Galil Motion Control

HARDWARE COMMAND REFERENCE

Edition 5/19/2010 5:33:13 PM (svn 210)

- Hardware Referenced:
  - All
  - RIO
- Command Details included for:
  - All

  - RIO

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

Table of Contents

#

Label (subroutine)

Full Description

The # operator denotes the name of a program label (for example #Move). Labels can be up to seven characters long and are often used to implement subroutines or loops. Labels are divided into (a) user defined (b) automatic subroutines. User defined labels can be printed with LL and the number of labels left available can be queried with MG _DL. The automatic subroutines include #CMDERR, #LIMSWI, #POSERR, #ININT, #AUTO, #AUTOERR, and #MCTIME (no RIO).
A label can only be defined at the beginning of a new line.

There is a maximum of 62 labels available on the RIO-47xx0
There is a maximum of 126 labels available on the RIO-47xx2

Arguments

#nnnnnnn where
nnnnnnn is a label name up to seven characters. Uppercase or lowercase characters are valid.

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (no RIO)	Yes
In a Program	Yes
Command Line	No
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

LL - List labels
_DL - Labels available
JP - Jump statement
JS - Jump subroutine

Examples:

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

#AUTO

Subroutine to run automatically upon power up

Full Description

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

Upon controller startup, application code will automatically begin running in thread 0 at #AUTO.

Use EN to end the routine.

Arguments

N/A

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving	Yes
In a Program	Yes
Command Line	No
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

BP - Burn program
EN - End program
#AUTOERR - Automatic Subroutine for EEPROM error

Examples:

```
'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
 SB1;'     Set bit 1
 WT500;'   Wait 500msec
 CB1;'     Clear bit 1
 WT500;'   Wait 500msec
JP#AUTO;'  Jump back to #AUTO
```

#AUTOERR

Automatic subroutine for notification of EEPROM checksum errors

Full Description

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

Use EN to end the routine.

Arguments

N/A

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving	Yes
In a Program	Yes
Command Line	No
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

_RS - Checksum error code operand
EN - End program

Examples:

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

#CMDERR

Command error automatic subroutine

Full Description

#CMDERR is an 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 (all threads) will stop.

Use EN to end the routine.

#CMDERR will only run from errors generated within embedded DMC code.

Arguments

N/A

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving	Yes
In a Program	Yes
Command Line	No
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

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

Examples:

```
'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
```

#COMINT

Communication interrupt automatic subroutine

Full Description

#COMINT is an automatic subroutine which 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.

#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. Use EN to end the routine

Arguments

N/A

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving	Yes
In a Program	Yes
Command Line	No
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

P1CD - Serial port code
P1CH - Serial port single character
P1NM - Serial port number
P1ST - Serial port string (maximum length of 6 characters)
CI - Configure #COMINT (and set operator data entry mode)
EN - End subroutine

Examples:

```
CI2;'                  interrupt on any character
#Loop
MG "Loop";'            print a message every second
WT 1000
JP#Loop
#COMINT
MG "COMINT=",P1CH;'    print character received
EN1,1
NOTE: An application program must be executing for the automatic subroutine to
function, which
runs in thread 0. Use EN to end the routine.
```

#TCPERR

Ethernet communication error automatic subroutine

Full Description

The following error (see TC) occurs when a command such as MG "hello" {EA} is sent to a failed Ethernet connection:

123 TCP lost sync or timeout

This error means that the client on handle A did not respond with a TCP acknowledgement (for example because the Ethernet cable was disconnected). Handle A is closed in this case.

#TCPERR allows the application programmer to run code (for example to reestablish the connection) when error 123 occurs.

Arguments

N/A

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving	Yes
In a Program	Yes
Command Line	No
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

TC - Tell error code
_IA4 - Last dropped handle
MG - Print message
SA - Send ASCII command via Ethernet

Examples:

```
#L
 MG {EA} "L"
 WT1000
JP#L
#TCPERR
 MG {P1} "TCPERR.  Dropped handle", _IA4
RE
'NOTE: Use RE to end the routine
```

$

Hexadecimal

Full Description

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

Arguments

$nnnnnnnn.mmmm
n is up to eight hexadecimal digits (denoting 32 bits of integer)
m is up to four hexadecimal digits (denoting 16 bits of fraction)

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

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

Examples:

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

& , |

Bitwise Logical Operators AND and OR

Full Description

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

Arguments

n & m or n | m where
n and m are signed numbers in the range -2147483648 to 2147483647.
For IF, JP, and JS, n and m are typically the results of logical expressions such as (x > 2) & (y=8)

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

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

Examples:

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

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

(,)

Parentheses (order of operations)

Full Description

The parentheses denote the order of math and logical operations. Note that the controller DOES NOT OBEY STANDARD MATHEMATICAL OPERATOR PRECEDENCE. For example, multiplication is NOT evaluated before addition. Instead, the controller follows left-to-right precedence. Therefore, it is required to use parentheticals to ensure intended precedence.

Arguments

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

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

+ - * / - Math Operators
& \| - Logical Operators

Examples:

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

;

Semicolon (Command Delimiter)

Full Description

The semicolon operator allows multiple Galil commands to exist on a single line. It is used for the following three reasons:
(1) To put comments on the same line as the command (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.

Arguments

n;n;n;n
where
n is a valid Galil command

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

NO - No Op, comment
' - comment

Examples:

```
SB1;WT500;CB1;'multiple commands separated by semicolons with a comment
```

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

#Low;'      #Low when run in parallel
 c = c + 1
 d = d + 1
JP#Low
```

@ABS

Absolute value

Full Description

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

Arguments

@ABS[n] where
n is a signed number in the range -2147483647 to 2147483647

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

All math operators

Examples:

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

@ACOS

Inverse cosine

Full Description

Returns in degrees the arc cosine of the given number.

Arguments

@ACOS[n] where
n is a signed number in the range -1 to 1.

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

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

Examples:

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

@AN

Analog Input Query

Full Description

Returns the value of the given analog input in volts

Arguments

@AN[n] where n is the input number assigned to a particular analog input pin (0-7)

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

While Moving Yes Default Value -
In a Program Yes Default Format -
Command Line Yes

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

Related Commands

AQ Analog Range
AO Analog Output

Examples:

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

@AO

Analog Output Query

Full Description

The @AO[n] command 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

@AO[n] where
n is the I/O number assigned to a particular analog output pin (0-7)

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

AO Analog Output
DQ Set analog output range

Examples:

