n ... | Arguments specified with an implicit, comma-separated order |
|--------------|----------|-------------------------------------------------------------|

### Description

The AO command sets the analog outputs on the Galil or for a Modbus Slave.

### Arguments

| Argument | Min     | Max    | Default | Resolution | Description                       | Notes                                                                      |
|----------|---------|--------|---------|------------|-----------------------------------|----------------------------------------------------------------------------|
| $n_0$    | 0       | 7      | N/A     | 1          | Set Local Analog Output           | See Remarks                                                                |
| $n_0$    | 1,000   | 3,999  | N/A     | 1          | Set Analog Output on Modbus Slave | For RIO-47xx0. See "Using AO with a Modbus Slave" in Remarks               |
|          | 1,000   | 5,999  | N/A     | 1          | Set Analog Output on Modbus Slave | For RIO-47xx2 and RIO-473xx. See "Using AO with a Modbus Slave" in Remarks |
| $n_1$    | -9.9998 | 9.9998 | N/A     | 20/65,536  | Analog Output Voltage             | See Remarks                                                                |

### Remarks

- RIO-4712x
  - The output voltage range can be adjusted using the DQ command. Use the ID command to check your model number and I/O configuration.
- RIO-472xx
  - Analog outputs are not standard, must be ordered with AO option.

#### Using AO with a Modbus Slave

- RIO as Modbus Slave
- 3rd Party Modbus Slave Device
- $n_0$  is the I/O number calculated using the following equations:
- $n_0 = (\text{HandleNum} * 1000) + ((\text{Module} - 1) * 4) + (\text{Bitnum} - 1)$ 
  - HandleNum is the handle specifier from A to H.
    - Handle must be assigned to port 502 for Modbus comms (See IH)
  - Module is the position of the module in the rack from 1 to 16.
  - BitNum is the I/O point in the module from 1 to 4

### Examples

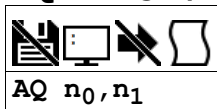
```
'Galil DMC Code Example
:AO 3005,3.2;'           Outputs 3.2 volts on Channel 5 of the Device connected to Handle C
```

```
'Galil DMC Code Example
:AO 2,1.324;'           Outputs 1.324 volts on Channel 2
```

**AO applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,RIO574x0,EDD3701x**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

## AQ Analog Input Configuration



AQ n<sub>0</sub>, n<sub>1</sub>

| Usage           | AQ n ...                                                     | Arguments specified with an implicit, comma-separated order |
|-----------------|--------------------------------------------------------------|-------------------------------------------------------------|
| <b>Operands</b> | _AQ0<br>_AQ1<br>_AQ2<br>_AQ3<br>_AQ4<br>_AQ5<br>_AQ6<br>_AQ7 | Operand has special meaning, see Remarks                    |

### Description

The AQ command is used to set the behavior of the analog inputs. This command will set the analog range and operation for the specified input.

### Arguments

| Argument       | Min | Max | Default | Resolution | Description                          | Notes                                                    |
|----------------|-----|-----|---------|------------|--------------------------------------|----------------------------------------------------------|
| n <sub>0</sub> | 0   | 7   | N/A     | 1          | Analog input channel                 |                                                          |
| n <sub>1</sub> | 0   | 1   | 0       | 1          | Single Ended or Differential Input   | Valid only for non-configurable analog inputs.           |
|                | 1   | 4   | 2       | 1          | Analog range setting                 | Valid only for configurable analog outputs. See remarks. |
|                | -4  | -1  | N/A     | 1          | Specify analog input is differential | valid only for n <sub>0</sub> =0,2,4,6, See Remarks      |

### Remarks

- AQ is a configuration command which must be set at the beginning of application code.
- The usage of this command depends on the type of analog inputs present on the particular controller model, check the ID command to determine the hardware configuration.

#### Configurable Analog Input Settings

| Argument       | Value | Description    | Notes   |
|----------------|-------|----------------|---------|
| n <sub>1</sub> | 1     | -5 to +5 VDC   |         |
|                | 2     | -10 to +10 VDC | Default |
|                | 3     | 0 to 5 VDC     |         |
|                | 4     | 0 to 10 VDC    |         |

- For RIO models with configurable analog inputs, setting a negative n<sub>1</sub> for inputs 0,2,4 or 6 configures those inputs as the differential input relative to input 1,3,5 and 7 respectively.
- Operands \_AQ0 through \_AQ7 return the setting for the specified input
  - Default resolution for analog inputs is 12bits. 16 bit is optional.

#### Differential Input Mapping (-n<sub>1</sub>)

Differential Input Mapping (negative n<sub>1</sub>)

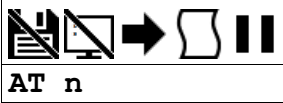
| Input (n <sub>0</sub> ) | Compliment (n <sub>0</sub> + 1) |
|-------------------------|---------------------------------|
| 0                       | 1                               |
| 2                       | 3                               |
| 4                       | 5                               |
| 6                       | 7                               |

### Examples

```
'Galil DMC Code Example
:AQ 2,1;'      Sets Analog input 2 and 3 to be differential inputs with non-configurable analog inputs.
:AQ 2,3;'      Specify analog input 2 as 0-5V with the configurable analog inputs.
:AQ 0,-3;'     Specify analog input 0 as 0-5V and the differential input to analog input 1 with the configurable analog inputs.
:MG_AQ2
3.0000
```

**AQ applies to DMC500x0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC52xx0,RIO574x0,EDD3701x**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**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       | 1          | 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

'I/O scan loop
'If inputs 1,4 and 5 are all high, set output 1 high
'Else, set output 1 low
#main
AT0;' set time reference for AT command
#loop
ti=_TI0&$32;' mask inputs 1,4 and 5 (00110010b = 32h)
IF ti=$32
out=1;' variable for output
ELSE
out=0;' variable for output
ENDIF
OB 1,out;' set output at the end of the scan
AT-100;' set loop scan time to 100 ms
JP#loop
```

**AT applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

## AZ Analog Output Select

|                     |
|---------------------|
|                     |
| AZm= n              |
| AZ n, n, n, n, n, n |

|                 |          |                                                                   |
|-----------------|----------|-------------------------------------------------------------------|
| <b>Usage</b>    | AZm= n   | Arguments specified with a single axis mask and an assignment (=) |
|                 | AZ n ... | Arguments specified with an implicit, comma-separated order       |
| <b>Operands</b> | _AZm     | Operand has special meaning, see Remarks                          |

### Description

The AZ command selects the Analog Outputs used for the process control loops.

### Arguments

| Argument | Min | Max | Default | Resolution | Description                            | Notes                                |
|----------|-----|-----|---------|------------|----------------------------------------|--------------------------------------|
| <b>m</b> | A   | B   | N/A     | Channel    | Channel to assign value                | For RIO-47xx0                        |
|          | A   | F   | N/A     | Channel    | Channel to assign value                | For RIO-47xx2 and RIO-473xx          |
| <b>n</b> | -1  | 7   | -1      | 1          | Analog output for process control loop | n = -1 disables process control loop |

### Remarks

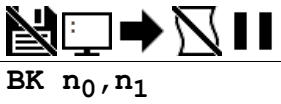
- The RIO-472xx requires additional daughter board hardware for analog outputs
  - See the part number generator for more information, [<http://www.galil.com/order/part-number-generator/rio-47xxx#RIO-47202>]
- \_AZm contains the analog output channel selected for the specified process control loop

### Examples

```
'Galil DMC Code Example
CL 25; '25msec update rate
AF 0; 'analog input 0 as feedback
AZ 0; 'analog output 0 as control
KP 1; 'proportional gain to 1
KD 10; 'derivative gain to 10
KI 0.5; 'integral gain to 0.5
DB 0.1; 'deadband of 0.1V
PS 1.8; 'set-point at 1.8V
```

#### AZ applies to RIO47xxx

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**BK Breakpoint**BK  $n_0, n_1$ 

|                 |          |                                                             |
|-----------------|----------|-------------------------------------------------------------|
| <b>Usage</b>    | BK n ... | Arguments specified with an implicit, comma-separated order |
| <b>Operands</b> | _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   | 199 | N/A     | 1          | Line number to set breakpoint   | For RIO-47xx0. n = null resumes execution               |
|          | 0   | 399 | N/A     | 1          | Line number to set breakpoint   | For RIO-47xx2 and RIO-473xx. 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**

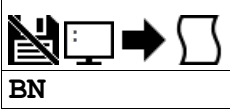
- \_BK will tell whether a breakpoint has been armed, whether it has been encountered, and the program line number of the breakpoint:
  - = -LineNumber: breakpoint armed
  - = LineNumber: breakpoint encountered
  - = -2147483648: breakpoint not armed

**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
```

**BK applies to DMC500x0, DMC52xx0, DMC40x0, DMC42x0, DMC41x3, DMC30010, DMC21x3, RIO47xxx, DMC18x6, DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**BN Burn**

BN

|                 |     |                                          |
|-----------------|-----|------------------------------------------|
| <b>Usage</b>    | BN  | Command takes no arguments               |
| <b>Operands</b> | _BN | Operand has special meaning, see Remarks |

**Description**

The BN command saves certain board parameters in non-volatile EEPROM memory. Once written to the memory, all parameters which can be burned will persist through a software reset (RS command), hardware reset (reset button) or power cycle. This command typically takes 1 second to execute and must not be interrupted. The controller returns a colon (:) when the Burn is complete. All parameters which have been burned into memory can be restored to their factory defaults through a master reset.

This command reference will denote commands 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**

- Issuing this command will pause the output of the Data Record until the command is completed.
- The following table shows the commands that have their parameters saved with the BN command:

*Parameters saved during burn*

|    |    |    |    |
|----|----|----|----|
| AF | IQ | MD | SB |
| AZ | KD | ME | SM |
| CB | KI | MI | RO |
| CW | KP | MS | VF |
| EO | LZ | MV |    |
| IA | MA | PW |    |

- The RIO product line has a maximum of 10,000 write cycles for burning (BV,BP, BN combined).

**Operand Usage**

- \_BN contains the serial number of the processor board.

**Examples**

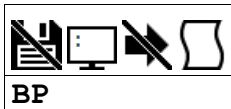
```
'Galil DMC Code Example
SB1;' Set bit 1
CB2;' Clear bit 2
CW1;' Set data adjustment bit
BN;' Burn all parameter states
```

**BN applies to**

**DMC500x0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2,RIO574x0,DMC52xx0,EDD3701x**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)



**BP** *Burn Program*

|              |    |                            |
|--------------|----|----------------------------|
| <b>Usage</b> | 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**

- Issuing this command will pause the output of the Data Record until the command is completed.
- The RIO product line has a maximum of 10,000 write cycles for burning (BV,BP, BN combined).
- 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
```

**BP applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**BV** *Burn Variables and Array*

BV

|                 |     |                                          |
|-----------------|-----|------------------------------------------|
| <b>Usage</b>    | BV  | Command takes no arguments               |
| <b>Operands</b> | _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**

- The RIO product line has a maximum of 10,000 write cycles for burning (BV,BP, BN combined).
- This command may cause the Gall 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 Gall software. This warning does not affect the operation of the board or software.
- Issuing this command will pause the output of the Data Record until the command is completed.

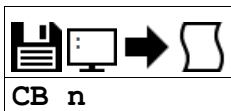
**Examples**

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

**BV applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@gallmc.com](mailto:documentation@gallmc.com)

## CB Clear Bit



| 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. When an output has been cleared, current will cease flowing through the optocoupler for that output.

### Arguments

| Argument | Min   | Max   | Default | Resolution | Description                  | Notes                                                             |
|----------|-------|-------|---------|------------|------------------------------|-------------------------------------------------------------------|
| n        | 0     | 15    | N/A     | 1          | Output bit to be cleared     | RIO-471xx and RIO-472xx                                           |
|          | 0     | 23    | N/A     | 1          | Output bit to be cleared     | RIO-473xx                                                         |
|          | 24    | 47    | N/A     | 1          | Digital output to be cleared | RIO-473xx with -24ExOut option                                    |
| n        | 1,000 | 3,999 | N/A     | 1          | Clear Modbus slave bit       | For RIO-47xx0. See "CB via Modbus Slave" in Remarks               |
|          | 1,000 | 5,999 | N/A     | 1          | Clear Modbus slave bit       | For RIO-47xx2 and RIO-473xx. See "CB via Modbus Slave" in Remarks |

### Remarks

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

#### Using CB with a Modbus Slave

- $n = (\text{SlaveAddress} * 10000) + (\text{HandleNum} * 1000) + ((\text{Module}-1) * 4) + (\text{Bitnum}-1)$ 
  - Slave Address is used when the ModBus device has slave devices connected to it and specified as Addresses 0 to 255. Please note that the use of slave devices for modbus are very rare and this number will usually be 0.
  - HandleNum is the handle specifier where A is 1, B is 2 and so on.
  - Module is the position of the module in the rack from 1 to 16.
  - BitNum is the I/O point in the module from 1 to 4

### 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
```

```
'Galil DMC Code Example
:SB 18;' Set digital output 18
:SB 21;' Set digital output 21
:CB 18;' Clear digital output 18
:CB 21;' Clear digital output 21
```

```
'Galil DMC Code Example
#modbus
REM connect to modbus slave at IP address 192.168.1.50
IHF=192,168,1,50<502>2
WT100
SB 5001;'set bit 1 on modbus slave
WT 10
CB 5003;'clear bit 3 on modbus slave
EN
```

For detailed information on connecting to a Modbus slave, see:

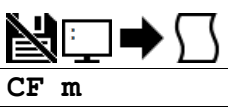
<http://www.galil.com/news/dmc-programming-io-control/setting-rio-pocket-plc-or-generic-modbus-slave-extended-io>

#### CB applies to

**DMC500x0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2,RIO574x0,DMC52xx0,EDD3701x**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

## CF *Configure Unsolicited Messages Handle*



CF m

|                 |       |                                          |
|-----------------|-------|------------------------------------------|
| <b>Usage</b>    | CF mm | Argument is an axis mask                 |
| <b>Operands</b> | _CF   | Operand has special meaning, see Remarks |

### Description

The CF command sets the port for unsolicited messages. The CF command directs the controller to send unsolicited responses to the Main or Aux Serial Port (if equipped), or to an Ethernet handle. An unsolicited message is data generated by the controller which is not in response to a command sent by the host.

### Arguments

| Argument | Min | Max | Default | Resolution | Description                                                        | Notes                       |
|----------|-----|-----|---------|------------|--------------------------------------------------------------------|-----------------------------|
| m        | A   | C   | S       | Handle     | Ethernet Handle to assign as unsolicited message port              | For RIO-47xx0               |
|          | A   | E   | S       | Handle     | Ethernet handle to assign as unsolicited message port              | For RIO-47xx2 and RIO-473xx |
|          | I   | I   | S       | Handle     | Set the port that sent the command as the unsolicited message port | Not valid in program        |
|          | S   | S   | S       | Handle     | Set Main serial port as unsolicited message port                   |                             |

### Remarks

- Examples of application code commands that will generate unsolicited messages follow.

```
'Galil DMC Code Example
MG"Hello";'      A message (MG)
TC1;'           A command that returns a response
TP;'           "
RPA;'           "
var=?;'         A variable interogation
var=;'         "
thisIsAnError;' A dmc error will generate an error message
```

### Ethernet Handle as Unsolicited Message Port

- When communicating over Ethernet, two Ethernet handles should be used:
  - 1.) The first handle should be used for command-and-response traffic. This is the primary handle that the host uses to communicate to the controller.
  - 2.) The second handle should be used for unsolicited traffic. This is the primary handle that the controller uses to asynchronously communicate to the host. Use CF to point unsolicited traffic to this handle.
- It is NOT recommended to use one Ethernet handle for both command-and-response, and unsolicited messages.

### Operand Usage

- \_CF contains the decimal value of the ASCII letter where unsolicited messages are currently routed.

### Examples

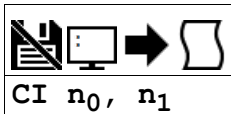
```
'Galil DMC Code Example
:CFI;' send unsolicited traffic to the terminal that sent the command
```

```
'Galil DMC Code Example
'Demonstrates from GalilTools terminal that the
'main handle is separate from the unsolicited handle
'Note the connection indicators IHA and IHB in the following:
'192.168.1.3, RIO47102 Rev 1.0c, 1480, IHA IHB
:TH
CONTROLLER IP ADDRESS 192,168,1,3 ETHERNET ADDRESS 00-50-4C-28-05-C8
IHA TCP PORT 23 TO IP ADDRESS 192,168,1,100 PORT 2420
IHB UDP PORT 60007 TO IP ADDRESS 192,168,1,100 PORT 2421
IHC AVAILABLE
IHD AVAILABLE
IHE AVAILABLE
:WH
IHA
:'Main handle is A
:MG_CF
66.0000
:'Unsolicited handle. 66 is ASCII for "B"
```

**CF applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@gallimc.com](mailto:documentation@gallimc.com)

## CI *Configure Communication Interrupt*



CI  $n_0, n_1$

| Usage | CI $n \dots$ | Arguments specified with an implicit, comma-separated order |
|-------|--------------|-------------------------------------------------------------|
|-------|--------------|-------------------------------------------------------------|

### Description

The CI command configures program interrupts based on input of characters over the communication port.

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

### Arguments

| Argument | Value | Description                                                               | Notes                                               |
|----------|-------|---------------------------------------------------------------------------|-----------------------------------------------------|
| $n_0$    | -1    | Clear interrupt data buffer                                               |                                                     |
|          | 0     | Do not interrupt                                                          | Default                                             |
|          | 1     | Interrupt on carriage return                                              |                                                     |
|          | 2     | Interrupt on any character                                                |                                                     |
| $n_1$    | 0     | Main serial port configured as standard port for input of Galil commands. | Default. CI execution disabled.                     |
|          | 1     | Main serial port configured for CI execution                              | Data received will not be interpreted as a command. |

### Remarks

- For more, see Operator Data Entry Mode in the user manual.
- It is the user's responsibility to ensure the communication buffer is not filled when in this mode, otherwise the controller will report error code 5 "Input Buffer Full". The buffer on the controller is cleared when either of the two following conditions are met:
  - A carriage return is received on the communication port.
  - CI-1 is called.


### Examples

```
'Galil DMC Code Example
:CI 2, 1;'      Interrupt on a single character received from serial port
:CI ,0;'       Reenable port for standard Galil commands
```

**CI applies to DMC500x0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

## CL Control Loop



|                     |
|---------------------|
| CLm= n              |
| CL n, n, n, n, n, n |

|                 |          |                                                                   |
|-----------------|----------|-------------------------------------------------------------------|
| <b>Usage</b>    | CLm= n   | Arguments specified with a single axis mask and an assignment (=) |
|                 | CL n ... | Arguments specified with an implicit, comma-separated order       |
| <b>Operands</b> | _CLm     | Operand holds the value last set by the command                   |

### Description

The CL command sets the sampling period for each of the optional process control loops. The process control loops may all have independent sampling periods.

### Arguments

| Argument | Min | Max    | Default | Resolution | Description                                           | Notes                   |
|----------|-----|--------|---------|------------|-------------------------------------------------------|-------------------------|
| <b>m</b> | A   | B      | N/A     | Channel    | Channel to assign value                               | RIO-47xx0               |
|          | A   | F      | N/A     | Channel    | Channel to assign value                               | RIO-47xx2 and RIO-47300 |
| <b>n</b> | 1   | 65,535 | 1       | 1          | Update rate for process control loops in milliseconds |                         |

### Remarks

- For more information, see the RIO user manual.

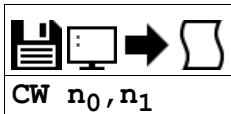
### Examples

```
'Galil DMC Code Example
CL 25; '25msec update rate
AF 0; 'analog input 0 as feedback
AZ 0; 'analog output 0 as control
KP 1; 'proportional gain to 1
KD 10; 'derivative gain to 10
KI 0.5; 'integral gain to 0.5
DB 0.1; 'deadband of 0.1V
PS 1.8; 'set-point at 1.8V
EN
```

```
'Galil DMC Code Example
MG _CLA; 'message loop A default update rate 1ms
MG _CLB; 'message loop B default update rate 1ms
CL 25; 'Set loop A update rate to 25ms
CL ,10; 'Set loop B update rate to 10ms
MG _CLA; 'message new loop A update rate 25ms
MG _CLB; 'message new loop B update rate 10ms
CLA= 10; 'Set loop A update rate to 10ms
CLB= 25; 'Set loop B update rate to 25ms
MG _CLA; 'message new loop A update rate 10ms
MG _CLB; 'message new loop B update rate 25ms
EN
```

### CL applies to RIO47xxx

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**CW** Copyright information and Data Adjustment bit on/offCW n<sub>0</sub>, n<sub>1</sub>

|              |          |                                                             |
|--------------|----------|-------------------------------------------------------------|
| <b>Usage</b> | 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                                                                          |
|----------------------|-------|-------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------|
| <b>n<sub>0</sub></b> | 0     | Causes controller to return a copyright information string                                            | Equivalent to n <sub>0</sub> = ?                                               |
|                      | 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 |
| <b>n<sub>1</sub></b> | 0     | Controller will pause program execution when hardware handshaking disables character transmissions    |                                                                                |
|                      | 1     | Controller will continue program execution when hardware handshaking disables character transmissions | Default. Output characters will be lost if serial buffer is full.              |

**Remarks**

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

**Operand Usage**

- \_CW contains the value set for n<sub>0</sub>
- \_CW4 contains the value set for n<sub>1</sub>

**Examples**

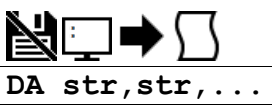
```
'Galil DMC Code Example
CW1;'      Set CW to Galil Driver mode (MSB set on unsolicited characters)

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

**CW applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,DMC18x6,DMC18x2,RIO47xxx,EDD3701x**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@gallimc.com](mailto:documentation@gallimc.com)

## DA Deallocate Variables and Arrays



|                 |          |                                                             |
|-----------------|----------|-------------------------------------------------------------|
| <b>Usage</b>    | DA n ... | Arguments specified with an implicit, comma-separated order |
| <b>Operands</b> | _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 deallocation. Different arrays and variables are separated by comma when specified in one command.

### Arguments

| Argument   | Min    | Max     | Default | Resolution | Description                 | Notes                                |
|------------|--------|---------|---------|------------|-----------------------------|--------------------------------------|
| <b>str</b> | 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

str[] - Defined array name

str - Defined variable name

str = \* deallocates all the variables

str = \*[] - Deallocates all the arrays

### Remarks

- \_DA contains the total number of arrays available.
- DA ? returns 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
```

**DA applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@gallimc.com](mailto:documentation@gallimc.com)



## DB Deadband

|                     |
|---------------------|
|                     |
| DBm= n              |
| DB n, n, n, n, n, n |

|                 |          |                                                                   |
|-----------------|----------|-------------------------------------------------------------------|
| <b>Usage</b>    | DBm= n   | Arguments specified with a single axis mask and an assignment (=) |
|                 | DB n ... | Arguments specified with an implicit, comma-separated order       |
| <b>Operands</b> | _DBm     | Operand holds the value last set by the command                   |

### Description

The DB command selects the deadband where the error must exceed the deadband to execute the process control loop.

### Arguments

| Argument | Min | Max | Default | Resolution | Description             | Notes                                                          |
|----------|-----|-----|---------|------------|-------------------------|----------------------------------------------------------------|
| <b>m</b> | A   | B   | N/A     | Channel    | Channel to assign value | For RIO-47xx0                                                  |
|          | A   | F   | N/A     | Channel    | Channel to assign value | For RIO-47xx2 and RIO-473xx                                    |
| <b>n</b> | -1  | 1   | 0       | 1/65,536   | Deadband in volts       | -n = flow control mode. +n = normal control mode. See Remarks. |

### Remarks

- AF and AZ commands must be set prior to the DB command.
- A negative value for n will place the RIO in a velocity or flow control mode. In this mode the output from the PID filter is held when the feedback is within the deadband range. This includes the contribution of OF.
- With a positive value for n, the output of the PID filter is cleared (set to 0) when the deadband is reached.


### Examples

| Galil DMC Code | Example                            |
|----------------|------------------------------------|
| AF0,1;AZ0,1;   | Set analog input/output on A and B |
| DB,5,.6;       | Set deadband on A and B            |
| MG_DBA;        | Query DB setting on A axis         |

#### DB applies to RIO47xxx

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**DF Dual Feedback (DV feedback swap)**

|                                                                                  |
|----------------------------------------------------------------------------------|
|  |
| DFm= n                                                                           |
| DF n, n, n, n, n, n, n, n, n                                                     |

|                 |          |                                                                   |
|-----------------|----------|-------------------------------------------------------------------|
| <b>Usage</b>    | DFm= n   | Arguments specified with a single axis mask and an assignment (=) |
|                 | DF n ... | Arguments specified with an implicit, comma-separated order       |
| <b>Operands</b> | _DFm     | Operand holds the value last set by the command                   |

**Description**

The DF command allows configuration of BSS or SSI feedback in Dual Loop mode as the load encoder. For users wishing to operate with SSI or BiSS in Dual Loop mode (DV), the DF command can be used to configure a load-side serial encoder and a motor-side incremental encoder with DV1.

**Arguments**

| Argument | Value | Description           | Notes   |
|----------|-------|-----------------------|---------|
| n        | 0     | Disable feedback swap | Default |
|          | 1     | Enable feedback swap  |         |

**Remarks**

- Wire the motor's incremental encoder per normal to the DMC-4xxx main encoder inputs. The load SSI encoder should be wired to the axis aux encoder lines:

*SSI Signals for DMC-4xxx*

| Nominal Signal Name | Signal Reassignment with SSI | Signal Reassignment with BiSS |
|---------------------|------------------------------|-------------------------------|
| AA+                 | Clock+                       | MA+                           |
| AA-                 | Clock-                       | MA-                           |
| AB+                 | Data+                        | SLO+                          |
| AB-                 | Data-                        | SLO-                          |

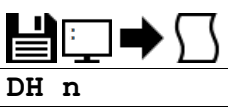
- Once wired, configure the serial encoder as an auxiliary encoder.
  - See SI or SS for configuration information.
- Once configured for Dual Loop (DV1), DF1 will swap the serial encoder to become the load (Main) encoder. The incremental encoder wired to the main encoder inputs becomes the motor (Auxiliary) encoder. TP will now report the serial encoder position, and TD will report the incremental encoder position.

**Examples**

```
'Galil DMC Code Example
MOA;'      Disable motor on X
SIA= 2,25,15,0<13>2;' Setup SSI encoder to fill the Aux encoder register
DF1;'      Enable Dual Feedback Swap
DV1;'      Enable Dual Loop mode
SHA;'      Enable servo with new configuration
```

**DF applies to DMC500x0,SER**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**DH** *DHCP Client Enable*

DH n

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

**Description**

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

**Arguments**

| Argument | Value | Description                    | Notes                                              |
|----------|-------|--------------------------------|----------------------------------------------------|
| n        | 0     | Enable BOOT-P and disable DHCP | Allows IP assignment through IA command.           |
|          | 1     | Disable BOOT-P and enable DHCP | Default. Allows IP assignment through DHCP server. |

**Remarks**

- DH 0 must be set to manually assign and burn in an IP address. With DH 1 set, the IA command will return an error if used to set the IP address.


**Examples**

```
'Galil DMC Code Example
DH 1;' Sets the DHCP function on. IA assignment will no longer work.
DH 0;' Sets the DHCP function off, and the Boot-P function on.
```

**DH applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

## DL Download



|         |
|---------|
| DL n    |
| DL #str |
| DL s    |

|              |          |                                                             |
|--------------|----------|-------------------------------------------------------------|
| <b>Usage</b> | 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      | 199     | 0       | 1          | Line number to begin program download        | For RIO-47xx0                                                         |
|          | 0      | 399     | 0       | 1          | Line number to begin program download        | For RIO-47xx2 and RIO-473xx                                           |
| str      | 1 char | 8 chars | ""      | String     | Name of label in RAM to begin download from. | If str = "", download begins at the end of the current program in RAM |
| s        | #      | #       | N/A     | Symbol     | Begins download at end of program in RAM     |                                                                       |

### Remarks

- Do not insert spaces before label declarations.
- \_DL gives the number of available labels.
- Issuing this command will pause the output of the Data Record until the command is completed.
- This command will be rejected by Galil software if sent via the terminal. In order to download a program using a Galil software package, use that package's prescribed programming interface (I.E. GDK's Editor Tool).
- The total number of labels is 62 on the RIO-47xx0.
- On the RIO-47xx2 this is increased to 126 labels.

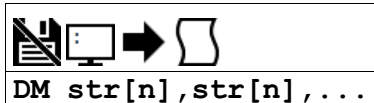
### Examples

```

Galil DMC Code Example
:DL; '          Begin Download
SB0
CB2
EN
\
:'Ended download
    
```

**DL applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**DM** *Dimension Array*

DM `str[n],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                       |
|------------|--------|---------|---------|------------|---------------------------------------------------------|-----------------------------|
| <b>str</b> | 1 char | 8 chars | N/A     | String     | Name of array to dimension                              |                             |
| <b>n</b>   | 1      | 400     | N/A     | 1          | Number of array elements to assign to dimensioned array | For RIO-47xx0               |
|            | 1      | 1,000   | N/A     | 1          | Number of array elements to assign to dimensioned array | For RIO-47xx2 and RIO-473xx |

**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.
- `DM ?` returns the available array space.
- The DM command can allocate any number of array in a single command up to the maximum command line length of the controller being used.

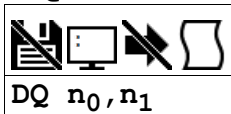
**Examples**

```
'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[100];' Define dimension of array Tests with 100 elements
```

**DM applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

## DQ *Change Analog Output Range*



DQ  $n_0, n_1$

| Usage           | DQ n ...                                                     | Arguments specified with an implicit, comma-separated order |
|-----------------|--------------------------------------------------------------|-------------------------------------------------------------|
| <b>Operands</b> | _DQ0<br>_DQ1<br>_DQ2<br>_DQ3<br>_DQ4<br>_DQ5<br>_DQ6<br>_DQ7 | Operand has special meaning, see Remarks                    |

### Description

The DQ command allows the ability to change the analog output range for individual channels.

### Arguments

| Argument             | Min | Max | Default | Resolution | Description                                         | Notes |
|----------------------|-----|-----|---------|------------|-----------------------------------------------------|-------|
| <b>n<sub>0</sub></b> | 0   | 7   | N/A     | 1          | Analog output channel to assign value               |       |
| <b>n<sub>1</sub></b> | 1   | 4   | 4       | 1          | Designate the analog output range. See table below. |       |

#### *Analog Output Range Setting*

| Argument             | Value | Description    | Notes   |
|----------------------|-------|----------------|---------|
| <b>n<sub>1</sub></b> | 1     | 0 to +5 VDC    |         |
|                      | 2     | 0 to +10 VDC   |         |
|                      | 3     | -5 to +5 VDC   |         |
|                      | 4     | -10 to +10 VDC | Default |

### Remarks

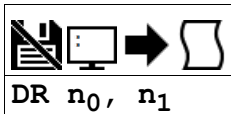
- This command is only valid for RIO models with configurable analog outputs. Check the ID command to see if the particular RIO model supports DQ.
- DQ is a configuration command which must be set at the beginning of application code.
- `_DQn` contains the present range setting.

### Examples

```
'Galil DMC Code Example
'Set output 1 to 0-10v
:DQ 1,2
```

**DQ applies to RIO47xxx,RIO574x0,DMC52xx0**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**DR** *Configures I O Data Record Update Rate*DR  $n_0$ ,  $n_1$ 

|                 |              |                                                             |
|-----------------|--------------|-------------------------------------------------------------|
| <b>Usage</b>    | DR n ...     | Arguments specified with an implicit, comma-separated order |
| <b>Operands</b> | _DR0<br>_DR1 | Operand has special meaning, see Remarks                    |

**Description**

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

For ethernet-based controllers, the controller creates a QR record and sends it to the unsolicited UDP Ethernet Handle at the specified rate. See the User Manual for the data record map.

**Arguments**

| Argument | Min | Max    | Default   | Resolution | Description                                            | Notes                                             |
|----------|-----|--------|-----------|------------|--------------------------------------------------------|---------------------------------------------------|
| $n_0$    | 2   | 30,000 | 0         | 1          | Data update rate specified in samples between packets. |                                                   |
|          | 0   | 0      | 0         | 0          | Turn off data record output                            |                                                   |
| $n_1$    | 0   | 2      | see Notes | 1          | Ethernet handle to output data record packet           | For RIO-47xx0. 0=A,1=B,2=C.                       |
|          | 0   | 4      | see Notes | 1          | Ethernet handle to output data record packet           | For RIO-47xx2 and RIO-473xx. 0=A,1=B,2=C,3=D,4=E. |

**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.
- If  $n_1$  is omitted, then the CF unsolicited message port is used by default.
- The DR port specified with  $n_1$  must be a UDP handle.
- \_DR0 contains the data record update rate ( $n_0$ ).
- \_DR1 contains the specified handle ( $n_1$ ). Will return an integer 0-7 for handles A-H.
- Issuing any of the following commands will pause the output of the data record until the command is complete: BN, BP, BV, DL, LS, LV, QD, QU, UL

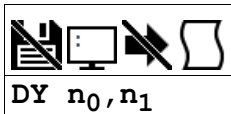
**Examples**

```
'Galil DMC Code Example
:WH
IHA
:DR1000,0
GX~P
_@~P
_H ~P
_0~P
:DR0
```

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

**DR applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**DY** PWM output duty cycleDY n<sub>0</sub>, n<sub>1</sub>

|                 |              |                                                             |
|-----------------|--------------|-------------------------------------------------------------|
| <b>Usage</b>    | DY n ...     | Arguments specified with an implicit, comma-separated order |
| <b>Operands</b> | _DY0<br>_DY1 | Operand holds the value last set by the command             |

**Description**

Sets the PWM duty cycle when using the PWM feature on outputs 14 and 15. PM enables the output of the PWM signal. FQ sets the PWM frequency.

**Arguments**

| Argument       | Min | Max | Default | Resolution | Description                                 | Notes       |
|----------------|-----|-----|---------|------------|---------------------------------------------|-------------|
| n <sub>0</sub> | 0   | 100 | 0       | 1          | Set PWM duty cycle percentage for output 14 | See Remarks |
| n <sub>1</sub> | 0   | 100 | 0       | 1          | Set PWM duty cycle percentage for output 15 | See Remarks |

**Remarks**

- Duty cycle is percentage ON with positive polarity, and percentage OFF with negative polarity (See PM for polarity)
- The -PWM option is necessary for precise resolution of the duty cycle. With the -PWM option the accuracy of the PWM output is +/- 0.5% of the DY setting.
- See the RIO user manual for more information.

**Examples**

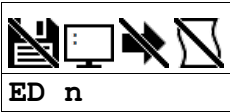
```
'Galil DMC Code Example
FQ 100,200;' Set output 14 to 100 Hz, and output 15 to 100 Hz
DY 50,25;' Set output 14 to 50%, output 15 to 25%
PM 1,1;' Turn PWM mode on for outputs 14 and 15
```

```
'Galil DMC Code Example
:MG _DY0;' Show duty cycle for OUT14
50.0000
```

**DY applies to RIO47xxx**©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@gallmc.com](mailto:documentation@gallmc.com)



## ED *Edit*



|                 |                     |                                                             |
|-----------------|---------------------|-------------------------------------------------------------|
| <b>Usage</b>    | ED n ...            | Arguments specified with an implicit, comma-separated order |
| <b>Operands</b> | _ED<br>_ED1<br>_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 Gall Software). In the Edit subsystem, programs can be created, changed, or destroyed.

### Arguments

| Argument | Min | Max | Default   | Resolution | Description                  | Notes                                                                                   |
|----------|-----|-----|-----------|------------|------------------------------|-----------------------------------------------------------------------------------------|
| n        | 0   | 199 | see Notes | 1          | Line number to begin editing | For RIO-47xx0. Default n is the last line of program space with commands.               |
|          | 0   | 399 | see Notes | 1          | Line number to begin editing | For RIO-47xx2 and RIO-473xx. Default n is the last line of program space with commands. |

### Remarks

- This command will be rejected by Gall software if sent to via the terminal. In order to edit a program using a Gall software package, use that package's prescribed programming interface (I.E. GDK's Editor Tool).
- 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

```
'Gallil 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.
```

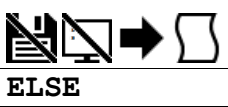
```
'Gallil DMC Code Example
'Using _ED4
XQ#id,1
XQ#id,2
XQ#id,3
XQ#id,4
XQ#id,5
XQ#id,6
XQ#id,7
#id
MG{z10.0}"This message is from thread",_ED4
EN

' 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
```

**ED applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**ELSE** *Else function for use with IF conditional statement***ELSE**

|              |            |                                                             |
|--------------|------------|-------------------------------------------------------------|
| <b>Usage</b> | 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**

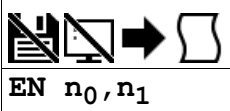
- None

**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
```

**ELSE applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**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.

**Arguments**

| Argument             | Value | Description                                                | Notes   |
|----------------------|-------|------------------------------------------------------------|---------|
| <b>n<sub>0</sub></b> | 0     | Return from subroutine without restoring trippoint         | Default |
|                      | 1     | Return from subroutine and restore trippoint               |         |
| <b>n<sub>1</sub></b> | 0     | Return from #COMINT without restoring CI interrupt trigger | Default |
|                      | 1     | Return from #COMINT and restore CI interrupt trigger       |         |

**Remarks**

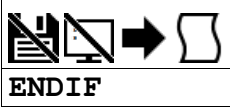
- The EN command is used to end the automatic subroutines #MCTIME #COMINT and #CMDERR.
  - Use the RI command to end the #ININT0,#ININT1,#ININT2 and #ININT3 subroutines

**Examples**

```
'Galil DMC Code Example
#A: '      Program A
SB1: '    Set output 1 high
WT500; '   wait for 500 msec
CB1: '    Set output 1 low
MG "DONE"; 'Print message
EN; '     End of Program
```

**EN applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**ENDIF** *End of IF conditional statement***ENDIF**

|              |             |                                                             |
|--------------|-------------|-------------------------------------------------------------|
| <b>Usage</b> | 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**

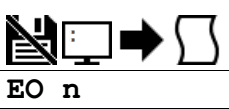
- None

**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
```

**ENDIF applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**EO** *Echo*

|                 |          |                                                             |
|-----------------|----------|-------------------------------------------------------------|
| <b>Usage</b>    | EO n ... | Arguments specified with an implicit, comma-separated order |
| <b>Operands</b> | _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 EOO
- The EO command is accepted over the serial port only.
  - The ethernet port will not echo commands

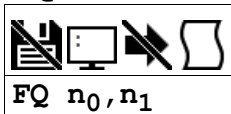
**Examples**

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

**EO applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

## FQ PWM output frequency



FQ n<sub>0</sub>, n<sub>1</sub>

|                 |              |                                                             |
|-----------------|--------------|-------------------------------------------------------------|
| <b>Usage</b>    | FQ n ...     | Arguments specified with an implicit, comma-separated order |
| <b>Operands</b> | _FQ0<br>_FQ1 | Operand holds the value last set by the command             |

### Description

Sets the PWM frequency when using the PWM feature on outputs 14 and 15. DY sets the duty cycle and PM enables the PWM output feature.

### Arguments

| Argument       | Min | Max    | Default | Resolution | Description                                 | Notes                             |
|----------------|-----|--------|---------|------------|---------------------------------------------|-----------------------------------|
| n <sub>0</sub> | 10  | 20,000 | 0       | 1          | Frequency of PWM signal for output 14 in Hz | PWM Resolution defined in Remarks |
| n <sub>1</sub> | 10  | 20,000 | 0       | 1          | Frequency of PWM signal for output 15 in Hz | PWM Resolution defined in Remarks |

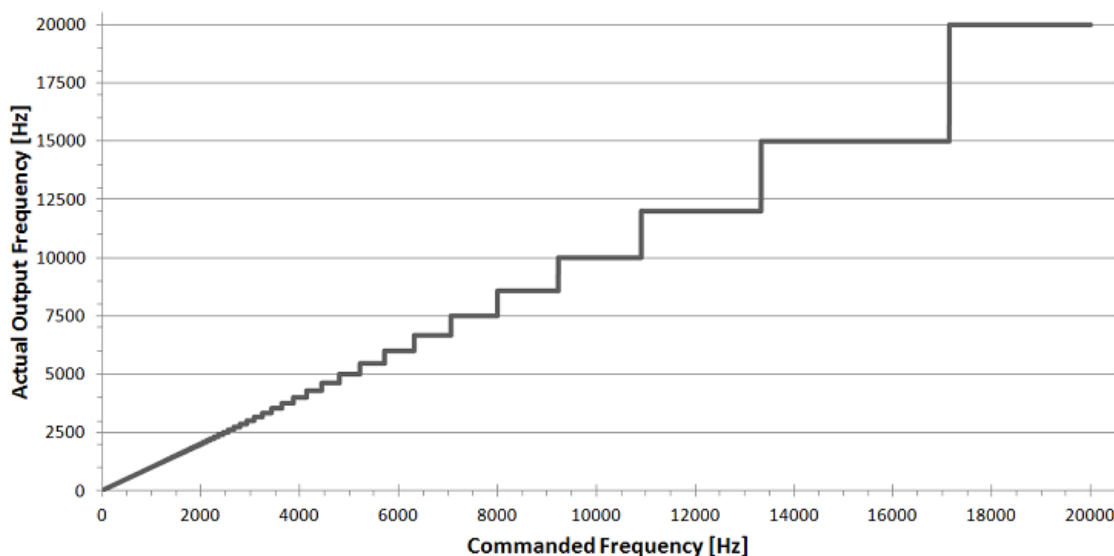
### Remarks

- The -PWM option is necessary in order to obtain the full frequency range for the PWM output. See the RIO user manual for more information.
- **Usage note:** The FQ command should be set to a valid value before the PWM output is enabled.
- The actual resolution of the PWM frequency depends on the frequency set. See table and figures below and refer to the user manual for the equations necessary to determine the exact frequency the PWM feature will generate.

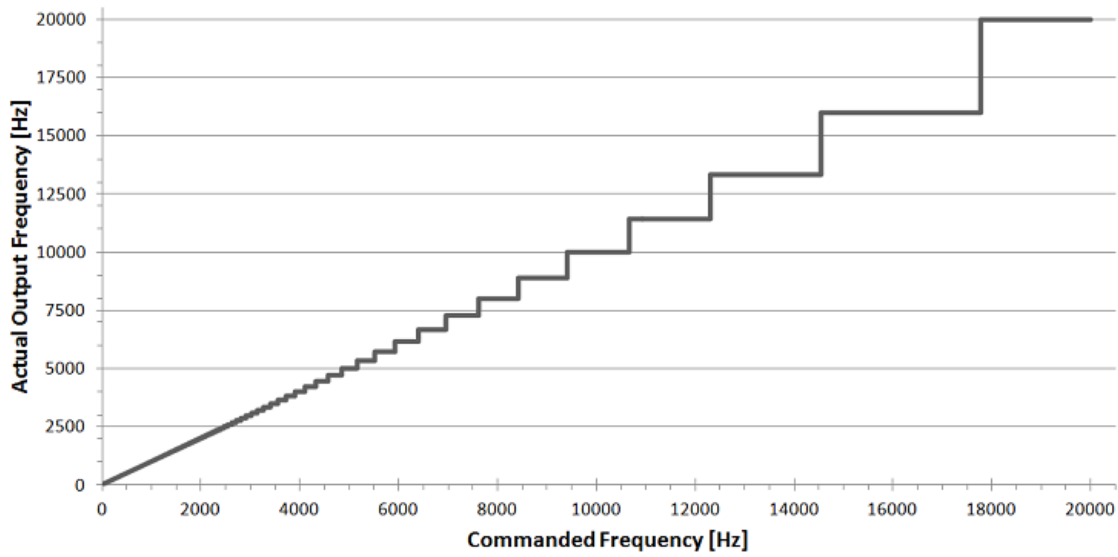
#### PWM Resolution

| RIO model | Resolution Type Below |
|-----------|-----------------------|
| 47100     | I                     |
| 47120     | I                     |
| 47200     | I                     |
| 47102     | II                    |
| 47122     | II                    |
| 47142     | II                    |
| 47202     | II                    |
| 47300     | II                    |

FQ Commanded versus Actual Frequency Output  
Resolution Type I



FQ Commanded versus Actual Frequency Output  
Resolution Type II



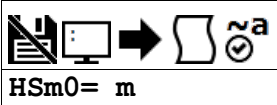
### Examples

```
'Galil DMC Code Example
:FQ 100,200;' Set output 14 to 100 Hz, and output 15 to 100 Hz
:DY 50,25;' Set output 14 to 50%, output 15 to 25%
:PM 1,1;' Turn PWM mode on for outputs 14 and 15
```

**FQ applies to RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)



**HS Handle Assignment Switch**

HSm0= m

|              |          |                                                                                                 |
|--------------|----------|-------------------------------------------------------------------------------------------------|
| <b>Usage</b> | HSm0= m  | Arguments are single axis masks and are specified with a single axis mask and an assignment (=) |
|              | HS m ... | Arguments are single axis masks specified with an implicit, comma-separated order               |

**Description**

The HS command is used to switch the ethernet handle assignments between two handles. Handles are opened when a connection is established by an external client (TCP or UDP), or when a handle is assigned explicitly with the IH command. Should those assignments need modifications, the HS command allows the handles to be reassigned.

**Arguments**

| Argument  | Min | Max | Default | Resolution | Description             | Notes                                                    |
|-----------|-----|-----|---------|------------|-------------------------|----------------------------------------------------------|
| <b>m0</b> | A   | C   | N/A     | Handle     | First handle to switch  | For RIO-47xx0                                            |
|           | A   | E   | N/A     | Handle     | First handle to switch  | For RIO-47xx2 and RIO-473xx                              |
|           | S   | S   | N/A     | Handle     | First handle to switch  | S = current handle sending command. Not valid in program |
| <b>m</b>  | A   | C   | N/A     | Handle     | Second handle to switch | For RIO-47xx0                                            |
|           | A   | E   | N/A     | Handle     | Second handle to switch | For RIO-47xx2 and RIO-473xx                              |
|           | S   | S   | N/A     | Handle     | Second handle to switch | S = current handle sending command. Not valid in program |

**Remarks**

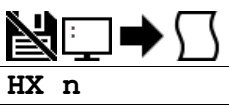
- A handle encapsulates the following 4 pieces of information:
  - 1. Local IP address (same for all handles)
  - 2. Remote IP address
  - 3. Local Port
  - 4. Remote Port
- Handles are used as a pointer to the network socket in commands such as SAh, MBh, {Eh}, and IHh where h is the handle letter

**Examples**

```
'Galil DMC Code Example
:HSC= D;' Connection for handle C is assigned to handle D. Connection for handle D is assigned to handle C.
:HSS= E;' Executing handle connection is assigned to handle E. Connection for handle E is assigned to executing handle.
```

**HS applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@gallinc.com](mailto:documentation@gallinc.com)

**HX** *Halt Execution*

HX n

|                 |                              |                                                             |
|-----------------|------------------------------|-------------------------------------------------------------|
| <b>Usage</b>    | HX n ...                     | Arguments specified with an implicit, comma-separated order |
| <b>Operands</b> | _HX0<br>_HX1<br>_HX2<br>_HX3 | 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   | 3   | 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

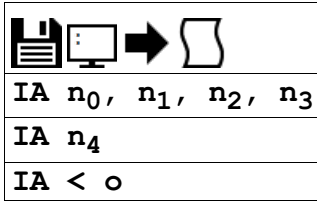
**Examples**

```
'Galil DMC Code Example
XQ #A;' Execute program #A, thread zero
XQ #B,3;' Execute program #B, thread three
HX0;' Halt thread zero
HX3;' Halt thread three
```

**HX applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**IA IP Address**



| Usage           | IA n ... | Arguments specified with an implicit, comma-separated order |
|-----------------|----------|-------------------------------------------------------------|
| <b>Operands</b> | _IA0     | Operand has special meaning, see Remarks                    |
|                 | _IA1     |                                                             |
|                 | _IA2     |                                                             |
|                 | _IA3     |                                                             |
|                 | _IA4     |                                                             |
|                 | _IA5     |                                                             |

**Description**

The IA command assigns the controller IP address and the TCP time out. The IP address can also be assigned via Galil software or from an external server. The controller defaults to DHCP and will receive an IP address from a DHCP server if present. To manually set an IP address over the serial connection, send DH0 to disable DHCP prior to setting the new IP address with IA. GalilTools and GalilSuite software packages feature a DHCP/BOOTP capability to assign the IP address to the controller. Please refer to the user manuals for those products for more information.

**Arguments**

| Argument             | Min                | Max           | Default | Resolution | Description                                                               | Notes                 |
|----------------------|--------------------|---------------|---------|------------|---------------------------------------------------------------------------|-----------------------|
| <b>n<sub>0</sub></b> | 0                  | 255           | 0       | 1          | Byte 3 of the IP address                                                  |                       |
| <b>n<sub>1</sub></b> | 0                  | 255           | 0       | 1          | Byte 2 of the IP address                                                  |                       |
| <b>n<sub>2</sub></b> | 0                  | 255           | 0       | 1          | Byte 1 of the IP address                                                  |                       |
| <b>n<sub>3</sub></b> | 0                  | 255           | 0       | 1          | Byte 0 of the IP address                                                  |                       |
| <b>n<sub>4</sub></b> | -<br>2,147,483,648 | 2,147,483,647 | 0       | 1          | The full IP address specified as a signed 32 bit two's complement integer |                       |
| <b>o</b>             | 1                  | 2,147,483,647 | 250     | 1          | The time in update samples between TCP retries                            | Up to 5 retries occur |

**Remarks**

- When specifying the IP address with IA, remember to use commas as delimiters instead of periods.
- n<sub>4</sub> = ? will return the IP address of the controller in comma separated format.
- Setting the IP address over Ethernet to a new value will cause an immediate disconnect/timeout. Reconnect to the controller on the new IP address and issue a BN to save the new value to flash.
- To change the IP address manually over Ethernet on a controller which was initially assigned via DHCP, send "DH 0;IA n<sub>0</sub>,n<sub>1</sub>,n<sub>2</sub>,n<sub>3</sub>" as one command line. Reconnect on the new IP and issue BN to save.

**Operands**

- \_IA0 contains the IP address representing a 32 bit signed number (Two's complement). See the example below.
- \_IA1 contains the value for o (retry time).
- \_IA2 contains the number of available handles.
- \_IA3 contains the number of the handle using this operand where the number is 0 to 7. 0 represents handle A, 1 handle B, etc. This is used by a remote device to detect its outgoing handle (see WH).
- \_IA4 contains the number of the handle that lost communication last, contains a -1 on reset to indicate no handles lost.
- \_IA5 returns autonegotiation Ethernet speed. Returns 10 for 10-Base T and returns 100 for 100-Base T, it will return -1 if there is no physical link.

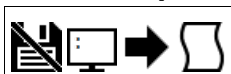
**Examples**

```
'Galil DMC Code Example
IA 151,12,53,89;' Assigns the controller with the address 151.12.53.89
IA 2534159705;' Assigns the controller with the address 151.12.53.89
IA < 500;' Sets the timeout value to 500 msec
```

```
'Galil DMC Code Example
REM The individual IP address bytes can be derived within embedded code using _IA0
a=@INT[ (_IA0&($FF000000))/$1000000]&$FF
b=@INT[ (_IA0&($00FF0000))/$10000]
c=@INT[ (_IA0&($0000FF00))/$100]
d=@INT[ (_IA0&($000000FF))]
REM IP address = a.b.c.d
```

**IA applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**ID Identify**

ID

|              |    |                            |
|--------------|----|----------------------------|
| <b>Usage</b> | ID | Command takes no arguments |
|--------------|----|----------------------------|

**Description**

The ID command is used to query the controller for the hardware configuration and factory programming.