```
MG@AO[0]       Displays status of Analog output 0
Temp=@AO[0]     Sets variable Temp to the value of Analog output 0
```

@ASIN

Inverse sine

Full Description

Returns in degrees the arc sine of the given number.

Arguments

@ASIN[n] where
n is a signed number in the range -1 to 1.

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

@ACOS[n] - Arc cosine
@SIN[n] - sine
@ATAN[n] - Arc tangent
@COS[n] - Cosine
@TAN[n] - Tangent

Examples:

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

@ATAN

Inverse tangent

Full Description

Returns in degrees the arc tangent of the given number.

Arguments

@ATAN[n]
n is a signed number in the range -2147483647 to 2147483647

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

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

Examples:

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

@COM

Bitwise complement

Full Description

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

Arguments

@COM[n] where
n is a signed integer in the range -2147483647 to 2147483647.
The integer is interpreted as a 32-bit field.

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

& | - Logical operators AND and OR

Examples:

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

@COS

Cosine

Full Description

Returns the cosine of the given angle in degrees

Arguments

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

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

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

Examples:

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

@FRAC

Fractional part

Full Description

Returns the fractional part of the given number

Arguments

@FRAC[n], n is a signed number in the range -2147483648 to 2147483647.

Operand Usage

N/A

Usage

Usage and Default Detail

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

@INT[n] - Integer part

Examples:

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

@IN

Read digital input

Full Description

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

Arguments

@IN[n] where
n is an unsigned integer in the range 0 to 15

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

@AN[n] - Read analog input
@OUT[n] - Read digital output
SB - Set digital output bit
CB - Clear digital output bit

AO - set analog output

Examples:

```
MG @IN[1]
:1.0000
x = @IN[1]
x = ?
:1.000  print digital input 1
```

@INT

Integer part

Full Description

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

Arguments

@INT[n]
n is a signed number in the range -2147483648 to 2147483647.

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

@FRAC - Fractional part

Examples:

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

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

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

@OUT

Read digital output

Full Description

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

Arguments

@OUT[n] where
n is an unsigned integer in the range 0 to 15

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

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

Examples:

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

@RND

Round

Full Description

Rounds the given number to the nearest integer

Arguments

@RND[n]
n is a signed number in the range -2147483648 to 2147483647.

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

@INT[n] - Truncates to the nearest integer

Examples:

```
:MG @RND[1.2]
 1.0000
:MG @RND[5.7]
 6.0000
:MG @RND[-1.2]
 -1.0000
:MG @RND[-5.7]
 -6.0000
:MG @RND[5.5]
 6.0000
:MG @RND[-5.5]
 -5.0000
```

@SIN

Sine

Full Description

Returns the sine of the given angle in degrees

Arguments

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

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

@ASIN[n] - Arc sine
@COS[n] - Cosine
@ATAN[n] - Arc tangent
@ACOS[n] - Arc cosine
@TAN[n] - Tangent

Examples:

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

@SQR

Square Root

Full Description

Takes the square root of the given number. If the number is negative, the absolute value is taken first.

Arguments

@SQR[n] where
n is a signed number in the range -2147483648 to 2147483647.

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

@ABS[n] - Absolute value

Examples:

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

[,]

Square Brackets (Array Index Operator)

Full Description

The square brackets are used to denote the array index for an array, or to denote an array name. (They are also used to designate the argument to a function, such as @ABS[n].)

Arguments

mmmmmmmm[n] where

mmmmmmmm is the array name
n is the array index and is an integer between 0 and 399 for the RIO-47xx0 (0-999 on the RIO-47xx2)

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

DM - Dimension Array
QU - Print/Upload Array

Examples:

```
DM A[50]        ;'define a 50 element array
A[0] = 3        ;'set first element to 3
MG A[0]         ;'print element 0
```

^L^K

Lock program

Full Description

<control>L<control>K locks user access to the application program. When locked, the ED, UL, LS, and TR commands will give privilege error #106. The application program will still run when locked.
The locked or unlocked state can be saved with a BN command. Upon master reset, the controller is unlocked. Once the program is unlocked, it will remain accessible until a lock command or a reset (with the locked condition burned in) occurs.

Arguments

<control>L<control>Kpassword,n where
When n is 1, this command will lock the application program.
When n is 0, the program will be unlocked.

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	No
Command Line	Yes
Controller Usage	DMC-40x0, DMC-18x6, RIO-47xxx
Default Value	N/A
Default Format	N/A

Related Commands

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

Examples:

```
:PWtest,test             Set password to "test"
:^L^K test,1             Lock the program
:LS                      Attempt to list the program
?
:TC1
106 Privilege violation
:
```

^R^S

Master Reset

Full Description

The Master Reset command resets the RIO 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.

Note: 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.

Arguments

Operand Usage

N/A

Usage

Usage and Defalut Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	No
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Formula	N/A

Related Commands

RS - Reset

Examples:

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

:KP?
 6.00
:KP10
:BN
:RS

:KP?
 10.00
:^R^S

:KP?
 6.00
:
```

^R^V

Revision Information

Full Description

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

Arguments

N/A

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving	Yes
In a Program	No
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

Examples:

```
N/A
```

`

Line Continuation Character

Full Description

(ASCII 96)
Allows a command in an application program to extend beyond the confines of the maximum line length of 40 characters. This is especially useful for code compression, long MG statements, or multiple conditions in an IF,JP or JS statement.
Note: 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.

Note: The lines following the Line Continuation Character (`) will not be displayed in the trace output (TR1).

```
#A
a=123`
456;'not displayed with TR1 output
EN
```

Arguments

none

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	No
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

MG - Print Message

Examples:

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

+,-,*, /,%

Math Operators

Full Description

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

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

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

Arguments

(n + m) or (n - m) or (n * m) or (n / m) or (n % m) where
n and m are signed numbers in the range -2147483648 to 2147483647

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

( ) - Parenthesis

Examples:

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

, =, <=, >=, <>

Comparison Operators

Full Description

The comparison operators are as follows:
< less than
> greater than
= equals
<= less than or equal
>= greater than or equal
<> not equals
These are used in conjunction with IF, JP, JS, ( ), &, and | to perform conditional jumps. The result of a comparison expression can also be printed with MG or assigned to a variable.