**Arguments**

ID is a command with no arguments

**Remarks**

- Refer to the Examples section for actual controller responses
- The following are descriptions of the ID response

The ID command follows this pattern:

outputs 0-3 = [power] [type] outputs

where

[power] - power indicates the outputs are high power outputs. No string here indicates low power outputs

[type]- specifies whether the outputs are sourcing or sinking

Output information is displayed in sets of 4 bits at a time, until all outputs have been described.

analog inputs 0-3 = [bit] [configuration]

where

[bit] - '12 bits' indicates 12 bit resolution of the analog inputs

'16 bits' indicates 16 bit resolution of the analog inputs

[configuration] - 'range of 0 to 5' indicates the unit has non-configurable analog inputs.

- 'programmable range(AQ)' indicates the unit has configurable analog inputs.

Analog input information is displayed in sets of 4 signals at a time, until all inputs have been described.

analog outputs 0-3 = [bit] [configuration]

where

[bit] - '12 bits' indicates 12 bit resolution of the analog inputs

'16 bits' indicates 16 bit resolution of the analog inputs

[configuration] - 'range of 0 to 5' indicates the unit has non-configurable analog outputs.

- 'programmable range(DQ)' indicates the unit has configurable analog outputs.

Analog output information is displayed in sets of 4 signals at a time, until all outputs have been described.

real time clock

Presence of the string 'real time clock' indicates power-loss persistent, real time clock circuitry

encoder options=[encoder\_0] [encoder\_1] [encoder\_2] [encoder\_3]

where

[encoder\_n] = 'ssi' for SSI encoder input for the specified channel

'biss' for BiSS encoder input for the specified channel

'quad' for quadrature encoder input for the specified channel

**Examples**

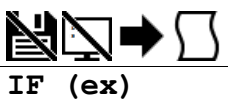
```
'Galil DMC Code Example
:ARV
RIO47200 Rev 1.1j1
:ID
outputs 0-3 = power sourcing outputs
outputs 4-7 = power sourcing outputs
outputs 8-11 = power sourcing outputs
outputs 12-15 = power sourcing outputs
analog inputs 0-3 = 12 bits range 0 to 5
analog inputs 4-7 = 12 bits range 0 to 5
analog outputs 0-3 = none
analog outputs 4-7 = none
```

```
'Galil DMC Code Example
:ARV
RIO47300 Rev 1.1b
:ID
outputs 0-3 = power sourcing outputs
outputs 4-7 = power sourcing outputs
outputs 8-11 = power sourcing outputs
outputs 12-15 = power sourcing outputs
outputs 16-19 = power sourcing outputs
outputs 20-23 = power sourcing outputs
analog inputs 0-3 = 12 bits programmable range(AQ)
analog inputs 4-7 = 12 bits programmable range(AQ)
analog outputs 0-3 = 12 bits programmable range(DQ)
analog outputs 4-7 = 12 bits programmable range(DQ)
```

---

**ID applies to DMC500x0,DMC40x0,DMC42x0,DMC41x3,DMC30010,RIO47xxx,DMC52xx0,RIO574x0,EDD3701x**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**IF** *IF conditional statement***IF** (ex)

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

**Description**

The IF command is used in conjunction with an ENDF 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 ENDF 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 a 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
#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 simplified 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
```

**IF applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**IH** *Open IP Handle*

|                                                                                  |
|----------------------------------------------------------------------------------|
|  |
| IHm= n <sub>0</sub> ,n <sub>1</sub> ,n <sub>2</sub> ,n <sub>3</sub> <o> >p       |
| IHm= n <o> >p                                                                    |
| IHm= >p                                                                          |

| Usage           | IHm= n                                    | Arguments specified with a single axis mask and an assignment (=) |
|-----------------|-------------------------------------------|-------------------------------------------------------------------|
| <b>Operands</b> | _IHm0<br>_IHm1<br>_IHm2<br>_IHm3<br>_IHm4 | Operand has special meaning, see Remarks                          |

**Description**

The IH command is used when the controller is operated as a master (client) to open a handle and connect to a slave (server).

The IH command is also used to set the realtime clock from a TIME protocol server.

**Arguments**

| Argument             | Min                | Max           | Default   | Resolution | Description                                                | Notes                                                             |
|----------------------|--------------------|---------------|-----------|------------|------------------------------------------------------------|-------------------------------------------------------------------|
| <b>m</b>             | A                  | C             | N/A       | Handle     | Handle to assign connection                                | For RIO-47xx0                                                     |
|                      | A                  | E             | N/A       | Handle     | Handle to assign connection                                | For RIO-47xx2 and RIO-473xx                                       |
|                      | S                  | T             | N/A       | Handle     | Special handle designator used when closing handles        | See Remarks                                                       |
| <b>n<sub>0</sub></b> | 0                  | 255           | 0         | 1          | Byte 3 of the slave IP address                             |                                                                   |
| <b>n<sub>1</sub></b> | 0                  | 255           | 0         | 1          | Byte 2 of the slave IP address                             |                                                                   |
| <b>n<sub>2</sub></b> | 0                  | 255           | 0         | 1          | Byte 1 of the slave IP address                             |                                                                   |
| <b>n<sub>3</sub></b> | 0                  | 255           | 0         | 1          | Byte 0 of the slave IP address                             |                                                                   |
| <b>n</b>             | -<br>2,147,483,648 | 2,147,483,647 | 0         | 1          | Slave IP address in its 32 bit value                       |                                                                   |
| <b>o</b>             | 0                  | 65,535        | see Notes | 1          | Specify the slave port to connect over                     | If o is omitted, the controller selects the port starting at 1000 |
| <b>p</b>             | 1                  | 2             | 2         | 1          | Specify the connection type to open                        | n = 2 is TCP. n = 1 is UDP.                                       |
|                      | -3                 | -1            | N/A       | 1          | Specify the connection type to close when closing a handle | See Remarks                                                       |

**Remarks**

- All 4 bytes must be assigned for an IP address to be valid.
- IHm=? returns the IP address as 4, 1-byte numbers.
- Use the following equation to change the 4 byte IP (n<sub>0</sub>,n<sub>1</sub>,n<sub>2</sub>,n<sub>3</sub>) to a single 32 bit number, n.
  - $n = (n_0 * 2^{24}) + (n_1 * 2^{16}) + (n_2 * 2^8) + n_3$ .
- When using Modbus, port 502, note that Galil Modbus supports one master per slave.

**Opening a Handle**

- To open a handle, the user must specify:
  - The IP address of the slave.
  - (optional) The port number of the slave. If not specified, the firmware will choose a port.
    - Modbus connections must always be specified as port 502.
  - (optional) The connection type as TCP/IP or UDP/IP. If not specified, the controller will make a TCP connection.
- Issue the IH command on an available handle with the correct settings for IP (n<sub>0</sub>-n<sub>3</sub>), port (o) and connection type (p).
  - See TH to list handle status.

**Closing a Handle**

- Closing multiple handles is done with the T handle identifier along with a connection type p selector.
  - IHT => p closes all handles matching the p type selector, where p = -1 closes UDP handles, p = -2 closes TCP handles and p = -3 closes all handle types.
- Closing individual handles, other than the handle being used to send the IH command, is done with IHN => -1 where n is the handle to close (A-H).
- Closing the handle that sent the command is done with the S handle identifier, along with connection type p selector.
  - IHS => p closes the handle that sent the command if its type matches the p selector, where p = -1 closes UDP handles, p = -2 closes TCP handles and p = -3 closes all handle types.

**Time server**

- Connecting to the time server requires using port 37 with a TCP connection. Port 37 is used by the TIME protocol server. Only TCP is supported.
- The server transmits a 32 bit unsigned integer in network byte order representing the number of seconds since midnight, January 1, 1900 GMT.
- The Command RO is used to set the timezone offset from GMT and to see the raw value returned by the time server

**Operand Usage**

| Operand     | Reported Value            | Description of Value                                          | Notes                    |
|-------------|---------------------------|---------------------------------------------------------------|--------------------------|
| <u>IHm0</u> | -2147483648 to 2147483648 | IP address of handle m as a 32 bit number (n)                 |                          |
| <u>IHm1</u> | 0 to 65535                | Slave port number for handle m                                |                          |
| <u>IHm2</u> | 0                         | Handle is free                                                | Handle 'Available' in TH |
|             | 1                         | Handle connected as UDP slave                                 |                          |
|             | 2                         | Handle connected as TCP slave                                 |                          |
|             | -1                        | Handle connected as UDP master                                |                          |
|             | -2                        | Handle connected as TCP master                                |                          |
|             | -5                        | Attempting to establish UDP handle                            |                          |
|             | -6                        | Attempting to establish TCP handle                            |                          |
| <u>IHm3</u> | 0                         | ARP was successful                                            |                          |
|             | 1                         | ARP failed or still in progress                               |                          |
| <u>IHm4</u> | 1                         | Waiting for ACK from slave controller after issuing a command |                          |
|             | 2                         | Received ":" as response to a command                         |                          |
|             | 3                         | Received "?" as response to a command                         |                          |
|             | 4                         | Connection timed-out waiting for a response to a command      |                          |

## Examples

```
'Galil DMC Code Example
IHA=251,29,51,1;' Open handle A at IP address 251.29.51.1
'TCP is used as default
IHA= -2095238399;' Open handle A at IP address 251.29.51.1
'when the IH command is given,
'the controller initializes an ARP
'on the slave device before opening a handle.
'This operation can cause a small time delay
'before the controller responds
```

```
'Galil DMC Code Example
'setting up a modbus handle
MW1;' setup modbus wait
IHE= 192,168,100,200<502>2;' setup a modbus handle to slave
#wt;' wait for handle to be connected
WT2;' before issuing a command
JP#wt, _IHE2<>-2;'
SB5003;' Set output 3 on slave
WT1000;' 1 second wait
MBE= ,5,3,0;' Clear output 3 using MB command
EN
```

```
'Galil DMC Code Example
'Setting and printing the time with a TIME protocol server
#set
RO-7;' set the timezone offset for Pacific Daylight Time
IHE=>-3;' close handle E in case it's open
IHE=10,0,62,23<37>2;' query the TIME server
WT10;' wait briefly for the transaction to occur
MG_RO1{${8.0}};' display the raw data returned from the server
JS#print;' call the time print subroutine
EN

#print
MG_RT2{F2.0},":"{N};' print the current hours
MG_RT1{F2.0},":"{N};' print the current minutes
MG_RT0{F2.0};' print the current seconds
EN

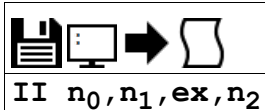
'Example output:
'$CF93127A
' 16: 17: 14
```

**IH applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)



## II Input Interrupt



II  $n_0, n_1, ex, n_2$

|              |          |                                                             |
|--------------|----------|-------------------------------------------------------------|
| <b>Usage</b> | II n ... | Arguments specified with an implicit, comma-separated order |
|--------------|----------|-------------------------------------------------------------|

### Description

The II command enables the input interrupt function for the specified inputs.

This function can trigger one of four input interrupt subroutines (#ININTn) when the controller sees that the conditional statement is satisfied. This subroutine will be executed in the specified thread, causing any trippoint set in that thread to be cleared. The cleared trippoint can be re-enabled by the proper termination of the interrupt subroutine using RI.

### Arguments

| Argument             | Min | Max | Default | Resolution | Description                                            | Notes                                                                           |
|----------------------|-----|-----|---------|------------|--------------------------------------------------------|---------------------------------------------------------------------------------|
| <b>n<sub>0</sub></b> | 0   | 3   | N/A     | 1          | Specify the #ININTn routine to use for input interrupt | n <sub>0</sub> =0 is #ININT0, n <sub>0</sub> =1 is #ININT1, and so on           |
| <b>n<sub>1</sub></b> | -1  | 3   | -1      | 1          | Thread number for #ININTn routine to run               | n <sub>1</sub> =-1 disables input interrupt                                     |
| <b>ex</b>            | N/A | N/A | N/A     | Expression | Inputs to be used for interrupt routine                | -IN= input must go low, +IN=input must go high, see Remarks                     |
| <b>n<sub>2</sub></b> | 0   | 1   | 0       | 1          | Sets the condition between multiple inputs             | n <sub>2</sub> =0 sets condition to AND. n <sub>2</sub> =1 sets condition to OR |

### Remarks

- The specified thread, n<sub>1</sub>, needs to be running when the interrupt occurs, otherwise the #ININTn subroutine will not be executed.
  - Upon interrupt, the existing thread n<sub>1</sub> will be interrupted to allow the execution of the interrupt subroutine. Upon completion of the interrupt, the main program in thread n<sub>1</sub> will once again be enabled from the point at which the interrupt occurred.
- ex is a statement of all inputs to be used, with polarity defining the input level to trigger the interrupt. All inputs specified must be separated by & characters
  - For example II0,0,1&2&-3 sets the interrupt to trigger when inputs 1 and 2 are logic '1' and input 3 is logic '0'
- If n<sub>2</sub> is omitted or is 0, the input statements must all be true to trigger #ININTn (&). If type is 1, then any true input statement will trigger #ININTn (!)
- The RI command is used to return from the #ININTn routines.
- Note: An application program must be running on the controller for the interrupt function to work.
- To avoid returning to the program on an interrupt, use the command ZS to zero the subroutine stack and use the II command to reset the interrupt.
- Any digital inputs can be used in an II expression with the exception of the RIO 47300 which is limited to inputs 0 to 15

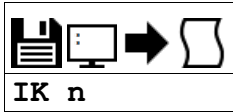
### Examples

```
'Galil DMC Code Example
#A;'          Program A
II 2,0,3&5&-10,1;'  Specify interrupt #2 on main thread when
                    inputs 3 OR 5 go high, OR 10 goes low
#LOOP;JP #LOOP;'  Loop
EN;'           End Program
#ININT2;'       Interrupt subroutine number 2
MG "INTERRUPT";' Print Message
AI -1;'        wait for input to switch states before continuing.
                    Otherwise we'll jump back in to #ININT2
RI 1,0;'       Return to main program, re-arm the input
                    interrupt and the interrupt routine.
```

**II applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

# IK Block Ethernet ports



|              |          |                                                             |
|--------------|----------|-------------------------------------------------------------|
| <b>Usage</b> | IK n ... | Arguments specified with an implicit, comma-separated order |
|--------------|----------|-------------------------------------------------------------|

## Description

The IK command blocks client connections to the controller on most ports below port number 1000. Specific port numbers and ports above 1000 are unaffected.

## Arguments

| Argument | Value | Description                                              | Notes                                                        |
|----------|-------|----------------------------------------------------------|--------------------------------------------------------------|
| n        | 0     | Allow controller to receive Ethernet packets on any port |                                                              |
|          | 1     | Blocks Ethernet packets on ports lower than 1000.        | Default. Ports 0, 23, 25, 37, 68, 80 and 502 are unaffected. |

## Remarks

- A Galil Ethernet controller simultaneously operates as a server (listening for Ethernet connections from a client) and a client (able to create connections to a server).
- Ports 0, 23, 68 and 502 are used for standard client connections to the controller.
- Port 37 is used for TIME protocol synchronization. This mechanism has the controller connect as a client to the TIME server.
- Port 25 and 80 are allowed connections for SMTP Mail support and the Web Server Feature.


## Examples

```
'Galil DMC Code Example
:IK1;' Blocks undesirable port communication
:IK0;' Allows all Ethernet ports to be used
```

**IK applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

## IL Integrator Limit

|                                                                                  |
|----------------------------------------------------------------------------------|
|  |
| ILm= n                                                                           |
| IL n, n, n, n, n, n, n, n, n, n                                                  |

|                 |          |                                                                   |
|-----------------|----------|-------------------------------------------------------------------|
| <b>Usage</b>    | ILm= n   | Arguments specified with a single axis mask and an assignment (=) |
|                 | IL n ... | Arguments specified with an implicit, comma-separated order       |
| <b>Operands</b> | _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.

For example: IL 2 will limit the output for the A process control loop integrator to +/-2 Volt range.

### Arguments

| Argument | Min | Max    | Default | Resolution | Description                        | Notes                   |
|----------|-----|--------|---------|------------|------------------------------------|-------------------------|
| <b>m</b> | A   | B      | N/A     | Channel    | Channel to assign value            | RIO-47xx0               |
|          | A   | F      | N/A     | Channel    | Channel to assign value            | RIO-47xx2 and RIO-473xx |
| <b>n</b> | 0   | 9.9982 | 9.9982  | 20/65,536  | Value of Integrator limit in volts | RIO-4712x               |
|          | 0   | 5      | 5       | 5/4,096    | Value of Integrator limit in volts | RIO-4710x               |

### 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.
- The process control loop input and output (AF and AZ) must be set prior to adjusting IL.
- For further details see the Process Control section in the User's Manual.

### Examples

```
'Galil DMC Code Example
AF0,1;'      Set analog feedback
AZ0,1;'      Set analog output
IL 3,2;'     Integrator limits
IL ?;'       Returns the first limit
```

**IL applies to DMC500x0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2,EDD3701x**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**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       |
|------------|---------|----------|---------|------------|------------------------------------------------|-------------|
| <b>str</b> | 0 chars | 34 chars | N/A     | String     | The prompt message                             | See Remarks |
| <b>arg</b> | 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.
- Backslash '\' character will clear the IN command trippoint. The variable will not be overwritten in (will be last set value).
- The IN command can only be used with a serial connection to the controller
- Input Interrupts will still be active during the prompt

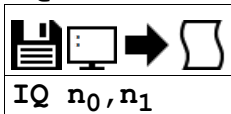
**Examples**

```
'Galil DMC Code Example
'Operator specifies a bit that is to be turned on.
#A
'Prompt operator for bit to set high
IN "Enter bit number to turn on",N1
'
SB N1;'      Set the specified bit high
MG "DONE";'  Print Message
EN;'        End Program
```

**IN applies to DMC500x0,DMC40x0,DMC42x0,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@gallmc.com](mailto:documentation@gallmc.com)

## IQ Digital Input Configuration



IQ n<sub>0</sub>, n<sub>1</sub>

|                 |              |                                                             |
|-----------------|--------------|-------------------------------------------------------------|
| <b>Usage</b>    | IQ n ...     | Arguments specified with an implicit, comma-separated order |
| <b>Operands</b> | _IQ0<br>_IQ1 | Operand has special meaning, see Remarks                    |

### Description

The IQ command sets the bitwise active level for each of the digital inputs. The input for IQ is a bitmask representing the inputs of the controller. When bit n of IQ is 0, then current flowing through the opto of input n returns a 0. When bit n is 1, current flow returns a 1.

### Arguments

| Argument       | Min | Max        | Default | Resolution | Description                           | Notes                         |
|----------------|-----|------------|---------|------------|---------------------------------------|-------------------------------|
| n <sub>0</sub> | 0   | 65,535     | 0       | 1          | Active level setting for inputs 0-15  | RIO-471xx and RIO-472xx       |
|                | 0   | 16,777,215 | 0       | 1          | Active level setting for inputs 0-24  | RIO-473xx                     |
| n <sub>1</sub> | 0   | 16,777,215 | 0       | 1          | Active level setting for inputs 25-47 | RIO-473xx with -24ExIn option |

### Remarks

- Setting IQ to the Min value means current flowing through the opto results in logic 0 for all inputs (@IN[n]=0). Current not flowing through the opto results in logic 1 (@IN[n]=1).
- Setting IQ to the Max value means means current not flowing through the opto is logic 0 for all inputs (@IN[n]=0). Current flowing through the opto results in logic 1 (@IN[n]=1).
- Current flowing through the opto can be seen on the RIO when the green LED turns ON.
- IQ is not queriable with a ?
- \_IQn contains the current setting for IQ


### Examples

```
'Galil DMC Code Example
:IQ255;'This sets inputs 0-7 such that current flowing results in logic 1
:
:IQ192;'Current flowing on inputs 6 and 7 results in logic 1
'current flowing on 0-5 and 8-15 are logic 0
```

**IQ applies to RIO47xxx,RIO574x0**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

## JP Jump to Program Location



|                      |
|----------------------|
| <b>JP #str, (ex)</b> |
| <b>JP n, (ex)</b>    |

|              |          |                                                             |
|--------------|----------|-------------------------------------------------------------|
| <b>Usage</b> | 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                                                       |
|------------|--------|-----------|---------|------------|-------------------------------------------------------------------|-------------------------------------------------------------|
| <b>str</b> | 1 char | 7 chars   | N/A     | String     | Label name for jump destination                                   | Must be a valid label in application code                   |
| <b>n</b>   | 0      | see Notes | N/A     | 1          | Line number for jump destination                                  | Maximum is number of lines of controller program memory - 1 |
| <b>ex</b>  | 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
```

**JP applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

## JS Jump to Subroutine



JS #str, (ex)

JS n, (ex)

|              |        |                                                                   |
|--------------|--------|-------------------------------------------------------------------|
| <b>Usage</b> | JSm= n | Arguments specified with a single axis mask and an assignment (=) |
|--------------|--------|-------------------------------------------------------------------|

### 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      | see Notes | N/A     | 1          | Line number for jump destination                                  | Maximum is maximum number of lines of program memory |
| 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                        |

### Remarks

- JS can be nested, called up to 16 deep
- 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 |

### 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
```

JS applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: documentation@galilmc.com



## KD *Derivative Constant*

|                                                                                  |
|----------------------------------------------------------------------------------|
|  |
| KDm= n                                                                           |
| KD n, n, n, n, n, n, n, n, n, n                                                  |

|                 |          |                                                                   |
|-----------------|----------|-------------------------------------------------------------------|
| <b>Usage</b>    | KDm= n   | Arguments specified with a single axis mask and an assignment (=) |
|                 | KD n ... | Arguments specified with an implicit, comma-separated order       |
| <b>Operands</b> | _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:

$$D(z) = KP + KD \frac{z-1}{z} + KI \frac{z}{2(z-1)}$$

### Arguments

| Argument | Min | Max       | Default | Resolution | Description              | Notes                   |
|----------|-----|-----------|---------|------------|--------------------------|-------------------------|
| <b>m</b> | A   | F         | N/A     | Channel    | Channel to assign value  | RIO-47xx2 and RIO-473xx |
|          | A   | B         | N/A     | Channel    | Channel to assign value  | RIO-47xx0               |
| <b>n</b> | 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
- RIO-47xx2 and RIO-473xx have 6 control loops (KD n,n,n,n,n,n). RIO-47xx0 has 2 control loops (KD n,n)
- For further details see the Process Control section in the RIO user manual.

### Examples


```
'Galil DMC Code Example
:KD 12,14;' Implicit notation to set channel A and B term
:KDB= 14;' Explicit notation to set channel B only
:KD ,8;' Implicit notation to set B
:KD ?,?;' Return A,B values
12, 14
:KDB= ?;' Return B value
14
:MG _KDA;' Message the operand for the A channel
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;' Turn off position error limit
OF 1,2;' Set one volt on A and two volts on B
EN
```

**KD applies to DMC500x0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2,EDD3701x**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

## KI Integrator

|                                                                                  |
|----------------------------------------------------------------------------------|
|  |
| <b>KIm= n</b>                                                                    |
| <b>KI n, n, n, n, n, n, n, n, n</b>                                              |

|                 |          |                                                                   |
|-----------------|----------|-------------------------------------------------------------------|
| <b>Usage</b>    | KIm= n   | Arguments specified with a single axis mask and an assignment (=) |
|                 | KI n ... | Arguments specified with an implicit, comma-separated order       |
| <b>Operands</b> | _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:

$$D(z) = KP + KD \frac{z-1}{z} + KI \frac{z}{2(z-1)}$$

### Arguments

| Argument | Min | Max     | Default | Resolution | Description             | Notes                   |
|----------|-----|---------|---------|------------|-------------------------|-------------------------|
| <b>m</b> | A   | F       | N/A     | Channel    | Channel to assign value | RIO-47xx2 and RIO-473xx |
|          | A   | B       | N/A     | Channel    | Channel to assign value | RIO-47xx0               |
| <b>n</b> | 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
- RIO-47xx2 and RIO-473xx have 6 control loops (KI n,n,n,n,n,n). RIO-47xx0 has 2 control loops (KI n,n)
- For further details see the Process Control section in the RIO user manual.

### Examples


```
'Galil DMC Code Example
:KIB= 14;' Explicit notation to set channel B only
:KI ,8;' Implicit notation to set B
:KI ?,?;' Return A,B values
7, 14
:KIB= ?;' Return B value
14
:MG _KIA;' Message the operand for the A channel
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
```

**KI applies to DMC500x0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2,CLS,EDD3701x**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@gallimc.com](mailto:documentation@gallimc.com)

## KP *Proportional Constant*

|                                                                                  |
|----------------------------------------------------------------------------------|
|  |
| KPm= n                                                                           |
| KP n, n, n, n, n, n, n, n, n, n                                                  |

|                 |          |                                                                   |
|-----------------|----------|-------------------------------------------------------------------|
| <b>Usage</b>    | KPm= n   | Arguments specified with a single axis mask and an assignment (=) |
|                 | KP n ... | Arguments specified with an implicit, comma-separated order       |
| <b>Operands</b> | _KPm     | Operand holds the value last set by the command                   |

### Description

KP designates the proportional constant in the controller filter. The proportional gain outputs a control signal proportional to the amount of error. The filter transfer function follows.

$$D(z) = KP + KD \frac{z-1}{z} + KI \frac{z}{2(z-1)}$$

### Arguments

| Argument | Min | Max       | Default | Resolution | Description                | Notes                   |
|----------|-----|-----------|---------|------------|----------------------------|-------------------------|
| <b>m</b> | A   | F         | N/A     | Channel    | Channel to assign value    | RIO-47xx2 and RIO-473xx |
|          | A   | B         | N/A     | Channel    | Channel to assign value    | RIO-47xx0               |
| <b>n</b> | 0   | 1,023.875 | 6       | 1/8        | Value of proportional term |                         |

### Remarks

- n=? will return the currently set value of KP
- RIO-47xx2 and RIO-473xx have 6 control loops (KP n,n,n,n,n,n). RIO-47xx0 has 2 control loops (KP n,n)
- For further details see the Process Control section in the RIO user manual.

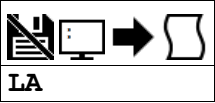
### Examples

```
'Galil DMC Code Example
:KP 12,14;' Implicit notation to set channel A and B term
:KPB= 14;' Explicit notation to set channel A only
:KP ,8;' Implicit notation to set B
:KP ?,?;' Return A,B values
7, 14
:KPB= ?;' Return B value
14
:MG _KPA;' Message the operand for the A channel
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
```

**KP applies to DMC500x0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2,CLS,EDD3701x**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**LA** *List Arrays*

LA

|              |    |                            |
|--------------|----|----------------------------|
| <b>Usage</b> | 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**

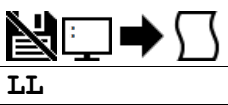
- The listing will be in alphabetical order.

**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 brackets
:LA;                                     Commands the controller to list arrays in alphabetical order
gold[100]
plat[200]
silver[50]
:DA *[];'                               Deallocates all arrays
:LA;'                                   List arrays now returns with no arrays
```

**LA applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@gallimc.com](mailto:documentation@gallimc.com)

**LL** *List Labels*

LL

|              |    |                            |
|--------------|----|----------------------------|
| <b>Usage</b> | 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**

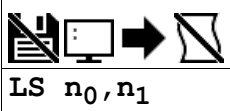
- The LL command label listing will be in alphabetical order.
- The LL command returns all of the program labels in memory and their associated line numbers

**Examples**

```
'Galil DMC Code Example
:LL
#FIVE=5
#FOUR=4
#ONE=1
#THREE=3
#TWO=2
```

**LL applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@gallmc.com](mailto:documentation@gallmc.com)

**LS** *List*LS  $n_0, n_1$ 

|              |          |                                                             |
|--------------|----------|-------------------------------------------------------------|
| <b>Usage</b> | 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                       |
|----------|-----|-----|---------|------------|--------------------------------------------------------------------|-----------------------------|
| $n_0$    | 0   | 198 | 0       | 1          | Specifies the line in the program for which the listing will start | For RIO-47xx0               |
|          | 0   | 398 | 0       | 1          | Specifies the line in the program for which the listing will start | For RIO-47xx2 and RIO-473xx |
| $n_1$    | 1   | 199 | 199     | 1          | Specifies the line at which the listing will end                   | For RIO-47xx0               |
|          | 1   | 399 | 399     | 1          | Specifies the line at which the listing will end                   | For RIO-47xx2 and RIO-473xx |

**Remarks**

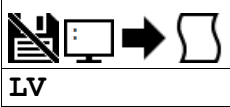
- $n_0 < n_1$  must always be true
- If  $n_0$  or  $n_1$  is omitted, default values are used
- $n_0$  and  $n_1$  can also specify a label, for example:
  - "LS #label,20" would print out program lines from #label to line 20.
- Issuing this command will pause the output of the Data Record until the command is completed.