Arguments

(n < m) or (n > m) or (n = m) or (n <= m) or (n >= m) or (n <> m) where
n and m are signed numbers in the range -2147483648 to 2147483647

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

( ) - Parentheses
IF - If statement
JP - Jump
JS - Jump subroutine

Examples:

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

=

Equals (Assignment Operator)

Full Description

The assignment operator is used for three reasons:
(1) to define and initialize a variable (x = 0) before it is used
(2) to assign a new value to a variable (x = 5)
(3) to print a variable or array element (x= which is equivalent to MG x). MG is the preferred method of printing.

Arguments

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

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

MG - Print Message

Examples:

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

AA

After Analog Trippoint

Full Description

AA is a trippoint that halts program execution until a voltage on a particular analog input has been reached. The range of AA is dependant on RIO hardware. See the AQ command for more info.

Arguments

AA m,n,o where
m is the analog input (0-7)
n = the voltage (range is 0-5V for RIO-47x0x)
(range dependent on AQ for RIO-47x2x)
o = 0 for Trippoint to clear when voltage is greater than n
= 1 for Trippoint to clear when voltage is less than n

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	No
Controller Usage	All
Default Value	N/A
Default Format	N/A

In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand No

N/A

Related Commands

@AN[x] - Function to query the voltage on an analog input
AQ - Set analog input ranges

Examples:

```
#A
REM Wait for analog input 3 to go above 2.5V
AA 3,2.5,0;
MG "Analog input 3 reached 2.5V";
EN
```

AB

Abort

Full Description

AB (Abort) aborts the application programs including any threads

Arguments

AB

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

In a Program Yes
Command Line Yes

Related Commands

HX - Halt Execution
XQ - Execute Program

Examples:

```
AB    ;Abort application program
```

AF

Analog Feedback Select

Full Description

The Analog Feedback (AF) command is used to select which analog input channel will be used for feedback if a control loop is desired. Note - The RIO-472xx does not have any control loops.

Arguments

AF n,n or AFm=n where
n = 0-7 selects the analog input channel for feedback. Set n=-1 to disable control loop
n = ? Returns the input channel m=A,B
RIO-47xx2 has 6 control loops AFn,n,n,n,n,n and m=A,B,C,D,E,F

Operand Usage

_AFm contains the analog input channel selected for the specified control loop

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	No
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	0,0,0,0 or -1,-1
Default Format	N/A

Related Commands

AQ - Analog Configuration
AZ - Analog Output Select
PS - Control Loop Setpoint

Examples:

```
AF0
AZ0
KP1
KD10
KI0.5
PS2.5
```

AI

After Input

Full 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.

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

Arguments

AI m1& m2& m3& m4 where
mn is an integer in the range 0 to 15 decimal. When mn is positive, the RIO will wait for the input to be high. When mn is negative, the RIO will wait for the input to be low.

Usage

Usage and Default Details

Usage	Value
While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

Operand Usage

N/A

Related Commands

@IN[n] - Read Digital Input
II - Input interrupt
#ININT - Label for input interrupt
TI - Tell Inputs

IQ Configure Inputs

Examples:

```
#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
```

AO

Analog Output

Full Description

The AO command sets the analog output voltage on an analog output pin. It is also used to set the analog output voltage on ModBus devices over Ethernet. For RIO-4712x models, the output voltage range can be adjusted using the DQ command. Use the ID command to check your model number and I/O configuration. Note: The RIO-472xx does not have any analog outputs by default.

Arguments

AO m, n where
m is the I/O number assigned to a particular analog output pin (0-7)

m can also be the I/O number of a ModBus device that is calculated using the following equations:
m = (HandleNum*1000) + ((Module-1)*4) + (Bitnum-1)
HandleNum is the handle specifier from A to C (1 to 3).
Module is the position of the module in the rack from 1 to 16.
BitNum is the I/O point in the module from 1 to 4.

n = analog output voltage. Default is 12bit resolution. (RIO-4712x-16 is 16bit resolution)

Usage

Usage and Default Details

Usage	Value
While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

Operand Usage

N/A

Related Commands

SB - Set Bit
CB - Clear Bit
MB - Modbus

DQ - Set analog output range
@AO[x] - Report back analog output value

Examples:

```
AO1,1.324;          Outputs 1.324 Volts on Channel 1
```

AQ

Analog Input Configuration

Full Description

The Analog Configuration (AQ) command is used to set the type of analog input - single ended (default) or differential.

For the RIO-4712x or RIO-472xx (with +/-10V analog inputs):
The AQ command also sets the range of the analog inputs. There are 4 different ranges that each analog input may be assigned.

Setting a negative range for inputs 0,2,4 or 6, configures those inputs as differential inputs, with the compliment input 1,3,5 and 7 respectively.

Note: Default resolution for analog inputs is 12bits. The RIO-4712x-16bit and RIO-472xx-(16bit) have 16bit resolution.

Arguments

For the RIO-47100 or RIO-472xx:
AQ n,m where
n is an even integer from 0-6 that represents the analog input channel
m = 0 (default) single ended inputs
m = 1 differential inputs (next odd channel is consumed)

For the RIO-4712x or RIO-472xx (with +/-10V analog inputs)
AQ n,m where
n is an integer from 0-7 that represents the analog input channel
m is an integer from 1-4 that designates the analog range

m	Analog Range
1	+/-5V
2	+/-10V (default for RIO-4712x)
3	0-5V
4	0-10V

Setting a negative range for inputs 0,2,4 or 6, configures those inputs as differential inputs, with the compliment input 1,3,5 and 7 respectively.

Usage

Usage and Default Details

Usage	Value
While Moving (no RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	DMC-40x0, DMC-21x3, RIO-47xxx
Default Value	n,2
Default Format	1.0000

Operand Usage

_AQn holds the range setting for that axis where n=0-7

Related Commands

@AN[n] - Read Analog Input
AQ - Analog Input Configuration

Examples:

```
:AQ 2,1        Sets Analog input 2 and 3 to be differential inputs on the RIO-
47100
:AQ 2,3        Specify analog input 2 as 0-5V on the RIO-47120
:AQ 0,-3       Specify analog input 0 as 0-5V and the differential input to
analog input 1 on the RIO-47120
:MG_AQ2
 3.0000
```

AT

At Time

Full Description

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

Arguments

AT n where
n is a signed, even integer in the range 0 to 2 Billion
n = 0 defines a reference time at current time
n > 0 specifies a wait time of n msec from the reference time
n < 0 specifies a wait time of n msec from the reference time and re-sets the reference time when the trippoint is satisfied.
(AT -n is equivalent to AT n; AT <old reference +n>)

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	No
Controller Usage	All
Default Value	0
Default Format	-

Operand Usage

Related Commands

TIME - Time Operand
WT - Wait

Examples:

```
#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 ouput 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
```

AZ

Analog Output Select

Full Description

Selects the Analog Outputs used for the control loops. Note - The RIO-472xx does not have any analog outputs or control loops.