**Examples**

```
'Galil DMC Code Example
:LS #a,6;           'List program starting at #A through line 6
002 #a
003 MG "Program A"
004 COUNT=1
005 SB 2
006 WT 2000
```

**LS applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**LV** *List Variables*

LV

|              |    |                            |
|--------------|----|----------------------------|
| <b>Usage</b> | 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**

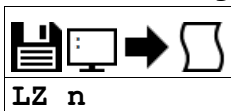
- Use the \_UL operand for total number of variables available for your controller.
  - See the UL command for more details.
- Issuing this command will pause the output of the Data Record until the command is completed.

**Examples**

```
'Galil DMC Code Example
:LV
apple = 60.0000
banana = 25.0000
zebra = 37.0000
```

**LV applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**LZ** *Omit leading zeros*

LZ n

|                 |          |                                                             |
|-----------------|----------|-------------------------------------------------------------|
| <b>Usage</b>    | LZ n ... | Arguments specified with an implicit, comma-separated order |
| <b>Operands</b> | _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
:TB;      'Tell status bits
001
:LZ 1;      'Inhibit leading zeros
:TB;      'Tell status
```

**LZ applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)



**MA** *Email Server IP Address***MA** *n<sub>0</sub>, n<sub>1</sub>, n<sub>2</sub>, n<sub>3</sub>*

|                 |          |                                                             |
|-----------------|----------|-------------------------------------------------------------|
| <b>Usage</b>    | MA n ... | Arguments specified with an implicit, comma-separated order |
| <b>Operands</b> | _MA      | Operand holds the value last set by the command             |

**Description**

The MA command sets the SMTP e-mail server IP address.

**Arguments**

| Argument             | Min | Max | Default | Resolution | Description                    | Notes |
|----------------------|-----|-----|---------|------------|--------------------------------|-------|
| <b>n<sub>0</sub></b> | 0   | 255 | 0       | 1          | First field of the IP address  |       |
| <b>n<sub>1</sub></b> | 0   | 255 | 0       | 1          | Second field of the IP address |       |
| <b>n<sub>2</sub></b> | 0   | 255 | 0       | 1          | Third field of the IP address  |       |
| <b>n<sub>3</sub></b> | 0   | 255 | 0       | 1          | Last field of the IP address   |       |

**Remarks**

- \_MA reports the IP address assigned to MA
- MA ? returns the setting for the MA command in 4 byte notation.

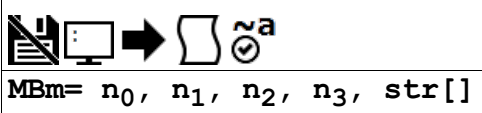
**Examples**

```
'Galil DMC Code Example
MA 10,0,62,10;'Set the SMTP server IP
MA ?;'Interrogates the MA address
 10, 0, 62, 10
MD someone@domain.com
'sets the e-mail recipient
MS RIO@domain.com
'Specifies it's from the RIO
MG {M}"Volts",@AN[1]
'E-mails user the message "Volts"
' following the analog input 1 value
```

**MA applies to RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**MB Modbus**



|              |        |                                                                   |
|--------------|--------|-------------------------------------------------------------------|
| <b>Usage</b> | MBm= n | Arguments specified with a single axis mask and an assignment (=) |
|--------------|--------|-------------------------------------------------------------------|

**Description**

The MB command is used to communicate with I/O devices using the Modbus TCP/IP protocol. The MB command supports the first two levels of Modbus commands. The function code -1 designates that the first level of Modbus is used (creates raw packets and receives raw data). The other codes are the 10 major function codes of the second level. The format of the command varies depending on each function code.

Galil Modbus supports one master per slave.

**Arguments**

*Level 2 Modbus Function Codes*

| Function Code, n <sub>1</sub> | Modbus Definition                          | Slaved Galil Description (RIO only) |
|-------------------------------|--------------------------------------------|-------------------------------------|
| <b>01</b>                     | Read Coil Status (Read Bits)               | Read Digital Outputs (RIO only)     |
| <b>02</b>                     | Read Input Status (Read Bits)              | Read Digital Inputs (RIO only)      |
| <b>03</b>                     | Read Holding Registers (Read Words)        | Read Analog Inputs (RIO only)       |
| <b>04</b>                     | Read Input Registers (Read Words)          | Read Analog Outputs (RIO only)      |
| <b>05</b>                     | Force Single Coil (Write One Bit)          | Write Digital Output (RIO only)     |
| <b>06</b>                     | Preset Single Register (Write One Word)    | Write Digital Outputs (RIO only)    |
| <b>07</b>                     | Read Exception Status (Read Error Code)    | Read Digital Outputs (RIO only)     |
| <b>15</b>                     | Force Multiple Coils (Write Multiple Bits) | Write Digital Outputs (RIO only)    |
| <b>16</b>                     | Preset Multiple Registers (Write Words)    | Write Analog Outputs (RIO only)     |
| <b>17</b>                     | Report Slave ID                            |                                     |

**01: MBm= n<sub>0</sub>, 1, n<sub>2</sub>, n<sub>3</sub>, str[]**

*Read Coil Status (Read Bits)*

| Argument             | Min    | Max     | Default   | Resolution | Description                   | Notes                                     |
|----------------------|--------|---------|-----------|------------|-------------------------------|-------------------------------------------|
| <b>m</b>             | A      | C       | N/A       | Handle     | Handle to send Modbus command | For RIO-47xx0.                            |
|                      | A      | E       | N/A       | Handle     | Handle to send Modbus command | For RIO-47xx2 and RIO-473xx               |
| <b>n<sub>0</sub></b> | 0      | 255     | see Notes | 1          | Unit ID                       | Default to Handle number (A=1, B=2, etc.) |
| <b>n<sub>2</sub></b> | 0      | 9,999   | N/A       | 1          | Address of first coil         |                                           |
| <b>n<sub>3</sub></b> | 0      | 99      | N/A       | 1          | Quantity of coils             | Or, number of IO points to read           |
| <b>str</b>           | 1 char | 8 chars | N/A       | String     | Name of array to store values | str[0] holds the first value.             |

```
'Galil DMC Code Example
MBC=,1,2,8,example[];' Read inputs 2-9 from handle C, save to example[]
'equivalent to reading Digital Outputs or registers mapped to 100xxx
```

**02: MBm= n<sub>0</sub>, 2, n<sub>2</sub>, n<sub>3</sub>, str[]**

*Read Input Status (Read Bits)*

| Argument             | Min    | Max     | Default | Resolution | Description                   | Notes                                     |
|----------------------|--------|---------|---------|------------|-------------------------------|-------------------------------------------|
| <b>m</b>             | A      | C       | N/A     | Handle     | Handle to send Modbus command | For RIO-47xx0.                            |
|                      | A      | E       | N/A     | Handle     | Handle to send Modbus command | For RIO-47xx2 and RIO-473xx               |
| <b>n<sub>0</sub></b> | 0      | 255     | 1       | see Notes  | Unit ID                       | Default to Handle number (A=1, B=2, etc.) |
| <b>n<sub>2</sub></b> | 0      | 9,999   | N/A     | 1          | Address of first input        |                                           |
| <b>n<sub>3</sub></b> | 0      | 99      | N/A     | 1          | Quantity of inputs            | Or, number of IO points to read           |
| <b>str</b>           | 1 char | 8 chars | N/A     | String     | Name of array to store values | str[0] holds the first value.             |

```
'Galil DMC Code Example
MBC=,2,4,3,example[];' Read inputs 4,5 and 6 from handle C, save to example[]
'equivalent to reading Digital Inputs or registers mapped to 000xxx
```

**03: MBm= n<sub>0</sub>, 3, n<sub>2</sub>, n<sub>3</sub>, str[]**

*Read Holding Registers (Read Words)*

| Argument             | Min | Max   | Default   | Resolution | Description                   | Notes                                     |
|----------------------|-----|-------|-----------|------------|-------------------------------|-------------------------------------------|
| <b>m</b>             | A   | C     | N/A       | Handle     | Handle to send Modbus command | For RIO-47xx0                             |
|                      | A   | E     | N/A       | Handle     | Handle to send Modbus command | For RIO-47xx2 and RIO-473xx               |
| <b>n<sub>0</sub></b> | 0   | 255   | see Notes | 1          | Unit ID                       | Default to Handle number (A=1, B=2, etc.) |
| <b>n<sub>2</sub></b> | 0   | 9,999 | N/A       | 1          | Address of first register     |                                           |

|            |        |         |     |        |                               |                                                                                               |
|------------|--------|---------|-----|--------|-------------------------------|-----------------------------------------------------------------------------------------------|
| <b>n3</b>  | 0      | 99      | N/A | 1      | Quantity of registers to read |                                                                                               |
| <b>str</b> | 1 char | 8 chars | N/A | String | Name of array to store values | str[0] holds the first value. 2 bytes per element. Array must be as large as the value for n3 |

```
'Galil DMC Code Example
MBB=,3,1,4,example[];' Read registers 1 through 4 from handle B, save to example[]
'equivalent to reading Analog Outputs, or registers mapped to 400xxx
```

#### 04: MBm= n0, 4, n2, n3, str[]

Read Input Registers (Read Words)

| Argument   | Min    | Max     | Default   | Resolution | Description                   | Notes                                                                                         |
|------------|--------|---------|-----------|------------|-------------------------------|-----------------------------------------------------------------------------------------------|
| <b>m</b>   | A      | C       | N/A       | Handle     | Handle to send Modbus command | For RIO-47xx0                                                                                 |
|            | A      | E       | N/A       | Handle     | Handle to send Modbus command | For RIO-47xx2 and RIO-473xx                                                                   |
| <b>n0</b>  | 0      | 255     | see Notes | 1          | Unit ID                       | Default to Handle number (A=1, B=2, etc.)                                                     |
| <b>n2</b>  | 0      | 9,999   | N/A       | 1          | Address of first register     |                                                                                               |
| <b>n3</b>  | 1      | 99      | N/A       | 1          | Quantity of registers to read |                                                                                               |
| <b>str</b> | 1 char | 8 chars | N/A       | String     | Name of array to store values | str[0] holds the first value. 2 bytes per element. Array must be as large as the value for n3 |

```
'Galil DMC Code Example
MBB=,4,1,2,example[];' Read registers 1 through 2 from handle B, save to example[]
'equivalent to reading Analog Inputs, or registers mapped to 300xxx
```

#### 05: MBm= n0, 5, n2, n3

Force Single Coil (Write One Bit)

| Argument  | Min | Max   | Default   | Resolution | Description                   | Notes                                     |
|-----------|-----|-------|-----------|------------|-------------------------------|-------------------------------------------|
| <b>m</b>  | A   | C     | N/A       | Handle     | Handle to send Modbus command | For RIO-47xx0                             |
|           | A   | E     | N/A       | Handle     | Handle to send Modbus command | For RIO-47xx2 and RIO-473xx               |
| <b>n0</b> | 0   | 255   | see Notes | 1          | Unit ID                       | Default to Handle number (A=1, B=2, etc.) |
| <b>n2</b> | 0   | 9,999 | N/A       | 1          | Address of coil               |                                           |
| <b>n3</b> | 0   | 1     | 0         | 1          | Set coil status               | 0 = turn off coil. 1 = turn on coil       |

```
'Galil DMC Code Example
MBB=,5,11,1;' Set coil 11 high
'equivalent to setting a Digital Output (SB/CB)
```

#### 06: MBm= n0, 6, n2, n3

Preset Single Register (Write One Word)

| Argument  | Min | Max        | Default   | Resolution | Description                   | Notes                                     |
|-----------|-----|------------|-----------|------------|-------------------------------|-------------------------------------------|
| <b>m</b>  | A   | C          | N/A       | Handle     | Handle to send Modbus command | For RIO-47xx0                             |
|           | A   | E          | N/A       | Handle     | Handle to send Modbus command | For RIO-47xx2 and RIO-473xx               |
| <b>n0</b> | 0   | 255        | see Notes | 1          | Unit ID                       | Default to Handle number (A=1, B=2, etc.) |
| <b>n2</b> | 0   | 9,999      | N/A       | 1          | Address of holding register   |                                           |
| <b>n3</b> | 0   | 65,535     | 0         | 1          | Set register value            | RIO-471xx and RIO-472xx                   |
|           | 0   | 16,777,215 | 0         | 1          | Set register value            | RIO-473xx                                 |

```
'Galil DMC Code Example
MBC=,6,10,128;' write 128 to holding register 10 on handle C
'equivalent to setting digital outputs on the RIO, or setting registers addressed 400xxx
```

#### 07: MBm= n0, 7, str[]

Read Exception Status (Read Error Code)

| Argument   | Min    | Max     | Default   | Resolution | Description                   | Notes                                           |
|------------|--------|---------|-----------|------------|-------------------------------|-------------------------------------------------|
| <b>m</b>   | A      | C       | N/A       | Handle     | Handle to send Modbus command | For RIO-47xx0                                   |
|            | A      | E       | N/A       | Handle     | Handle to send Modbus command | For RIO-47xx2 and RIO-473xx                     |
| <b>n0</b>  | 0      | 255     | see Notes | 1          | Unit ID                       | Default to Handle number (A=1, B=2, etc.)       |
| <b>str</b> | 1 char | 8 chars | N/A       | String     | Name of array to store value  | str[0] holds the received value, one byte only. |

- When using function code 7 with a Galil slave, array element zero will be set to the byte value of the combined first 8 digital outputs.
- Only one byte in the array will be populated, element zero of array str[].

```
'Galil DMC Code Example
MBE=,7,example[];' Read register and store in example[0]
```

**15: MBm= n0, 15, n2, n3, str[]***Force Multiple Coils (Write Multiple Bits)*

| Argument   | Min    | Max     | Default   | Resolution | Description                   | Notes                                             |
|------------|--------|---------|-----------|------------|-------------------------------|---------------------------------------------------|
| <b>m</b>   | A      | C       | N/A       | Handle     | Handle to send Modbus command | For RIO-47xx0                                     |
|            | A      | E       | N/A       | Handle     | Handle to send Modbus command | For RIO-47xx2 and RIO-473xx                       |
| <b>n0</b>  | 0      | 255     | see Notes | 1          | Unit ID                       | Default to Handle number (A=1, B=2, etc.)         |
| <b>n2</b>  | 0      | 9,999   | N/A       | 1          | Address of first coil         |                                                   |
| <b>n3</b>  | 1      | 16      | N/A       | 1          | Quantity of coils             | RIO-471xx and RIO-472xx                           |
|            | 1      | 24      | N/A       | 1          | Quantity of coils             | RIO-473xx                                         |
| <b>str</b> | 1 char | 8 chars | N/A       | String     | Array to set values for coils | str[0] holds the first value. 16 bits per element |

```
'Galil DMC Code Example
example[0]=255;'
MBC=,15,0,16,example[];' Set 1st byte of coils high and 2nd byte of coils low
'equivalent to setting digital outputs on RIO, or setting coils addressed 000xxx
```

**16: MBm= n0, 16, n2, n3, str[]***Preset Multiple Registers (Write Words)*

| Argument   | Min    | Max     | Default   | Resolution | Description                   | Notes                                                                      |
|------------|--------|---------|-----------|------------|-------------------------------|----------------------------------------------------------------------------|
| <b>m</b>   | A      | C       | N/A       | Handle     | Handle to send Modbus command | For RIO-47xx0                                                              |
|            | A      | E       | N/A       | Handle     | Handle to send Modbus command | For RIO-47xx2 and RIO-473xx                                                |
| <b>n0</b>  | 0      | 255     | see Notes | 1          | Unit ID                       | Default to Handle number (A=1, B=2, etc.)                                  |
| <b>n2</b>  | 0      | 9,999   | N/A       | 1          | Address of first register     |                                                                            |
| <b>n3</b>  | 0      | 99      | N/A       | 1          | Quantity of registers         |                                                                            |
| <b>str</b> | 1 char | 8 chars | N/A       | String     | Array containing modbus data  | str[0] holds the first value. 2 bytes per element. Array size must be > n3 |

```
'Galil DMC Code Example
example[0]=$AEAE
MBD=,16,2,1,example[];' Set $AEAE to holding register 2 on handle D
'equivalent to setting analog outputs, or writing to holding registers addressed 400xxx
```

**Raw Modbus Packet Send****MBm= n0,-1,n2,str[]***Raw Modbus Send*

| Argument   | Min    | Max     | Default | Resolution | Description                            | Notes                                     |
|------------|--------|---------|---------|------------|----------------------------------------|-------------------------------------------|
| <b>m</b>   | A      | C       | N/A     | Handle     | Handle to send Modbus command          | For RIO-47xx0                             |
|            | A      | E       | N/A     | Handle     | Handle to send Modbus command          | For RIO-47xx2 and RIO-473xx               |
| <b>n0</b>  | 0      | 255     | 1       | see Notes  | Unit ID                                | Default to Handle number (A=1, B=2, etc.) |
| <b>n2</b>  | 0      | 999     | N/A     | 1          | Number of array bytes to send          |                                           |
| <b>str</b> | 1 char | 8 chars | N/A     | String     | Name of array containing outgoing data | Array size >= n2. See Remarks             |

**Raw Modbus Packet Send/Receive****MBm= n0,-1,n2,str0[],n3,n4,str1[]****MB applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx**©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**MD** *Email Destination Address*

MD str

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

**Description**

The MD command sets the e-mail destination address.

**Arguments**

| Argument | Value  | Description                | Notes                 |
|----------|--------|----------------------------|-----------------------|
| str      | String | E-mail address destination | Maximum 30 characters |

**Remarks**

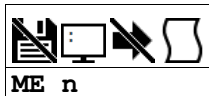
- Do not place quotation marks around email address
- Do not use semi-colon to build compound commands on the same line

**Examples**

```
'Galil DMC Code Example
MA 10,10,10,1; 'SMTP server address
MD example@domain.com
'Destination address of email is:
' "example@test.com"
MS RIO@domain.com
'Sender e-mail address
MG {M}"variable status",var;
'Sends the status of variable
' var to example@domain.com
```

**MD applies to RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**ME Modbus array write enable**

ME n

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

**Description**

The ME command enables the ability for Modbus masters to write to array locations in the hardware's array table. When enabled, array locations can be written to as 16 bit integers or as 32 bit floating point by a modbus master by specifying different address ranges. ME is not required to read array locations, reads are always supported.

**Arguments**

| Argument | Value | Description                                                         | Notes   |
|----------|-------|---------------------------------------------------------------------|---------|
| n        | 0     | Disables the ability for Modbus masters to write to the array table | Default |
|          | 1     | Enables ability for Modbus masters to write to the array table      |         |

**Remarks**

- The function code used to read arrays can be modified by MV. Gall recommends MV-1 when using array read/write functionality. This makes function code 3 the default read command.
- Array writes when enabled by ME are done using function code 16
- Gall Modbus supports one master per slave.
- 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>

**Modbus Register Map**

- Each element is accessible as a 16 bit unsigned integer (Modbus registers 1xxx) -OR- as a 32 bit floating point number (Modbus registers 2xxx).
- The table below shows the mapping for a Modbus master writing to the controller with ME 1 set.
- 1000 elements are available in the RIO-471x2/47300 array table and 400 in the RIO-471x0/47200.

*Modbus Register Map to Gall Array A[]*

| Modbus Registers:                                 | 1000-1xxx                          | 2000-2xxx                                        |
|---------------------------------------------------|------------------------------------|--------------------------------------------------|
| Register range for RIO-47xx2 and RIO-47300        | 1000-1999 (xxx=999)                | 2000-2999 (xxx=999)                              |
| Register range for RIO-471x0 and 47200            | 1000-1399 (xxx=399)                | 2000-2399 (xxx=399)                              |
| Available Modbus function codes                   | 3 (read) and 16 (write)            | 3 (read) and 16 (write)                          |
| Number Type                                       | 16 bit unsigned integer            | 32 bit floating point                            |
| References in A[] array (RIO-47xx2 and RIO-47300) | A[0]-A[999]                        | A[0]-A[999]                                      |
| References in A[] array (RIO-471x0 RIO-47200)     | A[0]-A[399]                        | A[0]-A[399]                                      |
| Number written to A[]                             | Integer only, fraction not changed | Gall 4.2 format (internal from float conversion) |
| Number read from A[]                              | Integer only, fraction not read    | 32 bit float (internal to float conversion)      |
| Example Modbus Master Write                       | MBH=0,16,1000,1,write[]            | MBH=0,16,2001,2,write[]                          |
| Example Modbus Master Read                        | MBH=0,3,1000,1,read[]              | MBH=0,3,2001,2,read[]                            |

**Embedded Array Mapping**

- Once enabled, the entire array table can be written remotely. These writes can span across dimensioned user arrays. It is the user's responsibility to partition the array table and to read/write remotely to the correct location.
- When using multiple array names, the array table is partitioned alphabetically (all capital letters first).
  - For example, a partitioned array of Grape[600] and Orange[200] would place the first 600 registers in Grape[], and the next 200 registers in Orange[]. The last 200 elements would be inaccessible from embedded code. If the user then dimensioned the array Apple[200], the register mapping would change. The first 200 registers would read/write from Apple[], the next 600 from Grape[], and finally the last 200 from Orange[].
  - Additionally, all capital letters come before lowercase letters. For example, a partitioned array of Banana[100] and apple[200] would place the first 100 registers in Banana[] and the next 200 registers in apple[].
- For simplicity, Gall recommends that a single, contiguous array be dimensioned with the array name "A".

**Examples**

```
'Galil DMC Code Example
:DA *[];'      Deallocates all arrays
:DM A[400];'   Allocates array for Modbus Read/Write
:ME0;'       Disables write access
:ME1;'       Enables write access
:ME?;'       Interrogate current value
1
```

```
'Galil DMC Code Example
'This example is written for a Galil modbus master to a DMC-40x0, DMC-41x3, DMC-42x0, DMC-3x01x, or RIO-47xxx (with expanded memory)
'Master is E.G. DMC-21x3, RIO, DMC-40x0
'This code runs on the master.
'Assumes a Modbus handle is available at H,
' and that ME1 has been set on the remote device

MW1;'        Turn on modbus wait
DM write[2];' Dimension an array for holding data to transmit

write[0]=1234;' Assign an integer to element 0
MBH=0,16,1000,1,write[];' Send the integer to register 1000 on the remote

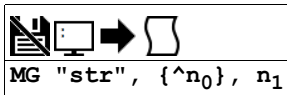
write[0]=$42F6;' Set the 32 bit float in two steps, the value is 123.456
write[1]=$E978
MBH=0,16,2001,2,write[];' Send the float to register 2001 on the remote
'note that register 2000 would have stepped on the integer memory written at 1000
```

|                                                  |                                                        |
|--------------------------------------------------|--------------------------------------------------------|
| <code>DM read[2];'</code>                        | Dimension an array for holding read data               |
| <code>MBH=0,3,1000,1,read[];'</code>             | Read the integer at register 1000                      |
| <code>MG"Integer=",read[0];'</code>              | Print the read integer                                 |
| <code>MBH=0,3,2001,2,read[];'</code>             | Read the float at register 1000                        |
| <code>float=(read[0]*\$10000) + read[1];'</code> | Construct the float. Shifting necessary for high bytes |
| <code>MG"Float=", float{\$8.0};'</code>          | Print the float in hex                                 |

**ME applies to DMC40x0,DMC42x0,DMC41x3,DMC30010,RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**MG Message**



MG "str", {^n<sub>0</sub>}, n<sub>1</sub>

**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 <sub>0</sub> | ASCII character in decimal | Allows users to print ASCII characters                     | Range of 0-255                                  |
| n <sub>1</sub> | Numeric value              | Prints the numeric value specified                         | See Examples for valid uses of n <sub>1</sub> . |
|                | 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            |                                                 |

| Argument | Value   | Description                                    | Notes |
|----------|---------|------------------------------------------------|-------|
| n        | Operand | Prints the numeric value stored by the operand |       |

**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.
  - Unsolicited messages can go to any of the Ethernet handles or serial ports.
  - The CF command sets the default communication port for routing 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.

**Message Routing**

MG can override the default CF setting by using the following modifiers at the beginning of the message, right after MG.

- {Pn} Sends the message out the Serial port n, where n is 1 or 2 denoting Main or Auxiliary (where equipped).
- {Ex} Sends the message out the Ethernet handle x, where x is A,B,C,D,E or F (RIO-47xx2 and RIO-473xx)
- {Cx} Sends the message out the Ethernet handle x, where x is A,B, or C (RIO-47xx1)
- {M} Sends message to the e-mail specified by the MD command.
- See the MD, MS, and MA command for more information.

**Examples**

**Valid uses of n<sub>1</sub> 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 {A13}, {A10}, {A48}, {A055}; 'Specifies carriage return, line feed, and the characters 0 and 7 in ASCII decimal values
```

07



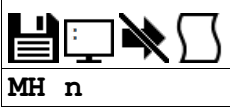
```
:MG TIME;           'Messages the operand TIME
261928200.0000
:variable= 10;      'Sets the variable equal to 10
:MG variable+5;     'Messages out variable + 5
15.0000
:MG _TI0;           'Messages the value stored in the operand _TI0
255.0000
```

```
'Galil DMC Code Example
CF A;               'Messages configured to go out Ethernet handle A
MG {EB}var;        'Override CF and send the value of variable var to B handle
```

**MG applies to**

**DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2,RIO574x0,EDD3701x**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**MH** *Modbus Heartbeat*

MH n

|                 |          |                                                             |
|-----------------|----------|-------------------------------------------------------------|
| <b>Usage</b>    | MH n ... | Arguments specified with an implicit, comma-separated order |
| <b>Operands</b> | _MH      | Operand holds the value last set by the command             |

**Description**

The MH command is used to setup the modbus timeout feature of the RIO when configured as a Modbus TCP slave.

**Arguments**

| Argument | Min | Max    | Default | Resolution | Description                                                                            | Notes                          |
|----------|-----|--------|---------|------------|----------------------------------------------------------------------------------------|--------------------------------|
| n        | 0   | 32,767 | 0       | 1          | The time in milliseconds for a modbus command to be received before #MODERR is called. | 0 disables the timeout feature |

**Remarks**

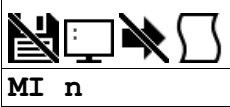
- \_MH contains the current value of MH
- Any commands received over the modbus port (port 502) will reset the time for a timeout to occur.
- Changing the value of the MH command resets the timer to 0 to generate a modbus timeout.

**Examples**

```
'Galil DMC Code Example
#AUTO;           'Auto Start
MH 100;          'Set timeout at 100msec
#loop;           'Main loop
JP #loop;        'Repeat loop
EN
'
#MODERR;         'Modbus Timeout Automatic subroutine occurred
IHA =>-2;        'Close ethernet handle A
RE;              'Return from subroutine
```

**MH applies to RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**MI Modbus Integer**

MI n

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

**Description**

The MI command configures how a RIO slave will respond to Analog I/O requests. MI specifies whether the RIO will respond to analog Modbus requests in 32-Bit Floating Point notation or as a 16-Bit Integer.

**Arguments**

| Argument | Value | Description                                                    | Notes                       |
|----------|-------|----------------------------------------------------------------|-----------------------------|
| n        | 0     | Analog I/O responses will be in 32-bit floating point notation | Default. Units in Volts [V] |
|          | 1     | Analog I/O responses will return as 16-bit bit integer         | Units in counts.            |

**Remarks**

- n = ? returns the state of the MI command, either a 0 or 1
- This command is of particular use for software packages that require data to be in integer format.
- The MI command affects the way the RIO responds to function codes 3, 4 and 16.
- When MI is set to 1 the range of the 16-bit integer depends on the AQ and DQ setting.

**Examples**

```
'Galil DMC Code Example
MI 1;' Set slave to respond to Analog I/O related Modbus requests in counts
```

```
'Galil DMC Code Example
'From RIO (modbus slave)
:MI0;' output analog IO data as 32bit float
:DQ1,2;' Set A01 to 0-10V
:A01,9.9;' Set A01 to 9.9V
:MG@AO[1];'output analog output 1
9.9001

'From controller (modbus master)
:IHF=192,168,1,2<502;' RIO IP, modbus port 502
:DM read[10];' Setup array to read modbus data
:MBF=,4,2,2,read[];' Read analog out 1 (2 regs x 16 bits= 32 bit float)
:float=read[0]*$8000;' Construct float, shift high byte
:float=float+float+read[1]
:MG@REAL[float];' Convert float and read
9.9001 Correct analog out for 32 bit float

'From RIO
:MI1;' Output analog IO data as 16 bit integer

'From controller
:MBF=,4,1,1,read[];' Read analog out 1 (1 reg x 16bits = 16 bit int)
:MGread[0]
64880 Correct for 9.9V read as 16 bit int (0-10V range)
```

**MI applies to RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**MS** *Email Source Address*

|              |          |                                                             |
|--------------|----------|-------------------------------------------------------------|
| <b>Usage</b> | MS n ... | Arguments specified with an implicit, comma-separated order |
|--------------|----------|-------------------------------------------------------------|

**Description**

The MS command sets the e-mail source address.

**Arguments**

| Argument   | Value  | Description           | Notes                                                              |
|------------|--------|-----------------------|--------------------------------------------------------------------|
| <b>str</b> | String | Source e-mail address | Does not need to be a valid e-mail address. Maximum 30 characters. |

**Remarks**

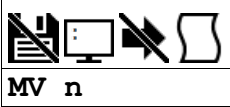
- Do not place quotation marks around email address
- Do not use semi-colon to build compound commands on the same line

**Examples**

```
'Galil DMC Code Example
MA 10,0,0,1; 'Example STMP server IP
MD someone@example.com
'Destination e-mail address
MS rio@example.com
'RIO e-mail address
MG {M}"Testing Email"
'Message to send via Email
EN
```

**MS applies to RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**MV Modbus Reversal**

MV n

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

**Description**

Enabling the MV command causes the firmware to change the way it responds to function codes 3 and 4. Some modbus software packages may require function codes 3 and 4 to be switched.

**Arguments**

| n  | Analog Inputs   | Analog Outputs  | Array Read Access (See ME) | Notes   |
|----|-----------------|-----------------|----------------------------|---------|
| 0  | Function Code 3 | Function Code 4 | Function Code 3            | Default |
| 1  | Function Code 4 | Function Code 3 | Function Code 4            |         |
| -1 | Function Code 4 | Function Code 3 | Function Code 3            |         |

**Remarks**

- Use of the Galil commands @AN[], @AO[] require MV to be set to 0.

**Examples**

```
'Galil DMC Code Example
MV 1;      'Swap function code 3 and 4 from defaults, e.g. for Array Reads
```

```
'Galil DMC Code Example
'From RIO (modbus slave)
:MV0;' Set analog outs read by FC4, analog ins read by FC3
:AO0,1.2;' Set AO[0] to 1.2V
:MG@AN[0];' Read AN[0] as 2.5V
2.5891

'From controller (modbus master)
:DM read[10];' Setup read array for MB command
:IHF=192,168,1,2<502;' RIO IP
:MBF=,3,0,2,read[];' Get analog ins
:float=read[0]*$8000
:float=float+float+read[1];' Reconstruct 32bit float from 2 words in read array
:MG@REAL[float];' Convert float to 4.2 and report value
2.5891 Confirm analog input is read

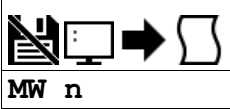
'From RIO
:MV1;' Set analog outs read by FC3, analog ins read by FC4

'From controller
:MBF=,3,0,2,read[];' Same FC gets analog outs now due to MV1
:float=read[0]*$8000
:float=float+float+read[1];' Reconstruct 32bit float
:MG@REAL[float];' Convert float to 4.2 and report value
1.2001 Confirm analog out is read
```

**MV applies to DMC30010,RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@gallimc.com](mailto:documentation@gallimc.com)

**MW Modbus Wait**



|                 |              |                                                             |
|-----------------|--------------|-------------------------------------------------------------|
| <b>Usage</b>    | MW n ...     | Arguments specified with an implicit, comma-separated order |
| <b>Operands</b> | _MW0<br>_MW1 | Operand has special meaning, see Remarks                    |

**Description**

Enabling the MW command causes the controller to hold up execution of the program after sending a Modbus command until a response from the Modbus device has been received. The MW command ensures that the command that was sent to the Modbus device was successfully received before continuing program execution.

**Arguments**

| Argument | Value | Description          | Notes   |
|----------|-------|----------------------|---------|
| n        | 0     | Disables Modbus wait |         |
|          | 1     | Enables Modbus wait  | Default |

**Remarks**

- n = ? returns the state of the Modbus wait, either 1 or 0
- If a Modbus response is never received, then thread 0 will jump to the #TCPERR subroutine, if it exists, and TC will report an error code of 123.
- MW prevents the controller from sending multiple commands to the same Modbus device before it has a chance to execute them.
- Operands
  - \_MW0 returns last function code received
  - \_MW1 returns Modbus error code

**MWn operands**

```
'Galil DMC Code Example
:MG_MW0{&8.0};' $ is the hex formatter
$00000001
:'above is an expected response to function code 1
:MG_MW1{&8.0}
$00000000
:'no error
```

*MW0 Responses*

| Function Code Sent | Normal _MW0 Response | _MW0 Exception Response |
|--------------------|----------------------|-------------------------|
| 1                  | \$01                 | \$81                    |
| 2                  | \$02                 | \$82                    |
| 3                  | \$03                 | \$83                    |
| 4                  | \$04                 | \$84                    |
| 5                  | \$05                 | \$85                    |
| 6                  | \$06                 | \$86                    |
| 7                  | \$07                 | \$87                    |
| 15                 | \$0F                 | \$8F                    |
| 16                 | \$10                 | \$90                    |

*MW1 Responses*

| _MW1 returns | Exception description                           |
|--------------|-------------------------------------------------|
| \$00         | Normal response                                 |
| \$01         | The request referenced an illegal function code |
| \$02         | The request referenced an illegal data address  |

**Examples**

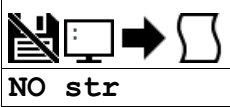
```
'Galil DMC Code Example
MW1; 'Enables Modbus Wait
SB1001; 'Set Bit 1 on Modbus Handle A
CB1001; 'Clear Bit 1 on Modbus Handle A
```

```
'Galil DMC Code Example
REM Example on Modbus master, DMC-40x0
REM Using _MW operands
:IHH=192,168,42,43<502>2;' connect to RIO
:MW1
:SB 8001;' set bit one on RIO
::MBH=,5,1,0;' clear it with MB
::'CB 8001 would also work
:MG_MW0
5.0000
:'funct code 5 confirmed
:MG_MW1
0.0000
```

```
: 'no errors  
: MBH=,5,100,1;' invalid output point  
: :TC1  
0  
: MG_MW0{8.0}  
$00000085  
: 'Exception on funct code 5  
: MG_MW1{8.0}  
$00000002  
: 'illegal data address
```

**MW applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@gallmc.com](mailto:documentation@gallmc.com)

**NO** *No Operation*

|                 |     |                                          |
|-----------------|-----|------------------------------------------|
| <b>Operands</b> | _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                                                                                    |
|------------|--------|-------------------------------------------------|------------------------------------------------------------------------------------------|
| <b>str</b> | 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**

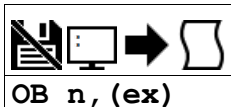
```
'Galil DMC Code Example
#a;           'Program A
NO;          'No Operation
NO This Program ; 'No Operation
NO Does Absolutely; 'No Operation
NO Nothing;   'No Operation
EN;          'End of Program
```

**NO applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)



## OB Output Bit



| 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        | 0     | 15    | 0       | 1          | Output bit specified                     | RIO-41xxx and RIO-42xxx                                                       |
|          | 0     | 23    | 0       | 1          | Output bit specified                     | RIO-473xx                                                                     |
|          | 24    | 47    | 0       | 1          | Output bit specified                     | RIO-473xx with -24ExOut option                                                |
| n        | 1,000 | 3,999 | N/A     | 1          | Modbus output bit specified              | For RIO-47xx0. See Remarks                                                    |
|          | 1,000 | 5,999 | N/A     | 1          | Modbus output bit specified              | For RIO-47xx2 and RIO-473xx. 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.

#### Using OB with a Modbus Slave

- $n = (\text{SlaveAddress} * 10000) + (\text{HandleNum} * 1000) + ((\text{Module} - 1) * 4) + (\text{Bitnum} - 1)$ 
  - Slave Address is used when the ModBus device has slave devices connected to it and specified as Addresses 0 to 255. Please note that the use of slave devices for modbus are very rare and this number will usually be 0.
  - HandleNum is the handle specifier where A is 1, B is 2 and so on.
  - Module is the position of the module in the rack from 1 to 16.
  - BitNum is the I/O point in the module from 1 to 4

### 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
```

```
'Galil DMC Code Example
'Toggle digital output 1
OB1,@COM[@OUT[1]] & 1;' read current state of output 1, take the bitwise complement, mask out bits.
```

**OB applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,DMC18x6,DMC18x2,RIO47xxx,EDD3701x**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

## OF Offset

|                                                                                  |
|----------------------------------------------------------------------------------|
|  |
| OFm= n                                                                           |
| OF n, n, n, n, n, n, n, n                                                        |

|                 |          |                                                                   |
|-----------------|----------|-------------------------------------------------------------------|
| <b>Usage</b>    | OFm= n   | Arguments specified with a single axis mask and an assignment (=) |
|                 | OF n ... | Arguments specified with an implicit, comma-separated order       |
| <b>Operands</b> | _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                                                |
|----------|---------|--------|---------|------------|------------------------------------------|------------------------------------------------------|
| <b>m</b> | A       | F      | N/A     | Channel    | Channel to assign value                  | Valid for RIO-47xx2 and RIO-473xx                    |
|          | A       | B      | N/A     | Channel    | Channel to assign value                  | Valid for RIO-47xx0                                  |
| <b>n</b> | -2.5    | 2.5    | 0       | 5/65,536   | Offset voltage applied to control output | Valid for RIO-4710x                                  |
|          | -9.9982 | 9.9982 | 0       | 20/65,536  | Offset voltage applied to control output | Valid for RIO-4712x. Range determined by DQ command. |

### Remarks

- The process control loop input and output (AF and AZ) must be set prior to adjusting OF.
- For more information see the "Process Loop" section of the User Manual.

### Examples

```
'Galil DMC Code Example
:AF0,1
:AZ0,1
:OF -2;' Set control channel A to -2 Leave other channel unchanged
:OF 0;' Set control channel B to 0 Leave other channel unchanged
:OF ?,?;' Return offsets
-2.0000,0.0000
:OF ?;' Return A offset
-2.0000
:OF ,?;' Return B offset
0.0000
```

**OF applies to DMC500x0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2,EDD3701x**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

## OP Output Port



OP  $n_0, n_1, n_2, n_3, n_4, n_5$

| Usage    | OP n ...                                     | Arguments specified with an implicit, comma-separated order |
|----------|----------------------------------------------|-------------------------------------------------------------|
| Operands | _OP0<br>_OP1<br>_OP2<br>_OP3<br>_OP4<br>_OP5 | 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   | 255 | 0       | 1          | Decimal representation: Outputs 0-7   |                                |
| $n_1$    | 0   | 255 | 0       | 1          | Decimal representation: Outputs 8-15  |                                |
| $n_2$    | 0   | 255 | 0       | 1          | Decimal representation: Outputs 16-23 | RIO-473xx                      |
| $n_3$    | 0   | 255 | 0       | 1          | Decimal representation: Outputs 24-31 | RIO-473xx with -24ExOut option |
| $n_4$    | 0   | 255 | 0       | 1          | Decimal representation: Outputs 32-39 | RIO-473xx with -24ExOut option |
| $n_5$    | 0   | 255 | 0       | 1          | Decimal representation: Outputs 40-47 | RIO-473xx with -24ExOut option |

### Remarks

- Bit patterns for extended I/O banks (where available) configured as inputs have no affect on the IO status.

### Output Mapping Examples

| Examples      | Command issued (Hex version) | Bits Set | Bits Cleared |
|---------------|------------------------------|----------|--------------|
| Set bank high | OP255 (OP\$FF)               | 0-7      | -            |
| Set bank low  | OP0 (OP\$00)                 | -        | 0-7          |

### Examples

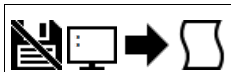
```
'Galil DMC Code Example
OP 0;' Clear Output Port -- all bits
OP $85;' Set outputs 1,3,8 and clear the others
MG _OP0;' Returns the parameter "n0"
```

### OP applies to

DMC500x0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2,RIO574x0,DMC52xx0,EDD3701x

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@gallimc.com](mailto:documentation@gallimc.com)

## P1CD *Serial port 1 code*



P1CD

|                 |                |                                          |
|-----------------|----------------|------------------------------------------|
| <b>Usage</b>    | variable= P1CD | Holds a value                            |
| <b>Operands</b> | P1CD           | Operand has special meaning, see Remarks |

### Description

P1CD returns the status of the serial port when in the operator data entry mode (CI,1). The value of P1CD returns zero after the corresponding string or number is read.

### Arguments

P1CD is an operand that holds a value corresponding to status. See Examples for use in code.

### Remarks

- P1CD contains the following status codes

#### *P1CD Status Codes*

| Status Code | Meaning                                     |
|-------------|---------------------------------------------|
| -1          | Mode disabled                               |
| 0           | Nothing received                            |
| 1           | Received character, but not carriage return |
| 2           | received a string, not a number             |
| 3           | received a number                           |

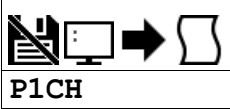
### Examples

```

Galil DMC Code Example
CI,1;'          Interrupt on main serial port
#wait
WT100000
JP#wait;'      dummy loop
'
'
#COMINT;'      Interrupt runs on carriage return
IF (P1CD = 2);' If string received
  MGP1ST{S6}{EA};' Message the string out the Ethernet port
ENDIF
IF (P1CD = 3);' If number received
  OP P1NM;'      Set outputs based on received number
ENDIF
EN
    
```

**P1CD applies to DMC30010,DMC21x3,RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**P1CH** *Serial port 1 character*

P1CH

|                 |                |                                          |
|-----------------|----------------|------------------------------------------|
| <b>Usage</b>    | variable= P1CH | Holds a value                            |
| <b>Operands</b> | P1CH           | Operand has special meaning, see Remarks |

**Description**

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

**Arguments**

P1CH is an operand that holds a value corresponding to ASCII characters sent over the serial port. See Examples for use in code.

**Remarks**

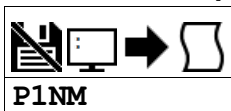
- None

**Examples**

```
'Galil DMC Code Example
:ARAS
:CI,1
:MG "TEST" {P1} ;'send a message to the hand terminal
:MG P1CH {S1} ;'the 6 button was pushed on the hand terminal
6
```

**P1CH applies to DMC30010,DMC21x3,RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**P1NM** *Serial port 1 number*

|                 |                |                                          |
|-----------------|----------------|------------------------------------------|
| <b>Usage</b>    | variable= P1NM | Holds a value                            |
| <b>Operands</b> | P1NM           | Operand has special meaning, see Remarks |

**Description**

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

**Arguments**

P1NM is an operand that holds a numerical value sent over the serial port. See Examples for use in code.

**Remarks**

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

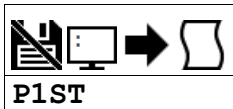
**Examples****Simple example with Ethernet and Serial port on PC**

1. Connect to the controller via Ethernet in Galil software
2. Download and run the following code
3. Now connect via Hyperterm or other terminal software to the RS232 port. "X" will print every second. This string can be changed to whatever command string is required for the third party device. Now type a number and enter.
4. The "Number Received" message will print on the Ethernet connection

```
'Galil DMC Code Example
CI 1,1;' Interrupt on CR
CW 2;' don't set MSB
#loop
'Send a character to the port
MG {P1} "X"
WT 1000
JP #loop
#COMINT
MG "Number Received:",P1NM
EN ,1
```

**P1NM applies to DMC30010,DMC21x3,RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**P1ST** *Serial port 1 string*

P1ST

|                 |                |                                          |
|-----------------|----------------|------------------------------------------|
| <b>Usage</b>    | variable= P1ST | Holds a value                            |
| <b>Operands</b> | P1ST           | Operand has special meaning, see Remarks |

**Description**

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

**Arguments**

P1ST is an operand that contains a string. See Examples for usage.

**Remarks**

- No more than 6 characters can be accessed.
- Strings must be messages out using the {Sx} operator in the MG command. Otherwise the string will be reported as its ASCII numerical value.

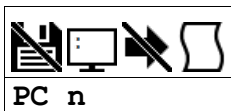
**Examples**

```
'Galil DMC Code Example
:CI ,1
:MG "TEST" {P1} ;'send a message to the hand terminal
:MG P1ST {S3} ;'the characters ABC were entered
ABC
```

**P1ST applies to DMC30010,DMC21x3,RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@gallimc.com](mailto:documentation@gallimc.com)

## PC *Pulse Counter Enable*



PC n

|                 |          |                                                             |
|-----------------|----------|-------------------------------------------------------------|
| <b>Usage</b>    | PC n ... | Arguments specified with an implicit, comma-separated order |
| <b>Operands</b> | _PC      | Operand has special meaning, see Remarks                    |

### Description

The PC command enables the use of a pulse counter input on the RIO controller. When turned on, input DI3 will not be available as a general purpose input accessible with IF or @IN[ ] and will be reconfigured as a pulse counter.

### Arguments

| Argument | Value | Description                             | Notes                                 |
|----------|-------|-----------------------------------------|---------------------------------------|
| n        | 0     | IN3 set as a general purpose input      | Default                               |
|          | 1     | IN3 set as a rising edge pulse counter  | PC 1 also clears current pulse count  |
|          | -1    | IN3 set as a falling edge pulse counter | PC -1 also clears current pulse count |

### Remarks

- Digital Input 2 with HS Option: When the -HS (high speed counter) option is ordered with the RIO, IN2 will be the differential input for the counter input. IN2 will not be available as a general purpose input.
- Expanded Memory RIO: When using the expanded memory RIO, IN3 cannot be used as the Pulse Counter Input unless the -HS option is ordered. Refer to the user manual for RIO part numbers with expanded memory
- Maximum frequencies:
  - Without -HS option - 300Hz (50% duty cycle)
  - With -HS option - 3MHz (50% duty cycle)
- See the -HS section in the RIO-47xxx User Manual for more information.

### Operand Usage

- \_PC contains the number of pulses counted. \_PC will return an integer in the range of 2147483647 to -2147483648

### Examples

```
'Galil DMC Code Example
:PC1
:MG_PC
214
:MG_PC
515
:t=_PC
:MG t
718
```

#### PC applies to RIO47xxx

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)



## PM PWM output enable



PM n, n

|                 |              |                                                             |
|-----------------|--------------|-------------------------------------------------------------|
| <b>Usage</b>    | PM n ...     | Arguments specified with an implicit, comma-separated order |
| <b>Operands</b> | _PM0<br>_PM1 | Operand holds the value last set by the command             |

### Description

PM enables the PWM output feature for the RIO. The PWM frequency is configurable up to 20kHz and the duty cycle from 0 to 100%. The PWM output can also be used as the control signal for process control loops.

### Arguments

| Argument | Value | Description                                | Notes                                                                              |
|----------|-------|--------------------------------------------|------------------------------------------------------------------------------------|
| n        | 3     | PWM Mode                                   | PWM used as control signal in process control loop 0 or 1                          |
|          | 2     | PWM/Sign-Magnitude Mode                    | PWM used as control signal in process control loop 0 or 1                          |
|          | 1     | Generic PWM                                | High polarity with DY 100 (100% duty cycle) is the same as the "CB" state with PM0 |
|          | 0     | Default Value, PWM Off                     | PWM output disabled for corresponding output                                       |
|          | -1    | Generic PWM, inverted duty cycle           | Low polarity with DY 100 (100% duty cycle) is the same as the "SB" state with PM0  |
|          | -2    | PWM/Sign-Magnitude Mode, inverted sign bit | PWM used as control signal in process control loop                                 |
|          | -3    | PWM Mode, inverted duty cycle              | PWM used as control signal in process control loop                                 |

### Remarks

- First and second arguments apply to PWM signal on output 14 and 15, respectively
- When used in the process control loop, the first argument applies to process loop A, and the second to B
- The -PWM option is necessary for full resolution of the frequency. See the RIO user manual for more information.
- PWM Mode vs. PWM/Sign-Magnitude Mode
  - PWM Mode: 50% duty cycle for zero command signal. 0% for full negative signal, 100% for full positive signal.
  - PWM/Sign-Magnitude Mode: 0% duty cycle for zero command signal. 100% for full signal. Negative/positive bit on outputs 12 and 13 for control signals 14 and 15, respectively.

#### Steps to use the PWM as a control signal for the PID process loop

1. Set PWM Frequency (FQ)
2. Turn on the PWM mode (PM)
3. Define analog feedback channels
4. Set up analog input configuration as necessary (AQ)
5. Set PIDs (KP,KD,KI)
6. Turn on control loop (CL)
7. Set the process loop setpoint (PS)

### Examples


```
'Galil DMC Code Example
:FQ 100,200;' Set output 14 to 100 Hz, and output 15 to 200 Hz
:DY 50,25;' Set output 14 to 50%, output 15 to 25%
:PM 1,1;' Turn PWM mode on for outputs 14 and 15
```