Arguments

AZ n,n or AZm=n
n = 0 through 7 selects the analog output channel for control. Set n=-1 to disable control loop
n=? returns analog output channel AF n,n or AFm=n where
n = 0-7 selects the analog input channel for feedback. Set n=-1 to disable control loop
n = ? Returns the input channel m=A,B
RIO-47xx2 has 6 control loops AFn,n,n,n,n,n and m=A,B,C,D,E,F

Operand Usage

_AZm contains the analog input channel selected for the specified control loop

Usage

Usage and Default Details

Usage	Value
While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	-1,-1
Default Format	1.0000

Related Commands

@AN[n] - Read Analog Input
AF - Analog Feedback
PS - Control Loop Setpoint

Examples:

```
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
```

BK

Breakpoint

Full Description

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

Arguments

BK n,m where
n is an integer in the range 0 to 199 which is the line number to stop at. n must be a valid line number in the chosen thread.
m is an integer in the range 0 to 3 that corresponds to the thread number you want to pause.

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

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	No
Command Line	Yes
Controller Usage	All
Default Value	of m 0
Default Format	N/A

Related Commands

SL - Single Step
TR - Trace

Examples:

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

BN

Burn

Full Description

The BN command saves certain board parameters in non-volatile EEPROM memory. This command typically takes 1 second to execute and must not be interrupted. The RIO board returns a colon (:) when the Burn is complete.

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

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	

Arguments

N/A

Operand Usage

_BN contains the serial number of the processor board.

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

BP - Burn Program
BV - Burn Variables and Array

Examples:

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

BP

Burn Program

Full Description

The BP command saves the application program in non-volatile EEPROM memory. This command typically takes up to 1 seconds to execute and must not be interrupted. The unit returns a : when the Burn is complete.

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

Arguments

None

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving	No
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

In a Program No
Command Line Yes
Can be Interrogated No
Used as an Operand No

Related Commands

"BN" Burn Parameters
"BV" Burn Variables

Examples:

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

BV

Burn Variables and Array

Full 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.

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

Arguments

None

Operand Usage

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

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

Examples:

CB

Clear Bit

Full Description

The CB command clears a particular output bit, setting the output to logic 0. The CB and SB (Set Bit) instructions can be used to control the state of output lines.
The CB command can also be used with modbus devices to toggle off-board outputs.

Arguments

CB n where
n is an integer corresponding to a specific output to be cleared (0-15 for on-board outputs).
When using Modbus devices, the I/O points of the Modbus devices are calculated using the following formula:
n = (HandleNum*1000) + ((Module-1)*4) + (Bitnum-1)
HandleNum is the handle specifier, labeled with letters from A to C (1 to 3).
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.

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand No

Related Commands

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

"SB" - Set Bit

Examples:

```
CB 0    Clear output bit 0
CB 1    Clear output bit 1
CB 2    Clear output bit 2
```

CF

Configure Unsolicited Messages Handle

Full Description

Sets the port for unsolicited messages. By default, the controller will send unsolicited data to the main RS-232 serial port. 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. Examples of commands that will generate unsolicited messages follow. These commands are unsolicited only when in embedded code, NOT when sent from a host.

```
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
```

Arguments

CFn
where n ia A,B,C for Ethernet handles*, and S for the serial port.

*On RIO-47xx2 n can be A,B,C,D,E

Operands

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

Usage

Usage and Default Details

Usage	Value
While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All Standalone Controllers
Default Value	S
Default Format	N/A

Related Commands

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

Examples

```
:CFI;' "Send to me"
'Sent from external hardware only, CFI directs
'unsolicited traffic to the port that sent the command
```

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.

GalilTools will by default establish a two handle connection when using Ethernet.

```
Demonstrates from GalilTools terminal that the
main handle is seperate from the unsolicited handle
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"
:
```

CI

Configure Communication Interrupt

Full Description

The CI command configures a program interrupt based on characters received on communications port 2, the AUX serial port (port 1 on DMC-21x2/3 & RIO). An interrupt causes program flow to jump to the #COMINT subroutine. If multiple program threads are used, the #COMINT subroutine runs in thread 0 and the remaining threads continue to run without interruption. The characters received can be accessed via the internal variables P2CH, P2ST, P2NM, P2CD (P1 on DMC-21x2/3 & RIO). For more, see Operator Data Entry Mode in the user manual.

Arguments

CI n, m (m on DMC-21x2/3 and RIO only)

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

RIO And DMC-21x2/3
m = 0 Default, received serial port data is interpreted as Galil command, returning data to the port as a standard interpreted port.
m = 1 Enable serial port for CI execution. Data received will not be interpreted as a command.

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	n = 0, m = 0
Default Format	N/A

Related Commands

CC - Configure communications
IN - Communication input
MG - Message output

Examples:

```
CI 2, 1 Interrupt on a single character received on DMC-21x2/3 serial port
```

CL

Control Loop

Full Description

The CL command allows the user to select the update rate in milliseconds for the optional control loops. The default update rate is 1msec.

Arguments

CL n where
n is an integer in the range of 1 to 65535 that represents the update rate in milliseconds.

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	1
Default Format	N/A

Related Commands

Examples:

```
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
```

CW

Copyright information Data Adjustment bit on off

Full Description

The CW command has a dual usage. The CW command will return the copyright information when the argument, n is 0. Otherwise, the CW command is used as a communications enhancement for use by the Galil terminal software programs. When turned on, the communication enhancement from the command sets the MSB of unsolicited, returned ASCII characters to 1. Unsolicited ASCII characters are characters that are returned from a program running on the controller. This command does not affect solicited characters - which are characters that are returned as a response to a command sent from a host PC.

Arguments

CW n,m where
n is a number, either 0,1 or 2:
0 or ? Causes the controller to return the copyright information
1 Causes the controller to set the MSB of unsolicited returned characters to 1
2 Causes the controller to not set the MSB of unsolicited characters.
m is 0 or 1 (optional)
0 Causes the controller to pause program execution when hardware handshaking disables character transmissions.
1 Causes the controller to continue program execution when hardware handshake disables character transmissions - output characters will be lost.

Operand Usage

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

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	No
Controller Usage	All
Default Value	N/A
Default Format	N/A

In a Program Yes
Command Line Yes
Can be Interrogated Yes
Used as an Operand Yes

Related Commands

Examples:

```
*Note:  The CW command can cause garbled characters to be returned by the
controller.
The default state of the board is to disable the CW command.
However, the terminal software may enable the CW command for internal usage.
If the board is reset while the Galil software is running,
the CW command could be reset to the default value, which creates difficulty
for the software.
It may be necessary to re-enable the CW command.  The CW command status can be
stored in EEPROM.
```

DA

Deallocate the Variables & Arrays

Full Description

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

Arguments

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

Operand Usage

_DA contains the total number of arrays available.

Usage

Usage and Default Details

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

Related Commands

"DM" - Dimension Array

Examples:

```
'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
NOTE:  Since this command deallocates the spaces and compacts the array spaces
in the memory, it is possible that execution of this command may take longer
time than 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.
```

DB

Deadband

Full Description

The DB command selects the deadband where the error must exceed the deadband to execute the control loop.
Note: AF and AZ commands must be set prior to the DB command.

Arguments

DB n,n or DBm=n where
n is a decimal value in the range of -1.000V to 1.0000V. n=? returns the value for the specified control loop. m=A,B

RIO-47xx2 has 6 control loops. DBn,n,n,n,n,n or DBm=n where m=A,B,C,D,E,F

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. With a positive value for n, the output of the PID filter is cleared (set to 0) when the deadband is reached.

Operand Usage

_DBm contains the dead band for the specified control loop

Usage

Usage and Default Details

Usage	Value
While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	0
Default Format	N/A

Default Value 0
In a Program Yes Default Format -----
Command Line Yes

Related Commands

AF - Analog Feedback Select
AZ - Analot Output Select
CL - Control Loop
PS - Setpoint

Examples:

```
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
```

DH

DHCP Server Enable

Full Description

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

Arguments

DH n where
n = 0 disables DHCP and enables BOOT-P
n = 1 disables BOOT-P and enables DHCP
n = ? returns the current state of the setting

Usage

Usage and Default Details

Usage	Value
While Moving	Yes
In a Program	Yes
Command Line	1.0
Controller Usage	DMC-40x0 / RIO-47xxx
Default Value	1.0
Default Format	N/A

Operand Usage

N/A

Related Commands

IA - IP Address

Examples:

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

DL

Download

Full Description

The DL command transfers a data file from the host computer to the controller. If the DL command is used in a terminal utility such as Hyper Terminal, a file will be accepted as a datastream without line numbers. In this case, the file is terminated using {control} Z, {control} Q, {control} D, or \.
If no parameter is specified, downloading a data file will clear any programs in the controller board's RAM. The data is entered beginning at line 0. If there are too many lines or too many characters per line, the controller will return a "?". To download a program after a label, specify the label name following DL. The # argument may be used with DL to append a file at the end of the controller program in RAM.

Arguments

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

Operand Usage

When used as an operand, _DL gives the number of available labels. The total number of labels is 62 on the RIO-47xx0. On the RIO-47xx2 this is increased to 126 labels.

Usage

Usage and Default Details

Usage	Value
While Moving	Yes
In a Program	No
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

In a Program No
Command Line Yes
Can be Interrogated No
Used as an Operand Yes

Related Commands

Examples:

```
DL       Begin Download
SB0      Data
CB2      Data
EN       Data
{control}Z      End download
```

DM

Dimension

Full Description

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

Arguments

DM c[n] where
c is a array name of up to eight alphanumeric characters, starting with an alphabetic character.
n is the number of array elements.

DM? returns the number of array elements available.

Operand Usage

_DM contains the available array space.

Usage

Usage and Default Details

Usage	Value
While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

"DA" - Deallocate Array

Examples:

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

DQ

Change Analog Output Range

Full Description

The DQ command allows the ability to change the analog output range for individual channels when using the RIO-4712x model only. Check the ID command to see if your model supports DQ.

Arguments

DQn,m where
n = 0 through 7 to represent the analog output channel
m = 1 through 4 to designate the analog output range
1 = 0 to +5 vdc
2 = 0 to +10 vdc
3 = -5 to +5 vdc
4 = -10 to +10 vdc (default)

Operand Usage

_DQn contains the present range setting.

Usage

Usage and Default Details

Usage	Value
While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

AO - Analog Output
ID - Identify hardware

Examples:

DR

Configures I O Data Record Update Rate

Full Description

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

Arguments

DR n, m
n specifies the data update rate in msec between updates. n=0 to turn it off, or n must be an integer of at least 2.
m specifies the Ethernet handle on which to periodically send the Data Record. 0 is handle A, 1 is B, 2 is C. The handle must be UDP (not TCP).

Mapping for DataRecord:
header UL
sample number UW
error code UB
general status UB
analog out 0 (counts) UW
analog out 1 (counts) UW
analog out 2 (counts) UW
analog out 3 (counts) UW
analog out 4 (counts) UW
analog out 5 (counts) UW
analog out 6 (counts) UW
analog out 7 (counts) UW
analog in 0 (counts) UW
analog in 1 (counts) UW
analog in 2 (counts) UW
analog in 3 (counts) UW
analog in 4 (counts) UW
analog in 5 (counts) UW
analog in 6 (counts) UW
analog in 7 (counts) UW
output state UW
input state UW
pulse count UL
ZC data SL
ZD data SL

Operand Usage

_DR contains the data record update rate.

Usage

Usage Default Details

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

Default Value DR0 (off)
In a Program Yes Default Format --
Command Line Yes

Related Commands

QR - Query a single data record

Examples:

```
:DR8,0
:G□x□□□□□~□□□P□
_□`□□□□□@~□□□P□
_□H□□□□□`~□□□P□
_□0□□□□□~□□□P□
DR0
'Note:  The data record is in a binary, non-printable format (the output above
is normal when printing to the terminal)
```

DY

PWM output duty cycle

Full Description

DY sets the PWM duty cycle of the RIO-471xx configurable PWM outputs 14 and 15.