```
'Galil DMC Code Example
REM Using the PWM as a control signal for the PID process loop
FQ 100,100;' Set PWM Freq
PM 3,3;' Turn on PWM/Sign-Magnitude Mode
AF 0,1;' Analog 0 as input to process loop A
AQ 0,2;' Set +/- 10v on analog input 1
AQ 1,2;' (RIO-47x2x only)
KD 10,10;' Set PID
KP 4,4
KI 0.5,0.5
CL 1,1;' 1000 Hz sample loop
PS 0,0;' Zero Setpoint
EN
```

PM applies to RIO47xxx

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

## PS Control Setpoint

|                                                                                  |
|----------------------------------------------------------------------------------|
|  |
| PSm= n                                                                           |
| PS n, n, n, n, n, n                                                              |

|                 |          |                                                                   |
|-----------------|----------|-------------------------------------------------------------------|
| <b>Usage</b>    | PSm= n   | Arguments specified with a single axis mask and an assignment (=) |
|                 | PS n ... | Arguments specified with an implicit, comma-separated order       |
| <b>Operands</b> | _PSm     | Operand holds the value last set by the command                   |

### Description

The PS command set the voltage setpoint for the process loop. The command is used in a process control loop to specify the analog value that will be used as the target voltage for the analog input.

### Arguments

| Argument | Min | Max | Default | Resolution | Description                       | Notes                                                |
|----------|-----|-----|---------|------------|-----------------------------------|------------------------------------------------------|
| <b>m</b> | A   | F   | N/A     | Axis       | Channel to assign value           | Valid for RIO-47xx2 and RIO-473xx                    |
|          | A   | B   | N/A     | Axis       | Channel to assign value           | Valid for RIO-47xx0                                  |
| <b>n</b> | -10 | 10  | -5      | 0.0001     | Voltage setpoint for process loop | Valid for RIO-47x2x. Range determined by AQ command. |
|          | 0   | 5   | 0       | 0.0001     | Voltage setpoint for process loop | Valid for RIO-47x0x                                  |

### Remarks

- The process control loop input and output (AF and AZ) must be set prior to adjusting PS.
- The range and resolution for the setpoint is dependant on the RIO hardware configuration. See the AQ command for the Analog Input range.

### Examples

```
'Galil DMC Code Example
PSA= 2.2;' set A setpoint to 2.2V
PS ,1.2;' set B setpoint to 1.2V
```

```
'Galil DMC Code Example
'enable process loop example
CL 25;' 25ms update rate
AF 0;' analog input 0 as feedback
AZ 0;' analog output 0 as control
KP 1;' proportional gain to 1
KD 10;' derivative gain to 10
KI 0.5;' integral gain to 0.5
DB 0.1;' deadband of 0.1V
PS 1.8;' set-point at 1.8V
```

#### PS applies to RIO47xxx

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**PW Password**

PW str, str

| 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.

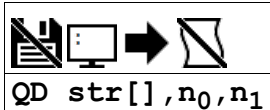
**Examples**

```
'Galil DMC Code Example
:PWapple,orange
?
:TC1
138 Passwords not identical
:PWapple,apple
:^L^K apple,1
```

```
'Galil DMC Code Example
:PWtest,test;'      Set password to "test"
:^L^K test,1;'      Lock the program
:ED;'              Attempt to edit program
?
:TC1
106 Privilege violation
```

**PW applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,RIO47xxx,DMC18x6**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**QD Download Array**

QD str[],n<sub>0</sub>,n<sub>1</sub>

|              |          |                                                             |
|--------------|----------|-------------------------------------------------------------|
| <b>Usage</b> | 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                                                                 |
|----------------------|--------|-----------|-----------|------------|---------------------------------------------|-----------------------------------------------------------------------|
| <b>str</b>           | 1 char | 7 chars   | N/A       | String     | Name of array to receive data via download. |                                                                       |
| <b>n<sub>0</sub></b> | 0      | see Notes | 0         | 1          | Index of the first array element.           | Value cannot exceed size of array - 2                                 |
| <b>n<sub>1</sub></b> | 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 may be separated by a comma ( , ), a carriage return (r), or a carriage return and line feed (r\n). Do not use spaces.
- The downloaded array is terminated by a \ character.
- QD is not supported in the terminal of Galil software packages.
  - It is recommended to use the array download functions available through the Galil software and drivers rather than directly using the QD command.
- Issuing this command will pause the output of the Data Record until the command is completed.

**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
```

**QD applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**QE** *Query Encoder*

QE n

|                 |              |                                                             |
|-----------------|--------------|-------------------------------------------------------------|
| <b>Usage</b>    | QE n ...     | Arguments specified with an implicit, comma-separated order |
| <b>Operands</b> | _QE0<br>_QE1 | Operand has special meaning, see Remarks                    |

**Description**

The QE command returns the current position of the encoders inputs. The encoder option is available for the RIO-47122 and RIO-473xx models. All RIO-47122 encoder options will utilize Digital Inputs 12,13,14 and 15 and Digital Outputs 12,13,14 and 15. These digital inputs and outputs will not be available as standard digital I/O when one of the encoder options is ordered with the RIO-47122. See the User manual for detailed pin out information. All RIO-473xx encoder options come in the form of add-on boards. More information about the RIO-473xx add-on boards can be found in the appendix of the RIO user manual .

**Arguments**

| Argument | Min | Max | Default | Resolution | Description                         | Notes                                                               |
|----------|-----|-----|---------|------------|-------------------------------------|---------------------------------------------------------------------|
| n        | 0   | 1   | N/A     | 1          | Channel selected to report position | If no number selected, both channel 0 and channel 1 report position |

**Remarks**

- The register that is read using the QE command is updated by the RIO every 25ms.
- Encoder support is only valid for RIO models that are ordered with the -QUAD, -SSI or -BISS options.

**Operand Usage**

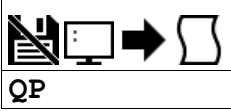
- \_QEn contains the current encoder position value for the specified channel (n = 0 or 1)

**Examples**

```
'Galil' DMC Code Example
:QE 0
10254
```

**QE applies to RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**QP Query Parameters**

QP

|              |    |                            |
|--------------|----|----------------------------|
| <b>Usage</b> | 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 1.1g 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,<br>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.                            |

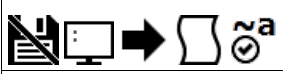
**Examples**

```
'Galil DMC Code Example
:ARV
RIO47202 Rev 1.1g
:QP;' only valid on firmware 1.1g and later
PR, 40, 402, 0
VA, 1000, 6, 254, 126
```

**QP applies to DMC500x0,DMC40x0,DMC42x0,DMC41x3,DMC30010,RIO47xxx,DMC18x6,DMC52xx0,EDD3701x**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

## QR I/O Data Record

|                                                                                  |
|----------------------------------------------------------------------------------|
|  |
| QR mm                                                                            |
| QR                                                                               |

|              |       |                          |
|--------------|-------|--------------------------|
| <b>Usage</b> | 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 I/O information. The details of the status information are described in the RIO user manual.

### Arguments

QR is an interrogation command with no arguments.

### Remarks

- The data returned by the QR command is in binary format and is unreadable in programs such as Galitools.
  - The Galitools API has specialized commands to parse the data record packet. See the Galitools User Manual for more details.

### Examples

```
'Galil DMC Code Example
:QR;' Return the data record, binary format
?8K)AAA ! !0!0! ! ! !
```

**QR applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**QU Upload Array**

**QU** `str[],n0,n1,n2`

|              |          |                                                             |
|--------------|----------|-------------------------------------------------------------|
| <b>Usage</b> | 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                                                                                 |
|----------------------|--------|-----------|-----------|------------|----------------------------------------------------|---------------------------------------------------------------------------------------|
| <b>str</b>           | 1 char | 7 chars   | N/A       | String     | Name of array to be uploaded                       |                                                                                       |
| <b>n<sub>0</sub></b> | 0      | see Notes | 0         | 1          | Index of first array element                       | Value cannot exceed size of array - 2                                                 |
| <b>n<sub>1</sub></b> | 1      | see Notes | see Notes | 1          | Index of last array element                        | Defaults to last element of array. Value cannot exceed size of array - 1              |
| <b>n<sub>2</sub></b> | 0      | 1         | 0         | 1          | Selects character delimiter between array elements | n <sub>2</sub> = 0 selects CR delimiting. n <sub>2</sub> = 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.
- Issuing this command will pause the output of the Data Record until the command is completed.

**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
```

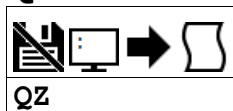
```
'Galil DMC Code Example
:DM array[5];'      Dimension Array
:QU array[],0,4,1;' Upload Array
0.0000, 0.0000, 0.0000, 0.0000, 0.0000
:array[0]=9;'      Set value
:array[1]=1
:QU array[],0,4,1
9.0000, 1.0000, 0.0000, 0.0000, 0.0000
:array[0]=?;'      Alternative method to return just one array value
9.0000
```

**QU applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@gallimc.com](mailto:documentation@gallimc.com)



## QZ Return Data Record information



QZ

|                 |                              |                                          |
|-----------------|------------------------------|------------------------------------------|
| <b>Usage</b>    | QZ                           | Command takes no arguments               |
| <b>Operands</b> | _QZ0<br>_QZ1<br>_QZ2<br>_QZ3 | Operand has special meaning, see Remarks |

### 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 control loops
  2. Second field returns the number of bytes in the general data block of the QR record. This is always 4 for the RIO.
  3. Third field returns the number of bytes in the I/O block of the QR record.
    - This value is different on the -SER version of firmware used when ordering an encoder option as compared to the standard firmware. This is to designate the extra bytes used to store encoder data.
  4. Fourth field returns 0, indicating there is no axis-specific data in the data record. This field is reserved because it has context on Galil motion controllers.

#### *\_QZ operands*

| Operand | Description                                                                                                                                      |
|---------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| _QZ0    | Holds the number of control loops                                                                                                                |
| _QZ1    | Holds the number of bytes in the general data block of the QR record. This is always 4 for the RIO                                               |
| _QZ2    | Holds the number of bytes in the I/O block of the QR record                                                                                      |
| _QZ3    | Holds 0, indicating there is no axis-specific data in the data record. This field is reserved because it has context on Galil motion controllers |

### Examples

```
'Galil DMC Code Example
:QZ;' standard RIO firmware response
2, 4, 48, 0
```

**QZ applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**RA** *Record Array*
**RA** *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                                                                                  |
|------------|--------|---------|---------|------------|--------------------------------------------------|----------------------------------------------------------------------------------------|
| <b>str</b> | 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**

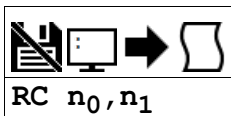
```
'Galil DMC Code Example
' try to start record array without defining array[]
:RA array[]
?
:TC1
82 Undefined array
:DM array[100]
:RA array[]
```

```
'Galil DMC Code Example
#Record; Label
DM input[100]; Define array for input status
RA input[]; Specify Record Mode
RD _TI; Specify data type for record
RC 1; Begin recording at 2 msec intervals
EN; End
```

```
'GalilTools: The GalilTools Realtime scope can
'often be used as an alternative to record array.
```

**RA applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**RC Record**RC  $n_0, n_1$ 

|                 |          |                                                             |
|-----------------|----------|-------------------------------------------------------------|
| <b>Usage</b>    | RC n ... | Arguments specified with an implicit, comma-separated order |
| <b>Operands</b> | _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                                                        |
|-----------|-----------|-----------|---------|------------|----------------------------------------------------------------|--------------------------------------------------------------|
| <b>n0</b> | 0         | 9         | 0       | 1          | Specify the record array time interval as $2^{(n-1)}$ samples. | $n_0 = 0$ and 1 have special behavior. See Remarks           |
| <b>n1</b> | 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.
- GallTools Note: Do not download arrays from GallTools, or call the arrayDownload() or arrayDownloadFile() functions while automatic record array mode is running.

**First Parameter Rules**

- $n_0 = \text{non zero number}$  automatically starts record mode.
- $n_0 = ?$  returns status of recording. '1' if recording, '0' if not recording.
- $n_0 = 0$  disables the record array mode
- $n_0 = 1$  sets the time interval to 1 millisecond for the record array mode.
- $n_0 = 2-9$  sets time interval to  $2^{(n-1)}$  milliseconds for a range of 2msecs - 256 msecs

**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**

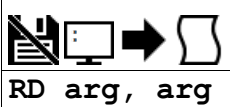
1. Dimension an array/arrays for storing data. Make sure you dimension the array with the number of elements required to capture data for your application.
2. Set the RA command with the arrays to be used for recording
3. Set the RD command with the data sources to be applied to the arrays. The order 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 Galltools software and the Upload array function.

**Examples**

```
'Gallil DMC Code Example
#record; Record label
DM anaout[1000]; Define Array
RA anaout[]; Specify Array to record data
RD _A01; Specify Data Type
RC 2; Begin recording and set 2 msec between records
JG 1000;BG; Begin motion
#A;JP #A,_RC=1; Loop until done
MG "DONE RECORDING"; Print message
EN; End program
```

**RC applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Gallil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@gallilmc.com](mailto:documentation@gallilmc.com)

**RD** *Record Data*

RD arg, arg

|                 |          |                                                             |
|-----------------|----------|-------------------------------------------------------------|
| <b>Usage</b>    | RD n ... | Arguments specified with an implicit, comma-separated order |
| <b>Operands</b> | _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**

| Argument   | Value | Description            | Notes                                                                                                                                                                                   |
|------------|-------|------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|            | _AFm  | Analog feedback status | m corresponds to the analog feedback axis. See AF command. <b>Syntax Note:</b> Unlike the operand _AFm, the symbol _AFm in the context of RD records the ADC value, not the AF setting. |
|            | _AOm  | Analog output status   | m corresponds to the analog output value. See AO command.                                                                                                                               |
|            | _OP   | Output status          | Returned bit mask represents all bits                                                                                                                                                   |
| <b>arg</b> | _TI   | Input status           | Returned bit mask represents all bits                                                                                                                                                   |

**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
#record;' Label
DM input[100];' Define array for input status
RA input[];' Specify Record Mode
RD _TI;' Specify data source for record
RC 1;' Begin recording at 2 msec intervals
EN;' End the program, RC continues to run until complete
```

'The GalilTools Realtime scope can often be used as an alternative to record array.'

```
'Galil DMC Code Example
AFA=0;' analog feedback ADC
' for control loop A is input 0
DA*[]
DM analog[200], time[200]
RA analog[], time[]
RD _AFA, TIME
RC 2
#LOOP;
WT2
JP #LOOP, _RC=1
MG"Done"
EN
```

**RD applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

## 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                                                   |
|----------------------------|---------------------------------------------------------|
| #SERERR                    | Only when equipped with serial encoder firmware support |
| #TCPERR                    |                                                         |

- 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)

### Examples

```
'Galil DMC Code Example
#L
MG {EA} "L"
WT1000
JP#L

#TCPERR
MG {P1} "TCPERR. Dropped handle", _IA4
RE
```

RE applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**REM Remark****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.

**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:
  - NO (or ') comments are downloaded to the controller and REM comments are not.
  - NO (or ') comments take up execution time and REM comments don't; therefore, REM should be used for code that needs to run fast.
  - REM comments cannot be recovered when uploading a program but NO (or ') comments are recovered. Thus the uploaded program is less readable with REM.
  - NO (or ') comments take up program line space and REM lines do not.
  - 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**

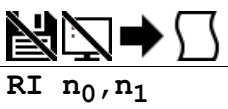
- REM DISABLE COMPRESSION
  - Inserting this line into the beginning of your application code disables Galitools download compression utility. This is not a controller function.

**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
PRX=1000 ;'this comment is to the right of the code
```

**REM applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**RI** *Return from Interrupt Routine*RI  $n_0, n_1$ 

|              |          |                                                             |
|--------------|----------|-------------------------------------------------------------|
| <b>Usage</b> | 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 #ININT0, #ININT1, #ININT2 and #ININT3. The second field of the RI command either restores or disables the input interrupt feature.

**Arguments**

| Argument             | Min | Max | Default | Resolution | Description                                                                            | Notes                                                                        |
|----------------------|-----|-----|---------|------------|----------------------------------------------------------------------------------------|------------------------------------------------------------------------------|
| <b>n<sub>0</sub></b> | 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.        |
| <b>n<sub>1</sub></b> | 0   | 1   | 0       | 1          | Sets whether the current input interrupt subroutine is re-enabled.                     | n = 0 re-enables the interrupt routine. n = 1 disables the interrupt routine |

**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. Refer to Application Note 2418.
  - <http://www.galil.com/download/application-note/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.
- An application program must be executing in thread 0 for the input interrupt subroutine to function.

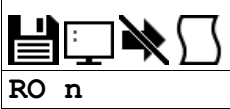
**Examples**

```
'Galil DMC Code Example
#A;'          Program label
II1,0,3;'    enable interrupt on input 3
AI5;'        AI trippoint on input 5
MG"DONE"
EN;'         end of program
#ININT1;'    Begin interrupt subroutine
MG "IN[3] INTERRUPTED"
CB 3;'       Set output line 1
RI 1,1;'     Return to the main program,
'restore AI trippoint and disable interrupt on input 3
```

**RI applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**RO Realtime Offset**



|                 |              |                                                             |
|-----------------|--------------|-------------------------------------------------------------|
| <b>Usage</b>    | RO n ...     | Arguments specified with an implicit, comma-separated order |
| <b>Operands</b> | _RO0<br>_RO1 | Operand has special meaning, see Remarks                    |

**Description**

The RO command sets the hour offset from GMT time for the realtime clock feature. RO is used when syncing the realtime clock with a TIME server. The TIME protocol provides GMT time, and RO allows for localization of time.

**Arguments**

| Argument | Min | Max | Default | Resolution | Description             | Notes                      |
|----------|-----|-----|---------|------------|-------------------------|----------------------------|
| n        | -12 | 12  | 0       | 1          | Hourly time zone offset | Offsets based on GMT time. |

**Remarks**

- RTC functionality is only available on the RIO-471x2 models

**Operand Usage**

- \_RO0 contains the current GMT offset
- \_RO1 contains the value last received from a network TIME protocol server (See IH and RT)

**Examples**

```
'Galil DMC Code Example
'Some Time Zone Examples:
RO -10;' Hawaii, USA
RO -9;' Alaska, USA
RO -8;' Pacific Time, US and Canada
RO -7;' Mountain Time, US and Canada
RO -6;' Central Time, US and Canada
RO -5;' Eastern Time, US and Canada
RO -3;' Buenos Aires, Argentina
RO 0;' London, England
RO 1;' Paris, France
RO 2;' Cairo, Egypt
RO 3;' Plovdiv, Bulgaria
RO 8;' Beijing, China
RO 9;' Tokyo, Japan
RO 10;' Melbourne, Australia
```

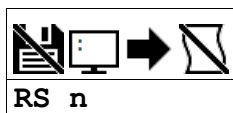
```
'Galil DMC Code Example
REM Changing RO for Daylight Savings
RO-7;'Pacific Daylight Time
REM Hit TIME server
IHE=10,0,62,23<37>2
WT10
JS#Print
RO-8;'Pacific Standard Time
IHE=10,0,62,23<37>2
WT10
JS#Print
EN
#Print;'Print Time
MG_RT2{F2.0};"{N}";'Hour
MG_RT1{F2.0};"{N}";'Minute
MG_RT0{F2.0};"{N}";'Second
EN
'SAMPLE OUTPUT:
' 11: 23: 16
' 10: 23: 16
```

**RO applies to DMC30010,RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)



## RS *Reset*



RS n

|                 |          |                                                             |
|-----------------|----------|-------------------------------------------------------------|
| <b>Usage</b>    | RS n ... | Arguments specified with an implicit, comma-separated order |
| <b>Operands</b> | _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 performs normal reset. n = -1 performs soft master reset. See Remarks. |

RS has no arguments.

### 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

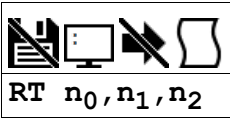
```
'Galil DMC Code Example
:RS;'   Reset the hardware
:RS-1;' Perform a soft master reset
```

### RS applies to

DMC500x0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2,RIO574x0,DMC52xx0,EDD3701x

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**RT** *Real Time*



RT n<sub>0</sub>, n<sub>1</sub>, n<sub>2</sub>

|                 |                              |                                                             |
|-----------------|------------------------------|-------------------------------------------------------------|
| <b>Usage</b>    | RT n ...                     | Arguments specified with an implicit, comma-separated order |
| <b>Operands</b> | _RT0<br>_RT1<br>_RT2<br>_RT3 | Operand has special meaning, see Remarks                    |

**Description**

The RT command provides the ability to set and query the current time on the realtime clock. RT can be set manually, or automatically with the TIME protocol (See IH).

**Arguments**

| Argument       | Min | Max | Default | Resolution | Description           | Notes                         |
|----------------|-----|-----|---------|------------|-----------------------|-------------------------------|
| n <sub>0</sub> | 0   | 59  | 0       | 1          | Current seconds value |                               |
| n <sub>1</sub> | 0   | 59  | 0       | 1          | Current minutes value |                               |
| n <sub>2</sub> | 0   | 23  | 0       | 1          | Current hour value    | 24 hour clock used (no AM/PM) |

**Remarks**

- Available on the RIO-471x2 only.
- The RIO-471x2 has two clock options:
  - 1. Processor RTC. Clock does not persist through power cycles and must be set at startup either by RT command or by TIME protocol over Ethernet (See IH).
  - 2. Precision RTC chip. Upgrade option for the RIO-41x2. More precise than processor RTC feature. Persists time through power cycle and Master Reset. Provides calendar function (See RY).

**Operand Usage**

- \_RT0 contains the seconds field of the current time.
- \_RT1 contains the minutes field of the current time.
- \_RT2 contains the hours field of the current time.
- \_RT3 (without precision RTC circuitry) contains the days since the time was last set with the RT command or with the TIME protocol.
  - If neither event occurred, \_RT3 contains the number of days since last power cycle.
  - With the precision RTC upgrade, use \_RY0 for the day of week counter.

**Examples**

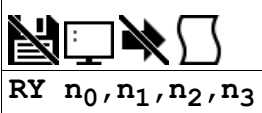
```
'Galil DMC Code Example
:RT 30,25,22;' Set time to 10:25:30 PM
```

```
'Galil DMC Code Example
RO-7;' set timezone
IHE=10,0,62,23<37>2;' hit TIME server
WT10
MG_RO1{$8.0};' print data from server
MG_RT2{F2.0};' print hour
MG_RT1{F2.0};' print minutes
MG_RT0{F2.0};' print seconds
EN
'Sample Output:
'$CF943C7B
' 13: 28: 43
```

**RT applies to DMC30010,RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**RY Real Year Calendar Function**



|                 |                              |                                                             |
|-----------------|------------------------------|-------------------------------------------------------------|
| <b>Usage</b>    | RY n ...                     | Arguments specified with an implicit, comma-separated order |
| <b>Operands</b> | _RY0<br>_RY1<br>_RY2<br>_RY3 | Operand has special meaning, see Remarks                    |

**Description**

RY provides a calendar feature for the realtime chip. Available information is day of week, day of month, month of year, and year. RY can be set manually, or automatically with the TIME protocol (See IH). The state of the calendar will persist through power cycle and Master Reset.

**Arguments**

| Argument  | Min | Max | Default | Resolution | Description                           | Notes                                                                                                          |
|-----------|-----|-----|---------|------------|---------------------------------------|----------------------------------------------------------------------------------------------------------------|
| <b>n0</b> | 0   | 7   | 0       | 1          | Day of the week to assign to calendar | n0 = 0 disables calendar feature. 1=Sunday, 2=Monday, 3=Tuesday, 4=Wednesday, 5=Thursday, 6=Friday, 7=Saturday |
| <b>n1</b> | 0   | 31  | 0       | 1          | Sets the current day of the month     |                                                                                                                |
| <b>n2</b> | 0   | 12  | 0       | 1          | Sets the current month of the year    | n2 = 0 disables calendar feature. 1=January, 2=February, 3=March, ....., 10=October, 11=November, 12=December  |
| <b>n3</b> | 0   | 99  | 0       | 1          | Sets the current year                 | 0 - 99 corresponds to year 2000 through 2099                                                                   |

**Remarks**

- Available only on RIO-471x2 with -RTC order option for precision realtime clock chip
- The calendar function is leap-year compliant.

**Operand Usage**

- \_RY0 contains the current day of the week field
- \_RY1 contains the current day of the month field
- \_RY2 contains the current month of the year field
- \_RY3 contains the current year field

**Examples**

```
'Galil DMC Code Example
:RY6,19,2,10;' Set to Fri, February 19th, 2010
```

```
'Galil DMC Code Example
REM DISABLE COMPRESSION <- This String Disables GalilTools Compression
#Print;'call sub when time is needed
JS#PrintD+_RY0;'helper subs w/ offsets
JS#PrintM+_RY2
MG_RY1{Z2.0},"{N}';print info
MG2000+_RY3{Z4.0}
MG_RT2{F2.0},"{N}
MG_RT1{F2.0},"{N}
MG_RT0{F2.0}
EN
'
REM The following Subs depend upon line spacing
REM Do not add or remove lines
#PrintD
MG"SUN" "{N}";EN
MG"MON" "{N}";EN
MG"TUE" "{N}";EN
MG"WED" "{N}";EN
MG"THR" "{N}";EN
MG"FRI" "{N}";EN
MG"SAT" "{N}";EN
'
#PrintM
MG"JAN" "{N}";EN
MG"FEB" "{N}";EN
MG"MAR" "{N}";EN
MG"APR" "{N}";EN
MG"MAY" "{N}";EN
MG"JUN" "{N}";EN
MG"JUL" "{N}";EN
MG"AUG" "{N}";EN
MG"SEP" "{N}";EN
MG"OCT" "{N}";EN
'
```