The -PWM option is necessary for precise resolution of the duty cycle. See the RIO user manual for more information.

With the -PWM option the accuracy of the PWM output is +/- 0.5% of the DY setting.

Arguments

DY n,n

Where
First argument corresponds to output 14, second to output 15.

n = is an integer in the range of 0 to 100 and sets the PWM Duty cycle percentage

Duty cycle is percentage ON with positive polarity, and percentage OFF with negative polarity (See PM for polarity)

Operands

N/A

Usage

Usage and Default Details

Usage	Value
In a Program	Yes
Command Line	Yes
Default Value	0,0

Related Commands

FQ - PWM output frequency
PM - PWM output enable

Examples

```
:FQ100,200   Set output 14 to 100 Hz, and output 15 to 100 Hz
:DY50,25     Set output 14 to 50%, output 15 to 25%
:PM1,1       Turn PWM mode on for outputs 14 and 15
:
```

ED

Edit

Full Description

Using Telnet style interface (not Galil Software). The ED command puts the RIO board into the Edit subsystem. In the Edit subsystem, programs can be created, changed or destroyed. The commands in the Edit subsystem are:
{cntrl}D Deletes a line
{cntrl}I Inserts a line before the current one
{cntrl}P Displays the previous line
{cntrl}Q Exits the Edit subsystem
{return} Saves a line

Arguments

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

Operand Usage

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

Usage

Usage and Default Details

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

In a Program No
Command Line Yes
Can be Interrogated No
Used as an Operand Yes

Related Commands

DL - Download
UL - Upload

Examples:

```
000 #BEGIN
001 CB1
002 SB3
003 SLKJ        Bad line
004 EN
005 #CMDERR     Routine which occurs upon a command error
006 V=_ED
007 MG "An error has occurred" {n}
008 MG "In line", V{F3.0}
009 AB
010 ZS0
011 EN
HINT:  Remember to quit the Edit Mode prior to executing or listing a program.
```

ELSE

Else function for use with IF conditional statement

Full 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 RIO board will only execute the commands between the IF and ELSE command and then will skip to ENDIF.

Arguments

ELSE

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	No
Controller Usage	All
Default Value	N/A
Default Format	N/A

In a Program Yes
Command Line No
Can be Interrogated No
Used as an Operand No

Related Commands

"ENDIF" - End of IF conditional Statement

Examples:

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

EN

End

Full 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.

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

The EN command is also used to end the automatic subroutines #CMDERR and #COMINT, and an argument is required to handle trippoints in the main thread upon returning from the subroutine.

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

Arguments

EN m, n where
m=0: Return from subroutine without restoring trippoint in main thread
m=1: Return from subroutine and restore trippoint in main thread
n=0: Return from #COMINT without restoring CI interrupt trigger
n=1: Return from #COMINT and restore CI interrupt trigger

Note 1: The default value for the argument is 0. For example EN0 and EN have the same effect.

Note 2: Trippoints cause a program to wait for a particular event. The AI command, for example, waits for specific inputs to go high or low. If the #CMDERR subroutine is executed due to an invalid command while the program is waiting for a trippoint, the #CMDERR can end by continuing to wait for the trippoint as if nothing happened, or clear the trippoint and continue executing the program at the command just after the trippoint. The EN arguments will specify how the #CMDERR and #COMINT routines handle trippoints.

Note 3: Use the RI command to return from the #ININT subroutine.

Note 4: Trippoints occurring in other threads other than the main thread will not be affected by the #CMDERR subroutine.

Usage

Usage and Default Details

Usage	Value
While Moving (no RIO)	Yes
In a Program	Yes
Command Line	No
Controller Usage	All
Default Value	m=0, n=0
Default Format	N/A

Operand Usage

N/A

Related Commands

RI - Return from interrupt subroutine

Examples:

```
#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
```

ENDIF

End of IF conditional statement

Full 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

N/A

Usage

Usage and Default Details

Usage	Value
While Moving	Yes
In a Program	Yes
Command Line	No
Controller Usage	All
Default Value	N/A
Default Format	N/A

Operand Usage

Related Commands

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

Examples:

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

EO

Echo

Full 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.

Serial only, no Ethernet.

Arguments

EO n where
n = 0 0 turns echo off
n = 1 1 turns echo on.

Usage

Usage and Default Details

Usage	Value
While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value (PCI)	0
Default Value (Stand Alone controllers)	1 (Galil software will set EO 0 upon connection)
Default Format	1.0

Operand Usage

_EO contains the state of the echo; 0 is off, 1 is on

Related Commands

Examples:

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

FQ

PWM output frequency

Full Description

FQ sets the PWM frequency of the RIO-471xx configurable PWM outputs 14 and 15.

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.

The FQ command should be set to a known value before the PWM output is enabled.

The resolution of the PWM output is defined by:

RIO-471x2 -
FQ setting of 312 to 20000 the resolution is defined by the following equations:
scale = @RND[80000/FQ]
frequency = 80000/scale
frequency is accurate to +/- 0.1 Hz

FQ setting of 10 to 311 the resolution is:
frequency = FQ
frequency is accurate to +/- 1 Hz

For example:
FQ 19000
scale = @RND[80000/19000] = 4
frequency = 80000/4 = 20000 Hz

FQ 20000
scale = @RND[80000/20000] = 4
frequency = 80000/4 = 20000 Hz

RIO-471x0 -
FQ setting of 235 to 20000 the resolution is defined by the following equations:
scale = @RND[60000/FQ]
frequency = 60000/scale
frequency is accurate to +/- 0.1 Hz

FQ setting of 10 to 234
frequency = FQ
frequency is accurate to +/- 1 Hz

For example:
FQ 19000
scale = @RND[60000/19000] = 3
frequency = 60000/3 = 20000 Hz

FQ 20000
scale = @RND[60000/20000] = 3
frequency = 60000/3 = 20000 Hz

Arguments

FQ n,n

Where
First argument corresponds to output 14, second to output 15.

n = is an integer in the range of 10 to 20,000 and sets the Frequency or the PWM signal in Hertz see Full Description for resolution information.

Operands

N/A

Usage

Usage and Default Details

Usage	Value
In a Program	Yes
Command Line	Yes

Related Commands

DY - PWM output duty cycle
PM - PWM output enable

Examples

```
:FQ100,200   Set output 14 to 100 Hz, and output 15 to 100 Hz
:DY50,25     Set output 14 to 50%, output 15 to 25%
:PM1,1       Turn PWM mode on for outputs 14 and 15
:
```

HS

Handle Assignment Switch

Full Description

The HS command is used to switch the 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.