```
MG"NOV "{N}";EN  
MG"DEC "{N}";EN
```

```
'Example Output  
'TUE MAY 11, 2010  
' 14: 07: 48
```

**RY applies to DMC30010,RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**SA** *Send Command*

**SAm**= str

**SAm**= str,n,n,n,n,n,n,n,n

| Usage | SAm= n | Arguments specified with a single axis mask and an assignment (=) |
|-------|--------|-------------------------------------------------------------------|
|-------|--------|-------------------------------------------------------------------|

**Description**

SA sends a command, and optionally receives a response, from one controller to another via Ethernet.

**Arguments**

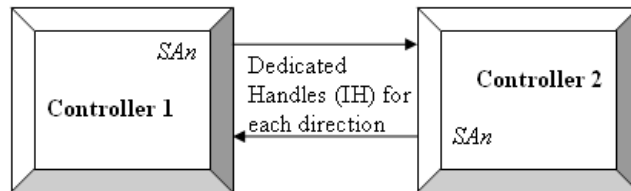
| Argument   | Min            | Max           | Default | Resolution | Description                               | Notes                       |
|------------|----------------|---------------|---------|------------|-------------------------------------------|-----------------------------|
| <b>m</b>   | A              | C             | N/A     | Handle     | Handle to specify for message output      | For RIO-47xx0               |
|            | A              | E             | N/A     | Handle     | Handle to specify for message output      | For RIO-47xx2 and RIO-473xx |
| <b>str</b> | 1 char         | 6 chars       | ""      | String     | String to send over handle                |                             |
| <b>n</b>   | -2,147,483,648 | 2,147,483,647 | 0       | 1          | Value to send for the specified parameter |                             |

**Remarks**

- Strings are encapsulated by quotations. This will typically begin an SA command.
- n is a number, controller operand, variable, mathematical function, or string. The range for numeric values is 4 bytes of integer followed by two bytes of fraction.
- Typical usage would have the first argument as a string such as "KI" and the subsequent arguments as the arguments to the command:
  - Example SAF="KI", 1, 2 would send the command: KI1,2
  - SA automatically adds commas between two number values being sent.
- There is a 38 character maximum payload length for the SA command.

**Operational Notes**

1. SA is non-blocking. A wait (e.g. WT10) must occur between successive calls to SA.
2. SA is not valid over a handle configured for Modbus (port 502).
3. When writing multi-threaded DMC code, send all traffic from only one thread.
4. SA command functionality is provided to assist with controller to controller communications under error handling conditions. Recommended commands for use with SA include, but are not limited to: HX, XQ, ST, and AB. Please contact a Galil Applications Engineer for additional details.

**Operand Usage**

- **\_SAmn** gives the value of the response to the command sent with an SA command.
  - The m value represents the handle A thru H and the n value represents the specific field returned from the controller (0-7).
  - If the specific field is not used, the operand will be -2^31.

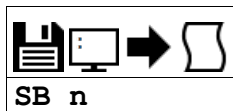
**Examples**

```
'Galil DMC Code Example
#A
IHA=10,0,0,12;'   Configures handle A to be connected to a controller with IP 10.0.0.12
#B:JP#B,_IHA2<-2;' wait for connection
SAA="KI", 1, 2;'  Sends the command to handle A (slave controller): KI 1,2
WT10
EN;'              End Program
```

**SA applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

## SB Set Bit



| 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. When an output has been set, current will begin flowing through the optocoupler for that output.

### Arguments

| Argument | Min   | Max   | Default | Resolution | Description                  | Notes                                                             |
|----------|-------|-------|---------|------------|------------------------------|-------------------------------------------------------------------|
| n        | 0     | 15    | N/A     | 1          | General output bit to be set | RIO-471xx and RIO-472xx                                           |
|          | 0     | 23    | N/A     | 1          | General output bit to be set | RIO-473xx                                                         |
|          | 24    | 47    | N/A     | 1          | General output bit to be set | RIO-473xx with -24ExOut option                                    |
| n        | 1,000 | 3,999 | N/A     | 1          | Set Modbus slave bit         | For RIO-47xx0. See "SB via Modbus Slave" in Remarks               |
|          | 1,000 | 5,999 | N/A     | 1          | Set Modbus slave bit         | For RIO-47xx2 and RIO-473xx. See "SB via Modbus Slave" in Remarks |

### Remarks

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

#### Using SB with a Modbus Slave

- $n = (\text{SlaveAddress} * 10000) + (\text{HandleNum} * 1000) + ((\text{Module}-1) * 4) + (\text{Bitnum}-1)$ 
  - Slave Address is used when the Modbus device has slave devices connected to it and specified as Addresses 0 to 255. Please note that the use of slave devices for modbus are very rare and this number will usually be 0.
  - HandleNum is the handle specifier where A is 1, B is 2 and so on.
  - Module is the position of the module in the rack from 1 to 16.
  - BitNum is the I/O point in the module from 1 to 4

### 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
```

```
'Galil DMC Code Example
#modbus
REM connect to modbus slave at IP address 192.168.1.50
IHF=192,168,1,50<502>2
WT100
SB 5001;'set bit 1 on modbus slave
WT 10
CB 5003;'set bit 3 on modbus slave
EN
```

For detailed information on connecting to a Modbus slave, see:

<http://www.galil.com/news/dmc-programming-io-control/setting-rio-pocket-plc-or-generic-modbus-slave-extended-io>

#### SB applies to

**DMC500x0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2,RIO574x0,DMC52xx0,EDD3701x**

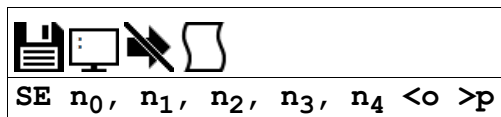
©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

## Description

This command is used to configure an RIO with the -SSI or -BISS option to read either SSI or BISS serial encoders. Both serial encoder types receive a master clock which is used to report position data serially to the RIO.

## Arguments

### SE Serial Encoder



SE  $n_0, n_1, n_2, n_3, n_4$  <o> p

|                 |              |                                                             |
|-----------------|--------------|-------------------------------------------------------------|
| <b>Usage</b>    | SE n ...     | Arguments specified with an implicit, comma-separated order |
| <b>Operands</b> | _SE0<br>_SE1 | Operand has special meaning, see Remarks                    |

| Argument             | Min | Max | Default | Resolution | Description         | Notes                                             |
|----------------------|-----|-----|---------|------------|---------------------|---------------------------------------------------|
| <b>n<sub>0</sub></b> | 0   | 1   | 0       | 1          | Channel to set      | n = 0 sets channel 0. n = 1 sets channel 1        |
| <b>n<sub>1</sub></b> | 0   | 2   | 0       | 1          | Select encoder type | 0 = No encoder, 1 = SSI encoder, 2 = BISS encoder |

#### For SSI encoders

| Argument             | Min | Max | Default | Resolution | Description                   | Notes                                                                                                   |
|----------------------|-----|-----|---------|------------|-------------------------------|---------------------------------------------------------------------------------------------------------|
| <b>n<sub>2</sub></b> | -31 | 31  | N/A     | 1          | Total number of bits          | n <sub>2</sub> < 0 enables rollover. See Remarks.                                                       |
| <b>n<sub>3</sub></b> | 0   | 31  | N/A     | 1          | Number of single-turn bits    |                                                                                                         |
| <b>n<sub>4</sub></b> | -8  | 8   | N/A     | 1          | Number of status (error) bits | n <sub>4</sub> < 0 specifies bits leading SSI data. n <sub>4</sub> > 0 specifies bits trailing SSI data |
| <b>o</b>             | 4   | 26  | N/A     | 1          | Clock divider                 | Read data on rising clock edge                                                                          |
|                      | -26 | -4  | N/A     | 1          | Clock divider                 | Read data on falling clock edge. See Table                                                              |
| <b>p</b>             | 1   | 2   | N/A     | 1          | Select data encoding style    | 1 = Binary, 2 = Gray Code                                                                               |

#### For BISS encoders

| Argument             | Min | Max | Default | Resolution | Description                            | Notes                                             |
|----------------------|-----|-----|---------|------------|----------------------------------------|---------------------------------------------------|
| <b>n<sub>2</sub></b> | -31 | 31  | N/A     | 1          | Number of single-turn bits             | n <sub>2</sub> < 0 enables rollover. See Remarks. |
| <b>n<sub>3</sub></b> | 0   | 31  | N/A     | 1          | Number of total bits before error bits | See Example for BISS example                      |
| <b>n<sub>4</sub></b> | 0   | 8   | N/A     | 1          | Number of zero padding bits            |                                                   |
| <b>o</b>             | 4   | 26  | N/A     | 1          | Clock divider                          | Read data on rising clock edge. See Table         |
|                      | -26 | -4  | N/A     | 1          | Clock divider                          | Read data on falling clock edge. See Table        |

## Remarks

- The -BISS/-SSI option is required to use this feature. See the User Manual for a list of RIOs that this option can be added to.
- For n<sub>2</sub>, A positive number designates No Rollover. A negative number will cause the RIO to act as an incremental encoder, allowing the encoder to count past the max value of the encoder.
  - Note: when the controller is powered down, the rollover values are lost
- The clock frequencies specified below by 'o' are approximate values and may vary from RIO to RIO.

| 'o' Value | Clock Frequency (MHz) |
|-----------|-----------------------|
| 4         | 2                     |
| 5         | 1.7                   |
| 6         | 1.4                   |
| 7         | 1.3                   |
| 8         | 1.1                   |
| 9         | 1.0                   |
| 10        | 0.91                  |
| 11        | 0.83                  |
| 12        | 0.77                  |
| 13        | 0.71                  |
| 14        | 0.67                  |
| 15        | 0.63                  |
| 16        | 0.59                  |
| 17        | 0.56                  |
| 18        | 0.53                  |
| 19        | 0.50                  |
| 20        | 0.48                  |
| 21        | 0.45                  |
| 22        | 0.43                  |
| 23        | 0.42                  |

|    |      |
|----|------|
| 24 | 0.40 |
| 25 | 0.38 |
| 26 | 0.37 |

### SSI

- Synchronous Serial Interface (SSI) allows for serial transmission of absolute position data (either binary or Gray code) from the encoder based on a timed clock pulse train from the controller.
- Connection between the controller and encoder is based on two signal lines, clock and data, which are usually differential for increased noise immunity.
- For each sequential clock pulse of the controller, the encoder transmits one data bit from shift registers on the encoder.
- The number of Multi-turn bits of the encoder is internally calculated by the following equation: Multi-turn Bits = (Total Bits) - (Single-Turn) - (Status Bits)

### BISS

- BISS is an open source digital interface for sensors and actuators. BISS is hardware compatible to the industrial standard SSI (Serial Synchronous Interface). It allows serial transmission of absolute position data from BISS encoders based on a master clock signal from the controller.
- Galil implementation of BISS is unidirectional. Contact Galil for a different implementation

### Operand Usage

- `_SEi` contains the status bit values where `i` = the channel number (0 or 1). The bitmap is found below
- Note: The encoder manufacturer may name the Error and Warning bits differently. Consult the encoder documentation for the naming convention.
- Galil defines the Warning bit as the bit directly preceding the CRC. The Error bit is defined as the bit directly preceding the Warning bit. See table 1.

#### SEI Bit Map

| Bit Position | Bit Meaning                             | Description                                                                                                                                          |
|--------------|-----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0            | No timeout = 0, timeout occurred = 1    | The BISS decoding hardware will timeout if the encoder doesn't set the start bit within 30uS                                                         |
| 1            | CRC valid = 0, invalid = 1              | BISS employs a Cyclic Redundancy Check to verify data after transmission                                                                             |
| 2            | Error bit* (active state set with SY)   | When SY is set correctly, this bit should be low when there is no active warning. Consult the encoder documentation for the Warning bit definition   |
| 3            | Warning bit* (active state set with SY) | When SY is set correctly, this bit should be low when there is no active alarm/error. Consult the encoder documentation for the Alarm bit definition |

### Examples

```
'Galil DMC Code Example
SE0,1,25,25,0<10>1;' SSI on channel 0, 25 bits total, all single turn, no status
SE0,0;' Disable serial encoder on channel 0
```

Table 1: SE Example for BISS encoder: Hengstler 12 bit MT 10 bit ST

| Bit Sequence:         | T-2  | T-1 (Delay)       | T0        | T1... T12           | T12... T22       | T23... T26   | T27                             | T28                         | T29... T34                              | T35                |
|-----------------------|------|-------------------|-----------|---------------------|------------------|--------------|---------------------------------|-----------------------------|-----------------------------------------|--------------------|
| Data (Data/SLO line): | 1    | 0                 | 1         | M11... M0           | S9... S0         | 0            | E                               | W                           | C5... C0                                | MCD                |
| Data Description:     | Idle | Encoder acquiring | Start Bit | Multi-Turn data     | Single-turn data | Zero Padding | Error Bit                       | Warning Bit                 | CRC                                     | Multi-Cycle Data   |
| SE command details:   | -    | -                 | -         | n3 = (10+12+4) = 26 | n2 = 10          | n4 = 4       | E bit read in <code>_SEn</code> | W read in <code>_SEn</code> | CRC valid bit read in <code>_SEn</code> | Ignored by default |

```
'Galil DMC Code Example
'Final command for table above
'on channel 0, with 2MHz clock
SE 0,2,10,26,4<4
EN
```

### SE applies to RIO47xxx

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)



## SI Configure the special Galil SSI feature



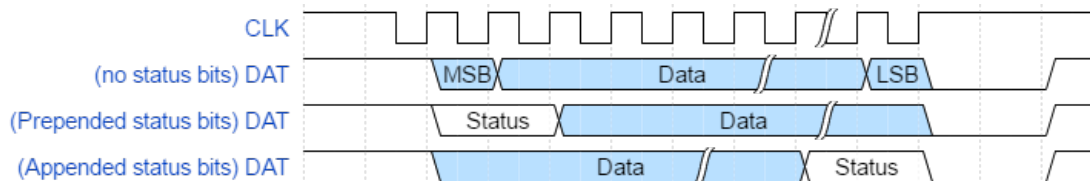
$SI_{m} = n_0, n_1, n_2, n_3 < o > p$

|                 |              |                                                                   |
|-----------------|--------------|-------------------------------------------------------------------|
| <b>Usage</b>    | $SI_{m} = n$ | Arguments specified with a single axis mask and an assignment (=) |
| <b>Operands</b> | $\_SI_{m}$   | Operand has special meaning, see Remarks                          |

### Description

The SI command enables and configures the controller to read SSI encoder data. Synchronous Serial Interface (SSI) allows for serial transmission of absolute position data (either binary or Gray code) from the encoder based on a timed clock pulse train from the controller. Connection between the controller and encoder is based on two signal lines, clock and data, which are usually differential for increased noise immunity. For each sequential clock pulse of the controller, the encoder transmits one data bit from shift registers on the encoder.

#### SSI Timing



During the controller's control loop, a clock will be transmitted for encoders configured for SSI. The SSI encoder responds by transmitting a serial data stream synchronized to the controller's clock. The data stream has 2 important features as described below.

- Data.** Shown in the above timing diagrams, *Data* contains the absolute position of the SSI encoder. *Data* is used for the  $n_2$  argument of the SI command
- Status (if applicable).** Shown above in the latter two timing diagrams, *Status* is used for the  $n_3$  argument of the SI command. The meaning of the *Status* and the is defined by the encoder manufacturer. The controller provides access to this value via the  $\_SI_{m}$  operand.
  - The sign of argument  $n_3$  selects if the *Status* are at the beginning of the SSI data or at the end. This is shown in the above timing diagrams and is based on the encoder's manufacturer.

### Arguments

- (2) The firmware will use the position data in one of two modes:
  - Absolute Mode**,  $n_1 < 0$ . The controller will use the absolute position as transmitted by the encoder. If the position rolls over either from zero to the highest position, or from the highest position to zero, the controller will calculate an incorrect discontinuity in position error. Absolute Mode is typically used with linear encoder applications where the motor will not cross zero or the highest position value.
  - Continuous Mode**,  $n_1 > 0$ . If the absolute position rolls over either from zero to the highest position, or from the highest position to zero, the firmware will artificially count past the absolute boundaries. This maintains continuity in the number line. Continuous Mode is typically used with rotary encoders that can move past their absolute range. When turning on the SSI function with SI, the data first loaded in Continuous Mode is the same as Absolute Mode.

### Remarks

- $SI_{m} = ?$  Returns the configuration parameters
- The number of Multi-turn bits of the encoder is internally calculated by the following equation:
  - Multi-turn Bits = (Total Bits) - (Single-Turn) - (StatusBits)
- See Application Note 2438 for more information

#### Required Hardware and Firmware

##### Operands

- $\_SI_{m}$  returns the value of the SSI encoder's status bits, as defined in  $n_3$ . The returned decimal number represents an 8 bit binary value. The meaning of the status bits is defined by the encoder manufacturer.

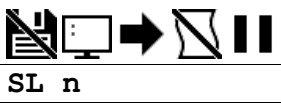
### Examples

```
'Galil DMC Code Example
SIA= 0;'           Disable SSI on axis A
```

#### SI applies to SER

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

## 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
```

**SL applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**SM Subnet Mask**

SM n<sub>0</sub>, n<sub>1</sub>, n<sub>2</sub>, n<sub>3</sub>

SM n

|                 |          |                                                             |
|-----------------|----------|-------------------------------------------------------------|
| <b>Usage</b>    | SM n ... | Arguments specified with an implicit, comma-separated order |
| <b>Operands</b> | _SM0     | Operand has special meaning, see Remarks                    |

**Description**

The SM command assigns a subnet mask to the controller. All packets sent to the controller whose source IP address is not on the subnet will be ignored by the controller. For example, for SM 255,255,0,0 and IA 10,0,51,1, only packets from IP addresses of the form 10.0.xxx.xxx will be accepted.

**Arguments**

| Argument       | Min            | Max           | Default | Resolution | Description                                                                | Notes |
|----------------|----------------|---------------|---------|------------|----------------------------------------------------------------------------|-------|
| n <sub>0</sub> | 0              | 255           | 0       | 1          | Byte 3 of the Subnet mask                                                  |       |
| n <sub>1</sub> | 0              | 255           | 0       | 1          | Byte 2 of the Subnet mask                                                  |       |
| n <sub>2</sub> | 0              | 255           | 0       | 1          | Byte 1 of the Subnet mask                                                  |       |
| n <sub>3</sub> | 0              | 255           | 0       | 1          | Byte 0 of the Subnet mask                                                  |       |
| n              | -2,147,483,648 | 2,147,483,647 | 0       | 1          | The full subnet mask specified as a signed 32 bit two's complement integer |       |

**Remarks**

- n = ? will return the subnet mask of the controller as n<sub>0</sub>,n<sub>1</sub>,n<sub>2</sub>,n<sub>3</sub>
- \_SM0 contains the subnet mask representing a 32 bit signed number (Two's complement)
- Use the following equation to change the 4 byte subnet (n<sub>0</sub>,n<sub>1</sub>,n<sub>2</sub>,n<sub>3</sub>) to a single 32 bit number, n
  - $n = (n_0 * 2^{24}) + (n_1 * 2^{16}) + (n_2 * 2^8) + n_3$
- For more information, see <http://www.galil.com/news/dmc-programming-software/blocking-unwanted-ethernet-devices-connecting>


**Examples**

```
'Galil DMC Code Example
SM 255,255,255,255;' Ignore all incoming Ethernet packets
SM 0,0,0,0;' Process all incoming Ethernet packets
```

**SM applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

## SS Configure the special Galil BiSS feature



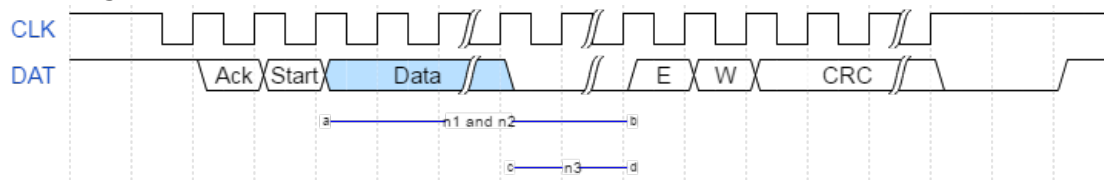
$SSm = n_0, n_1, n_2, n_3 < o$

|                 |        |                                                                   |
|-----------------|--------|-------------------------------------------------------------------|
| <b>Usage</b>    | SSm= n | Arguments specified with a single axis mask and an assignment (=) |
| <b>Operands</b> | _SSm   | Operand has special meaning, see Remarks                          |

### Description

The SS command enables and configures the controller to read BiSS encoder data. BiSS is an open-source, digital interface for encoders. BiSS is hardware compatible to the industrial standard SSI (Serial Synchronous Interface) but offers additional features and options.

#### BiSS Timing



During the controller's servo interrupt, a clock will be transmitted for encoders configured for BiSS. The BiSS encoder responds by transmitting a serial data stream synchronized to the controller's clock. The data stream has five important features as described below.

- Data.** Shown in the above timing diagram as the bit stream between *a* and *b*, *Data* contains the absolute position of the BiSS encoder. The length of the *Data*, in bits, is used for both the  $n_1$  and  $n_2$  arguments of the SS command.
- Zero Padding.** Some encoder manufacturers (Hengstler and possibly others) pad the encoder data with trailing zero bits in the *Data* stream. This bit segment is shown between *c* and *d* above. The number of zero padding bits is the  $n_3$  argument of the SS command.
- Error Bit.** Shown above as *E*, BiSS defines an Error bit that is transmitted by the encoder. It can be active high or low, depending on the manufacturer. The controller provides access to the Error bit via the `_SSm` operand.
- Warning Bit.** Shown above as *W*, BiSS also defines a Warning bit that is transmitted by the encoder. It is also active high or low based on the encoder. The controller provides access to the Warning bit via the `_SSm` operand.
- CRC.** The *Data*, *E* and *W* bits are all checked for validity via a 6-bit cyclic redundancy check, shown above as *CRC*. The controller provides access to the CRC validity via the `_SSm` operand.

### Arguments

- (2) The firmware will use the position data in one of two modes:
  - Absolute Mode**,  $n_1 = n_2$ . The controller will use the absolute position as transmitted by the encoder. If the position rolls over either from zero to the highest position, or from the highest position to zero, the controller will calculate an incorrect discontinuity in position error. Absolute Mode is typically used with linear encoder applications where the motor will not cross zero or the highest position value.
  - Continuous Mode**,  $n_1 = -n_2$ . If the absolute position rolls over either from zero to the highest position, or from the highest position to zero, the firmware will artificially count past the absolute boundaries. This maintains continuity in the number line. Continuous Mode is typically used with rotary encoders that can move past their absolute range. When turning on the BiSS function with SS, the data first loaded in Continuous Mode is the same as Absolute Mode.
- (3) Galil recommends using  $o=9$  for most encoders (1 MHz).
  - To calculate  $o$  for a desired frequency ( $f$ ) in MHz, use the following equation. Round to the closest integer in range.
    - $o = (10/f) - 1$

### Remarks

- SSm=? Returns the configuration parameters.

#### Required Hardware and Firmware

#### Operand Usage

- \_SSm Returns 4 bits of axis status data where m is the axis.

SSm Bit Map

| Bit Position | Bit Meaning     | Bit value = 0 | Bit value = 1    | Description                                                              |
|--------------|-----------------|---------------|------------------|--------------------------------------------------------------------------|
| 0            | Timeout Status  | No timeout    | Timeout occurred | Timeout will be set if the encoder doesn't set the start bit within 30uS |
| 1            | CRC Status      | CRC valid     | CRC invalid      | Indicates data validity                                                  |
| 2            | Error bit (1)   | No Error      | Error            |                                                                          |
| 3            | Warning bit (1) | No Warning    | Warning          |                                                                          |

- (1) The SY command must be set for the Error and Warning bits to report correctly. Consult the encoder documentation for definitions and active level of the Error and Warning bits.

#### Error Handling

- #SERERR is an automatic subroutine which will run in the event of the following encoder problems:
  - Encoder Timeout
  - CRC failure

- o Error bit set
- o Warning bit set
- Error bit active state set with SY command.
- Warning bit active state set with SY command.

### SS Command by Manufacturer

Axis A main encoder setup

| Manufacturer          | Part Number                  | SS Command          |
|-----------------------|------------------------------|---------------------|
| Hengstler ACURO       | AC58 1214 x x.xx BI x        | SSA = 1,26,26,0<13  |
| Hengstler ACURO       | AD36 1219 x x.xx BI x-xx     | SSA = 1,36,36,5<13  |
| Hengstler ACURO       | AC110 0017 x x.x xxx BI x-xx | SSA = 1,17,17,0<10  |
| Renishaw RESOLUTE (I) | RA 18B xx xxxx xx x          | SSA = 1,-19,19,0<10 |
| Renishaw RESOLUTE (I) | RL 26B xx xxxx xx x          | SSA = 1,27,27,0<14  |
| Renishaw RESOLUTE (I) | RL 36B xx xxxx xx x          | SSA = 1,37,37,0<10  |

- (I) Renishaw encoders transmit a leading zero in the *Data* field. A 26 bit encoder would use 27 for both n<sub>1</sub> and n<sub>2</sub>.

### Examples

```
'Galil DMC Code Example
'Setup Renishaw RESOLUTE 26-bit linear encoder
'Renishaw encoders have active low error and warning bits
'If the error bit is active or CRC fails
'the encoder data should not be considered valid
'Turn off output 1 and 3 if position data is not valid
#start
SYA=0;' Set Error bit = Active Low - Set Warning bit = Active Low
SSA=1,27,27,0<4;' 26-bit linear encoder, add 1 bit for leading zero on Renishaw
TP;'Get absolute position
'... continue setup
EN