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

Arguments

HSh=i where
h is the first handle of the switch (A through H, S).
i is the second handle of the switch (A through H, S)
S is used to represent the current handle executing the command.

Usage

Usage and Default Details

Usage	Value
While Moving	Yes
In a Program	Yes
Command Line	Yes
Default Value	N/A

Operand Usage

N/A

Related Commands

IH- IP Handle

Examples:

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

HX

Halt Execution

Full Description

The HX command halts the execution of any of the 4 programs that may be running independently in multitasking. The parameter n specifies the thread to be halted.

Arguments

HX n where
n is 0 to 3 to indicate the 4 threads

Operand Usage

When used as an operand, _HX n contains the running status of thread n with:
0 Thread not running
1 Thread is running
2 Thread has paused at trippoint

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	n=0
Default Format	N/A

In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand Yes

Related Commands

XQ - Execute program

Examples:

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

IA

IP Address

Full Description

The IA command assigns the controller with an IP address.
The IA command may also be used to specify the time out value. This is only applicable when using the TCP/IP protocol.
The IA command can only be used via RS-232. Since it assigns an IP address to the controller, communication with the controller via internet cannot be accomplished until after the address has been assigned.

The RIO is DHCP enabled. If an RS-232 connection is used to assign the IP address, DH must be set to 0 (DH 0) prior to issuing the IA command.

Arguments

IA ip0,ip1,ip2, ip3 or IA n or IA<t where
ip0, ip1, ip2, ip3 are 1 byte numbers separated by commas and represent the individual fields of the IP address.
n is the IP address for the controller, which is specified as an integer representing the signed 32 bit number (two's complement).
> u specifies the multicast IP address where u is an integer between 0 and 63.
< t specifies the time in update samples between TCP retries.
IA? will return the IP address of the RIO board.
DHCP must be disabled (DH0) in order to set the IP address manually with the IA command.

Operand Usage

_IA0 contains the IP address representing a 32 bit signed number (Two's complement)
_IA1 contains the value for t (retry time)
_IA2 contains the number of available handles
_IA3 contains the number of the handle using this operand where the number is 0-2 for RIO-47xx0 and 0-4 for RIO-47xx2. 0 represents handle A, 1 for handle B, 2 for C.
_IA4 contains the handle that lost communication last. Contains -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

Usage

Usage and Default Details

Usage	Value
While Moving (no RIO)	No
In a Program	Yes
Command Line	Yes
Controller Usage	Ethernet Controllers
Default Value	n=0, t=250
Default Format	-

Related Commands

IH - Internet Handle
DH - DHCP Server Enable

Examples:

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

| :

ID

Identify

Full Description

The ID command is used to query the RIO to identify the I/O hardware available. The following is an example response to the ID command and a description of each line.

```
:ID
outputs 0-3 = power sourcing outputs
outputs 4-7 = power sourcing outputs
outputs 8-11 = sinking outputs
outputs 12-15 = sinking 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)
real time clock
:
```

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 ins (eg. RIO-47x0x)
- 'programmable range(AQ)' indicates the unit has configurable analog ins (eg. RIO-47x2x)
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 outs (eg. RIO-47x0x)
- 'programmable range(AQ)' indicates the unit has configurable analog outs (eg. RIO-47x2x)
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 persistant, real time clock circuitry

Arguments

N/A

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	No

Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

Examples:

```
:ID
outputs 0-3 = power sourcing outputs
outputs 4-7 = power sourcing outputs
outputs 8-11 = sinking outputs
outputs 12-15 = sinking outputs
analog inputs 0-3 = 16 bits programmable range(AQ)
analog inputs 4-7 = 16 bits programmable range(AQ)
analog outputs 0-3 = 16 bits programmable range(DQ)
analog outputs 4-7 = 16 bits programmable range(DQ)
```

IF

IF conditional statement

Full Description

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

```
Each condition must be placed in parenthesis for proper evaluation by the
controller.
Example:
IF((var0=1)&(var1=2));' valid IF statement
 MG "GOOD"
ENDIF
IF var0=1&var1=2;'      invalid IF statement
 MG "BAD"
ENDIF
```

Arguments

IF (condition) where
Conditions are tested with the following logical operators:
< less than or equal to
> greater than
= equal to
<= less than or equal to
>= greater than or equal to
<> not equal

Note 1: Bit wise operators | and & can be used to evaluate multiple conditions.

Note 2: A true condition = 1 and an false condition = 0.

Usage

Usage and Default Details

Usage	Value
While Moving	Yes
In a Program	Yes
Command Line	No
Controller Usage	All
Default Value	N/A
Default Format	N/A

Operand Usage

Related Commands

ELSE - Optional command to be used only after IF command
ENDIF - End of IF conditional Statement
JS - Jump to subroutine
JP - Jump to label

Examples:

```
#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
```

```
#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
```

```
REM The conditions of an if statement can be simplied with the fact that
REM a true condition = 1 and a false condition = 0.
#true
v1=1
IF v1
 MG "True v1=",v1
ENDIF
#false
v1=0
IF v1
 'if statement evaluates false
ELSE
 MG "False v1=",0
ENDIF
EN
```

IH

Open IP Handle

Full Description

The IH command is used when the controller is operated as a master (also known as a client). This command opens a handle and connects to a slave (server).
To open a handle, the user must specify:
1. The IP address of the slave
2. The type of session: TCP/IP or UDP/IP
3. The port number of the slave. This number is not necessary if the slave device does not require a specific port value. If not specified, the board will specify the port value as 1000.

Each RIO-47xx0 board may have 3 handles open at any given time and the RIO-47xx2 can have 5 handles. They are designated by the letters A, B,C on the RIO-47xx0 and A,B,C,D, and E on the RIO-47xx2.

On the RIO-471x2, the IH command is used to set the realtime clock from a TIME protocol server. Port 37 is used by the TIME protocol server to transmit a 32 bit unsigned integer in network byte order representing the number of seconds since midnight, January 1, 1900 GMT. Only TCP is supported on the RIO.

The Command RO is used to set the timezone offset from GMT.