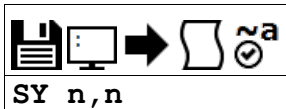
#SERERR;' display error, shutdown axis if data is invalid
MG "SERERR ", _SSA{$4};'Display hex value of SS bit mask
MG "TC: ", _TC;'Display any TC error codes
bisscode = _SSA;'Query SS bit mask
IF (bisscode & $1);'First bit in the mask is the Timeout bit
MG "BiSS Timeout";
'Add timeout code here
SSA=0;'Turn off BiSS encoder
ENDIF
IF (bisscode & $2);'Second bit in the mask is the CRC validity bit
MG "Invalid CRC"
'Add CRC failure code here, make the system safe
SSA=0;'Turn off BiSS
CB 1;'Turn off output 1
CB 3;'Turn off output 3
AB;'Abort motion and program operation
ENDIF
IF (bisscode & $4);'Third bit in the mask is the error bit
MG "Error Bit Set";
'Add Error code here, make the system safe
SSA=0;'Turn off BiSS
CB 1;'Turn off output 1
CB 3;'Turn off output 3
AB;'Abort motion and program operation
ENDIF
IF (bisscode & $8);'Fourth bit in the mask is the warning bit
MG "Warning Bit Set";
'Add warning code here, Notify and log but don't stop motion
XQ#logwarn,3;'Run routine to Log the warning and notify user
ENDIF
RE
```

### Errors

| Error Number | Description                    | Cause                                                               |
|--------------|--------------------------------|---------------------------------------------------------------------|
| 6            | Number out of range            | Argument out of bounds                                              |
| 140          | Serial Encoder Missing         | Encoder did not respond                                             |
| 141          | Incorrect ICM Configuration    | Incompatible hardware (requires BiSS part number option)            |
| 143          | TM Timed Out                   | Encoder start bit timed out                                         |
| 168          | Interrupt time period too long | Controller calculates not enough time to complete data transmission |

### SS applies to SER

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**SY** Serial encoder BiSS active level

SY n, n

|                 |          |                                                                   |
|-----------------|----------|-------------------------------------------------------------------|
| <b>Usage</b>    | SYm= n   | Arguments specified with a single axis mask and an assignment (=) |
|                 | SY n ... | Arguments specified with an implicit, comma-separated order       |
| <b>Operands</b> | _SYm     | Operand holds the value last set by the command                   |

**Description**

This command is used to designate the active level of the Error and Warning bits when using the Galil BiSS upgrade. The BiSS protocol defines two bits which can be used by the encoder to signal various events.

**Arguments**

| Argument | Value | Description                                        | Notes   |
|----------|-------|----------------------------------------------------|---------|
| n        | 0     | Warning bit = Active Low; Error bit = Active Low   |         |
|          | 1     | Warning bit = Active Low; Error bit = Active High  |         |
|          | 2     | Warning bit = Active High; Error bit = Active Low  |         |
|          | 3     | Warning bit = Active High; Error bit = Active High | Default |

**Remarks**

- RIO-47122-SSI or RIO-47122-BISS is required for use of this command
- The encoder manufacturer dictates the high/low active state of both of these bits. Consult your encoder documentation for details.
- The SY mask should be set appropriately to ensure that the #SERERR automatic subroutine will run when the bits are active, and that the \_SEN operand reports the fault state of the encoder correctly.
- Example of Warning and Alarm/Error bit use, Quoted from Renishaw Data Sheet L-9709-9005-03-A
  - Error (1 bit) "The error bit is active low: "1" indicates that the transmitted position information has been verified by the readhead's internal safety checking algorithm and is correct; "0" indicates that the internal check has failed and the position information should not be trusted. The error bit is also set to "0" if the temperature exceeds the maximum specification for the product."
  - Warning (1 bit) "The warning bit is active low: "0" indicates that the encoder scale (and/or reading window) should be cleaned. Note that the warning bit is not an indication of the trustworthiness of the position data. Only the error bit should be used for this purpose."

**Examples**

```
'Galil DMC Code Example
'configure SY for Renishaw Resolute encoder
SYA=0
```

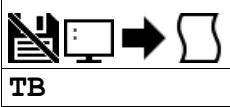
```
'Galil DMC Code Example
'Setup Renishaw RESOLUTE 26-bit linear encoder
'Renishaw encoders have active low error and warning bits
'If the error bit is active or CRC fails
'the encoder data should not be considered valid
'Turn off output 1 and 3 if position data is not valid
#start
SYA=0;' Set Error bit = Active Low - Set Warning bit = Active Low
SSA=1,27,27,0<4;' 26-bit linear encoder, add 1 bit for leading zero on Renishaw
TP;'Get absolute position
'... continue setup
EN

#SERERR;' display error, shutdown axis if data is invalid
MG "SERERR ", _SSA{$4};'Display hex value of SS bit mask
MG "TC: ", _TC;'Display any TC error codes
bisscode = _SSA;'Query SS bit mask
IF (bisscode & $1);'First bit in the mask is the Timeout bit
  MG "BiSS Timeout";
  'Add timeout code here
  SSA=0;'Turn off BiSS encoder
ENDIF
IF (bisscode & $2);'Second bit in the mask is the CRC validity bit
  MG "Invalid CRC"
  'Add CRC failure code here, make the system safe
  SSA=0;'Turn off BiSS
  CB 1;'Turn off output 1
  CB 3;'Turn off output 3
  AB;'Abort motion and program operation
ENDIF
IF (bisscode & $4);'Third bit in the mask is the error bit
  MG "Error Bit Set";
  'Add Error code here, make the system safe
  SSA=0;'Turn off BiSS
  CB 1;'Turn off output 1
  CB 3;'Turn off output 3
  AB;'Abort motion and program operation
ENDIF
IF (bisscode & $8);'Fourth bit in the mask is the warning bit
```

```
MG "warning Bit Set";  
'Add warning code here, Notify and log but don't stop motion  
XQ#logwarn,3;'Run routine to Log the warning and notify user  
ENDIF  
RE
```

**SY applies to SER**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@gallmc.com](mailto:documentation@gallmc.com)

**TB** *Tell Status Byte*

|                 |     |                                          |
|-----------------|-----|------------------------------------------|
| <b>Usage</b>    | TB  | Command takes no arguments               |
| <b>Operands</b> | _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**

TB is an interrogation command with no parameters

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 | N/A                               |
| Bit 5 | N/A                               |
| Bit 4 | N/A                               |
| Bit 3 | Input Interrupt enabled           |
| Bit 2 | Executing input interrupt routine |
| Bit 1 | 0 (Reserved)                      |
| Bit 0 | Echo on                           |

**Remarks**

- \_TB Contains the status byte reported by the TB command

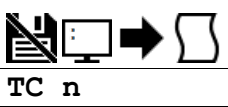
**Examples**

```
'Galil DMC Code Example
:TB:      Tell status information
129'      Executing program and echo on (2^7 + 2^0 = 128 + 1 = 129)
```

**TB applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,DMC18x6,DMC18x2,RIO47xxx,EDD3701x**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)



**TC Tell Error Code**

|                 |          |                                                             |
|-----------------|----------|-------------------------------------------------------------|
| <b>Usage</b>    | TC n ... | Arguments specified with an implicit, comma-separated order |
| <b>Operands</b> | _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                              |          |
| 9                | Variable error                                   |          |
| 10               | Empty program line or undefined label            |          |
| 11               | Invalid label or line number                     |          |
| 12               | Subroutine more than 16 deep                     |          |
| 14               | EEPROM check sum error                           |          |
| 15               | EEPROM write error                               |          |
| 17               | ED, BN and DL not valid while program running    |          |
| 19               | Application strand already executing             |          |
| 25               | Variable not given in IN command                 |          |
| 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                                |          |
| 97               | Bad Binary Command Format                        |          |
| 98               | Binary Commands not valid in application program |          |
| 99               | Bad binary command number                        |          |
| 106              | Privilege Violation                              |          |
| 120              | Bad Ethernet transmit                            |          |
| 121              | Bad Ethernet packet received                     |          |

|     |                                                    |                                  |
|-----|----------------------------------------------------|----------------------------------|
| 123 | TCP lost sync                                      |                                  |
| 124 | Ethernet handle already in use                     |                                  |
| 125 | No ARP response from IP address                    |                                  |
| 126 | Closed Ethernet handle                             |                                  |
| 127 | Illegal Modbus function code                       |                                  |
| 128 | IP address not valid                               |                                  |
| 130 | Remote IO command error                            |                                  |
| 131 | Serial Port Timeout                                | See Remarks                      |
| 132 | Analog inputs not present                          |                                  |
| 133 | Command not valid when locked / Handle must be UDP |                                  |
| 136 | Invalid Password                                   |                                  |
| 137 | Invalid lock setting                               |                                  |
| 138 | Passwords not identical                            |                                  |
| 140 | Serial encoder missing                             | Valid for BiSS support           |
| 141 | Incorrect ICM Configuration                        |                                  |
| 144 | Incompatible with encoder type                     | Valid on RIO with encoder inputs |

## 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.
- Note: Error code 131 means that an RS232/USB timeout is being generated while trying to transmit data to the serial port.
  - This is usually caused by MG. Numerous timeouts on serial communication can cause a slowdown in DMC code execution and should be avoided.

## Examples

```
'Galil DMC Code Example
:GF32;' Bad command
?
:TC1;' Tell error code
1 Unrecognized command
```

### TC applies to

**DMC500x0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2,DMC52xx0,RIO574x0,EDD3701x**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**TE** *Tell Error***TE** mm

|                 |       |                                          |
|-----------------|-------|------------------------------------------|
| <b>Usage</b>    | TE mm | Argument is an axis mask                 |
| <b>Operands</b> | _TEm  | Operand has special meaning, see Remarks |

**Description**

The TE command returns the current error in the control loop.

The error is the difference of the analog signal in the control loop from the setpoint. For further details see the Process Control section in the User's Manual.

**Arguments**

| Argument | Min | Max    | Default | Resolution      | Description                   | Notes                             |
|----------|-----|--------|---------|-----------------|-------------------------------|-----------------------------------|
| mm       | A   | ABCDEF | ABCDEF  | Multi-Axis Mask | Process loops to report error | Valid for RIO-47xx2 and RIO-473xx |
|          | A   | AB     | AB      | Multi-Axis Mask | Process loops to report error | Valid for RIO-47xx0               |

**Remarks****Operand Usage**

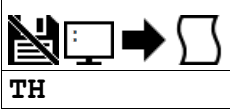
- \_TEm contains the current position error value for the specified process loop.

**Examples**

```
'Galil DMC Code Example
:TE;'          Return all position errors
1,-2
:Error =_TEA;' Sets the variable, Error
'with the channel A position error
```

**TE applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,DMC18x6,DMC18x2,RIO47xxx,EDD3701x**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**TH** *Tell Ethernet Handle*

TH

|              |    |                            |
|--------------|----|----------------------------|
| <b>Usage</b> | TH | Command takes no arguments |
|--------------|----|----------------------------|

**Description**

The TH command returns a list of data pertaining to the Galil's Ethernet connection. This list begins with the IP address and Ethernet address (physical address), followed by the status of each handle indicating connection type and IP address.

**Arguments**

TH is an interrogation command with no parameters

**Remarks**

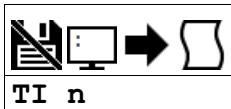
- If no handles are shown as *AVAILABLE*, the controller will be unable to create or accept more Ethernet connections with TCP or UDP. Ping will still function when all handles are taken.

**Examples**

```
'Galil DMC Code Example
:TH;' show the status of RIO-47xx0 handles
CONTROLLER IP ADDRESS 10,0,51,82 ETHERNET ADDRESS 10-80-3C-10-01-2F
IHA TCP PORT 1010 TO IP ADDRESS 10,0,51,87 PORT 1030
IHB TCP PORT 1020 TO IP ADDRESS 10,0,51,87 PORT 1070
IHC AVAILABLE
:'note that the RIO-47xx2 and RIO-473xx have 5 available ethernet handles
```

**TH applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**TI Tell Inputs**

TI n

| Usage           | TI n ...                                     | Arguments specified with an implicit, comma-separated order |
|-----------------|----------------------------------------------|-------------------------------------------------------------|
| <b>Operands</b> | _TI0<br>_TI1<br>_TI2<br>_TI3<br>_TI4<br>_TI5 | 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 0-7   | Default                       |
|          | 1     | Report status of Inputs 8-15  |                               |
|          | 2     | Report status of Inputs 16-23 | RIO-473xx                     |
|          | 3     | Report status of Inputs 24-31 | RIO-473xx with -24ExIn option |
|          | 4     | Report status of Inputs 32-39 | RIO-473xx with -24ExIn option |
|          | 5     | Report status of Inputs 40-47 | RIO-473xx with -24ExIn option |

**Remarks****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.

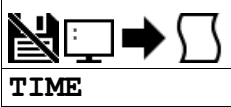
**Examples**

```
'Galil DMC Code Example
:TI1;'      Tell input state on bank 1
8          Bit 3 is high, others low
:TI0
0          All inputs on bank 0 low
:Input=_TI1;'  Sets the variable, Input, with the TI1 value
:Input=?
8.0000
```

**TI applies to**

**DMC500x0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2,RIO574x0,DMC52xx0,EDD3701x**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**TIME** *Time Operand***TIME**

|                 |                |                                          |
|-----------------|----------------|------------------------------------------|
| <b>Usage</b>    | variable= TIME | Holds a value                            |
| <b>Operands</b> | TIME           | Operand has special meaning, see Remarks |

**Description**

The TIME operand returns the value of the internal free running, real time clock.

The operand TIME will increase by 1 count every millisecond. The clock is reset to 0 with a standard reset or a master reset.

**Arguments**

TIME is an operand and has no parameters

**Remarks**

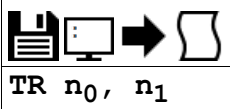
- The keyword, TIME, does not require an underscore ( \_ ) as with the other operands.
- TIME will increment up to +2,147,483,647 before rolling over to -2,147,483,648 and continuing to count up.
  - TIME rollover occurs after ~24-25 days of on-time with no reset.

**Examples**

```
'Galil DMC Code Example
MG TIME;'  Display the value of the internal clock
t1=TIME;'  Sets the variable t1 to the TIME value
```

**TIME applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,DMC18x6,DMC18x2,RIO47xxx,EDD3701x**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@gallmc.com](mailto:documentation@gallmc.com)

**TR** *Trace*TR n<sub>0</sub>, n<sub>1</sub>

|              |          |                                                             |
|--------------|----------|-------------------------------------------------------------|
| <b>Usage</b> | 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 <sub>0</sub> | 0   | 1   | 0       | 1          | Set status of trace function    | n <sub>0</sub> = 0 or null disables Trace. n <sub>0</sub> = 1 enables trace. |
| n <sub>1</sub> | 0   | 15  | 15      | 1          | Set threads to trace by bitmask | See Remarks                                                                  |

**Remarks**

- Note: The lines following the Line Continuation Character ( ` ) will not be displayed in the trace output.

```
'Galil DMC Code Example
#A
a=123`
456;'not displayed with TR1 output
EN
```

**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

:TR1
:
2 JP0
0 MGTIME
18006461.0000
1 WT1000
2 JP0
0 MGTIME
18007461.0000
1 WT1000

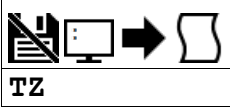
:TR0
:
18008461.0000
18009461.0000

:ST
```

**TR applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

## TZ Tell I/O Configuration



|              |    |                            |
|--------------|----|----------------------------|
| <b>Usage</b> | TZ | Command takes no arguments |
|--------------|----|----------------------------|

### Description

The TZ command is used to request the I/O status of the controller. This is returned to the user as a human-readable text string.

On each line, the information starts with the I/O block, the corresponding I/O number range, the configuration (as inputs or outputs), and the values (255 for all 8 output bits high etc.)

### Arguments

TZ is an interrogation command with no parameters

### Remarks

- The data reported by TZ is also accessible through the TI (inputs) and OP (outputs) command
- For the RIO, the analog input and output data is all queryable through the @AN[] and @AO[] command, respectively

### Examples

```
'Galil DMC Code Example
:TZ; 'issued for RIO-471xx and RIO-472xx
Block 0 (7-0) Inputs - value 255 (1111_1111)
Block 1 (15-8) Inputs - value 255 (1111_1111)
Block 0 (7-0) Outputs - value 0 (0000_0000)
Block 1 (15-8) Outputs - value 0 (0000_0000)
Analog Inputs(7-0) 0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000
Analog Outputs(7-0) 0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000
```

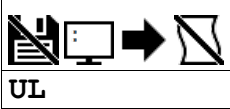
```
'Galil DMC Code Example
:TZ; 'issued for RIO-473xx
Block 0 (7-0) Inputs - value 255 (1111_1111)
Block 1 (15-8) Inputs - value 255 (1111_1111)
Block 2 (23-16) Inputs - value 255 (1111_1111)
Block 0 (7-0) Outputs - value 0 (0000_0000)
Block 1 (15-8) Outputs - value 0 (0000_0000)
Block 2 (23-16) Outputs - value 0 (0000_0000)
Analog Inputs(7-0) 0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000
Analog Outputs(7-0) 0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000
```

**TZ applies to DMC40x0,DMC42x0,DMC41x3,DMC21x3,RIO47xxx,DMC500x0,DMC52xx0**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)



## UL *Upload*



UL

|                 |     |                                          |
|-----------------|-----|------------------------------------------|
| <b>Usage</b>    | UL  | Command takes no arguments               |
| <b>Operands</b> | _UL | Operand has special meaning, see Remarks |

### Description

The UL command transfers the program 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 a 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 Telnnet, the UL command will bring the uploaded program to screen.
- From there, the user can copy it and save it to a file.
- Issuing this command will pause the output of the Data Record until the command is completed.

### Operand Usage

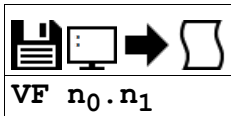
- When used as an operand, \_UL gives the number of available variables.

### Examples

```
'Galil DMC Code Example
:UL;'      Begin upload
#A;'      Line 0
SB1;'     This is an Example Line 1
CB1;'     Program Line 2
EN;'     Line 3
{cntrl}Z Terminator
```

**UL applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**VF Variable Format**VF n<sub>0</sub>.n<sub>1</sub>

|                 |          |                                                             |
|-----------------|----------|-------------------------------------------------------------|
| <b>Usage</b>    | VF n ... | Arguments specified with an implicit, comma-separated order |
| <b>Operands</b> | _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 <sub>0</sub> | -8  | 10  | 10      | 1          | Specify the number of digits displayed before the decimal point | A negative value specifies hexadecimal format, see Remarks |
| n <sub>1</sub> | 0   | 4   | 4       | 1          | Specify the number of digits displayed after the decimal point  |                                                            |

**Remarks**

- A negative n<sub>0</sub> 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<sub>0</sub> specifies standard decimal format.
- A ? is only valid for querying n<sub>0</sub>. When queried, the value reported will be the value of the format for variables and arrays specified by n<sub>0</sub> and n<sub>1</sub>
  - 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<sub>0</sub> is insufficient for representing the integer portion of a variable, the returned value will be the greatest number representable by n<sub>0</sub>.n<sub>1</sub>. For example, if *var=123*, and VF is 2.4, *var=?* will return 99.9999.

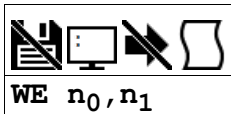
**Examples**

```
'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
:VF8,4;' 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
```

**VF applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**WE** Write EncoderWE n<sub>0</sub>, n<sub>1</sub>

|              |          |                                                             |
|--------------|----------|-------------------------------------------------------------|
| <b>Usage</b> | WE n ... | Arguments specified with an implicit, comma-separated order |
|--------------|----------|-------------------------------------------------------------|

**Description**

The WE command sets the current encoder position to a specified value. The units are in quadrature counts. This command will set the QE values.

**Arguments**

| Argument       | Min            | Max           | Default | Resolution | Description                     | Notes |
|----------------|----------------|---------------|---------|------------|---------------------------------|-------|
| n <sub>0</sub> | -2,147,483,648 | 2,147,483,647 | N/A     | 1          | Encoder value set for channel 0 |       |
| n <sub>1</sub> | -2,147,483,648 | 2,147,483,647 | N/A     | 1          | Encoder value set for channel 1 |       |

**Remarks**

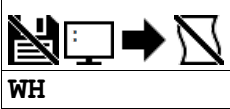
- A RIO with the QUAD option and -ser firmware are required to use this command.
- This command is not valid with SSI or BISS encoder channels.

**Examples**

```
'Galil DMC Code Example
:WE 1000,10000;'      Set encoder channel 0 to 1000, channel 1 to 10000
```

**WE applies to RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**WH** *Which Handle*

WH

|                 |     |                                          |
|-----------------|-----|------------------------------------------|
| <b>Usage</b>    | WH  | Command takes no arguments               |
| <b>Operands</b> | _WH | Operand has special meaning, see Remarks |

**Description**

The WH command is used to identify the handle from which the command was received. This is useful for determining what interface or handle you are connected to.

**Arguments**

WH is an interrogation command with no parameters

**Remarks**

- \_WH contains the numeric representation of the handle from which the command was received.
- The following table lists the possible string returned by WH, and the numerical value returned by \_WH

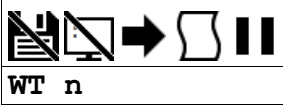
| Communication Channel | WH    | _WH | Notes                             |
|-----------------------|-------|-----|-----------------------------------|
| Main Serial Port      | RS232 | -1  |                                   |
| Ethernet Handle A     | IHA   | 0   |                                   |
| Ethernet Handle B     | IHB   | 1   |                                   |
| Ethernet Handle C     | IHC   | 2   |                                   |
| Ethernet Handle D     | IHD   | 3   | Valid for RIO-47xx2 and RIO-473xx |
| Ethernet Handle E     | IHE   | 4   | Valid for RIO-47xx2 and RIO-473xx |

**Examples**

```
'Galil DMC Code Example
:WH;' Request incoming handle identification
IHC
:MG_WH
```

**WH applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**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 amount 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        | 1   | 2,147,483,646 | N/A     | 1          | Number of milliseconds to hold execution of code. |       |

**Remarks**

- None

**Examples**

```
'Galil DMC Code Example
REM 2 seconds after input 1 goes low, turn on output 1 for 3 seconds
#main
AI -1; 'wait for input 1 to go low
WT 2000; 'wait 2 sec
SB 1; 'set output 1
WT 3000; 'wait 3 sec
CB 1; 'clear output 1
JP#main; 'start over again
```

**WT applies to DMC500x0, DMC52xx0, DMC40x0, DMC42x0, DMC41x3, DMC30010, DMC21x3, RIO47xxx, DMC18x6, DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**XQ** *Execute Program*

**XQ** #str, n<sub>1</sub>

**XQ** n<sub>0</sub>, n<sub>1</sub>

|                 |                              |                                                             |
|-----------------|------------------------------|-------------------------------------------------------------|
| <b>Usage</b>    | XQ n ...                     | Arguments specified with an implicit, comma-separated order |
| <b>Operands</b> | _XQ0<br>_XQ1<br>_XQ2<br>_XQ3 | 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 4 programs may be executed simultaneously to perform multitasking.

**Arguments**

| Argument             | Min    | Max     | Default   | Resolution | Description                         | Notes                                             |
|----------------------|--------|---------|-----------|------------|-------------------------------------|---------------------------------------------------|
| <b>str</b>           | 1 char | 7 chars | See Notes | String     | Label to begin code execution       | If omitted, start from line 0 (n <sub>0</sub> =0) |
| <b>n<sub>0</sub></b> | 0      | 199     | 0         | 1          | Line number to begin code execution | For RIO-47xx0                                     |
|                      | 0      | 399     | 0         | 1          | Line number to begin code execution | For RIO-47xx2 and RIO-473xx                       |
| <b>n<sub>1</sub></b> | 0      | 3       | 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
```

**XQ applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**ZC** *User Variable, ZC*

ZC n

|                 |          |                                                             |
|-----------------|----------|-------------------------------------------------------------|
| <b>Usage</b>    | ZC n ... | Arguments specified with an implicit, comma-separated order |
| <b>Operands</b> | _ZC      | Operand holds the value last set by the command             |

**Description**

ZC sets the first user variable. This variable provides a method for specific board information to be passed using the data record.

**Arguments**

| Argument | Min            | Max           | Default | Resolution | Description                                      | Notes |
|----------|----------------|---------------|---------|------------|--------------------------------------------------|-------|
| n        | -2,147,483,648 | 2,147,483,647 | 0       | 1          | Value of first User variable for the data record |       |

**Remarks**

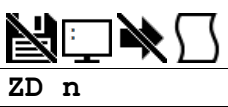
- n is an integer and can be a number, controller operand, variable, mathematical function, or string.
- Only 4 bytes are available for n. Fractional values are not stored or sent via the data record

**Examples**

```
'Galil DMC Code Example
ZC 2343;' Sets the first user variable to a number (2343)
ZC "YO";' Sets the first user variable to a string (YO)
```

**ZC applies to RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)

**ZD** *User Variable, ZD*

ZD n

|                 |          |                                                             |
|-----------------|----------|-------------------------------------------------------------|
| <b>Usage</b>    | ZD n ... | Arguments specified with an implicit, comma-separated order |
| <b>Operands</b> | _ZD      | Operand holds the value last set by the command             |

**Description**

ZD sets the second user variable. This variable provides a method for specific board information to be passed using the data record.

**Arguments**

| Argument | Min            | Max           | Default | Resolution | Description                                       | Notes |
|----------|----------------|---------------|---------|------------|---------------------------------------------------|-------|
| n        | -2,147,483,648 | 2,147,483,647 | 0       | 1          | Value of second user variable for the 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

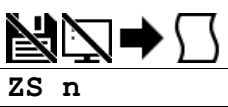
**Examples**

```
'Galil DMC Code Example
ZD "INPT";' Sets the second user variable to the string "INPT"
ZD 1234567;' Sets the second user variable to the number 1,234,567
```

**ZD applies to RIO47xxx**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)



**ZS** Zero Subroutine Stack

|                 |                              |                                                             |
|-----------------|------------------------------|-------------------------------------------------------------|
| <b>Usage</b>    | ZS n ...                     | Arguments specified with an implicit, comma-separated order |
| <b>Operands</b> | _ZS0<br>_ZS1<br>_ZS2<br>_ZS3 | 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 3.
  - The response, an integer between zero and sixteen, indicates zero for beginning condition and sixteen for the deepest value.

**Examples**

```
'Galil DMC Code Example
II0,1,7;' Input Interrupt on 7
#A,JP #A;EN;' Main program
#ININT0;' Input Interrupt
MG "INTERRUPT";' Print message
S=_ZS0;' Interrogate stack before ZS
S=?;' Print stack
ZS;' Zero stack
S=_ZS0;' Interrogate stack after ZS
S=?;' Print stack
EN
```

**ZS applies to DMC500x0,DMC52xx0,DMC40x0,DMC42x0,DMC41x3,DMC30010,DMC21x3,RIO47xxx,DMC18x6,DMC18x2**

©2018 Galil Motion Control. Revision: 1752 . Corrections, Feedback: [documentation@galilmc.com](mailto:documentation@galilmc.com)