Arguments

IHh= ip0,ip1,ip2,ip3 <p >q
IHh=n <p >q
IHh= >r
where
h is the handle
ip0,ip1,ip2,ip3 are integers between 0 and 255 and represent the individual fields of the IP address. These values must be separated by commas.
n is a signed integer between - 2147483648 and 2147483647. This value is the 32 bit IP address and can be used instead of specifying the 4 address fields.

<p specifies the port number of the slave where p is an integer between 0 and 65535. This value is not required for opening a handle.
>q specifies the connection type where q is 0 for no connection, 1 for UDP and 2 for TCP

IHS => r closes the handle that sent the command; where r = -1 for UDP/IP, or r = -2 for TCP/IP, or -3 for either
IHT => r closes all handles except for the one sending the command; where r = -1 UDP, or r = -2 TCP, or -3 for either

>r specifies that the connection be terminated and the handle be freed, where r is -1 for UDP, -2 for TCP/IP, or -3 for TCP/IP Reset

IHh=? returns the IP address as 4 1-byte numbers

Operand Usage

_IHh0 contains the IP address as a 32 bit number
_IHh1 contains the slave port number
_IHh2 contains a 0 if the handle is free
contains a 1 if it is for a UDP slave
contains a 2 if it is for a TCP slave
contains a -1 if it is for a UDP master
contains a -2 if it is for a TCP master
contains a -5 while attempting to establish a UDP handle
contains a -6 while attempting to establish a TCP/IP handle

_IHh3 contains a 0 if the ARP was successful
contains a 1 if it has failed or is still in progress

_IHh4 contains a 1 if the master controller is waiting for acknowledgment from the slave after issuing a command.
contains a 2 if the master controller received a colon from the slave after issuing a command.
contains a 3 if the master controller received a question mark from the slave after issuing a command.
contains a 4 if the master controller timed-out while waiting for a response from the slave after issuing a command.

Usage

Usage and Default Details

Usage	Value
While Moving	Yes
In a Program	Yes
Command Line	Yes
Default Value	0,0,0,0
Default Format	N/A

Related Commands

IA Internet Address

Examples:

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

```
'Setting and printing the time with a TIME protocol server (RIO-471x2 only)
#set
RO-7;'                  set the timezone offset for California (Pacific
Daylight Time)
IHE=>-3;'               close handle E in case it's open
IHE=10,0,62,23<37>2;'   querry 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
```

II

Input Interrupt

Full Description

This newly formatted II command enables the 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.

When the condition is satisfied, the program will jump to the subroutine with label #ININTn, where n ranges from 0 to 3. This subroutine will be executed in thread m, where m is between 0 (main) and 3, 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. The RI command is used to return from the #ININTn routines.

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

Arguments

II n,m,condition,type

n is an integer between 0 and 3. This number specifies the #ININTn subroutine to be executed when the interrupt occurs.

m is an integer between 0 and 3. This argument indicates the thread number in which the #ININTn subroutine is going to be executed. The specified thread needs to be running when the interrupt occurs, otherwise the #ININTn subroutine will not be executed. Upon interrupt, the existing thread m will be interrupted to allow the execution of the interrupt subroutine. Upon completion of the interrupt, the main program in thread m will once again be enabled from the point at which the interrupt occurred.

condition can consist of any number of inputs, using the "&" operator between each input. A positive input number means the condition is for that input to go high, and a negative input number for that input to go low.

type determines the logical operator to be employed in the condition statment when the "&" sign forms a conjunction of inputs. If type is omitted or is 0, the input statements must all be true to trigger #ININTn. If type is 1, then any true input statment will trigger #ININTn.

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving	Yes
In a Program	Yes
Command Line	
Controller Usage	All
Default Value	N/A
Default Format	3.0 (mask only)

In a Program Yes
Command Line No
Can be Interrogated No
Used as an Operand No

Related Commands

RI - Return from Interrupt
#ININTn - Interrupt Subroutine
AI - Trippoint for input

Examples:

```
#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
#CLEAR;
JP#CLEAR,@IN[1]=0       Check for 'reset' input 1 to clear interrupt
RI                      Return to main program
```

IK

Block Ethernet ports

Full Description

A Galil Ethernet controller simultaneusly operates as a server (listening for Ethernet connections from a client) and a client (able to create connections to a server). The IK command blocks clients from connecting to the controller on incoming ports lower than 1000 except for ports 0, 23, 68, and 502.

For SMTP Mail support and the Web Server Feature, IK on the RIO also unconditionally exposes ports 25 and 80.

For the RIO-471x2, the TIME protocol synchronization on port 37 is unaffected by IK. This mechanism employs the RIO as a client, connecting to the TIME server.

Arguments

IKn where
n = 0 allows controller to receive Ethernet packets on any port
n = 1 blocks controller from receiving Ethernet packets on all ports lower than 1000 except those mentioned in the Full Description above.
n = ? queries controller for value of IK

Operand Usage

_IK can not be used as an operand.

Usage

Usage and Default Details

Usage	Value
While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	Ethernet Only
Default Value	0 (DMC21x3/2)
Default Value	1 (DMC40x0/RIO)
Default Format	N/A

Related Commands

TH- Tell Handles
IH - Open new Ethernet handle

Examples

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

IL

Integrator Limit

Full Description

The IL command limits the effect of the integrator gain in the process control loop to a certain voltage. For example, IL 2 limits the output of the integrator of the A-channel to the +/-2 Volt range. AF/AZ must be set prior to adjusting IL. Note: the output from the KD and KP terms is not affected.
For further details see the Process Control section in the User's Manual.

Arguments

IL n,n or ILm=n
n is a positive number whose range is 0-5V on the RIO-4710x and is determined by the DQ command on the RIO-4712x. Resolution for both is 0.0001.
n = ? Returns the value of the integrator limit
m = A,B
RIO-47xx2 has 6 control loops. ILn,n,n,n,n,n and m=A,B,C,D,E,F

Operand Usage

_ILm contains the integrator limit for the specified control loop

Usage

Usage and Default Details

Usage	Value
While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	9.9982
Default Format	1.4

Default Value 5
In a Program Yes Default Format -
Command Line Yes

Related Commands

KI - Integrator

Examples:

```
AF0,1          Set analog feedback
AZ0,1          Set analog output
IL 3,2           Integrator limits
IL ?             Returns the first limit
3.0000
```

IN

Input Variable

Full Description

The IN command allows a variable to be input from a keyboard. When the IN command is executed in a program, the prompt message is displayed. The operator then enters the variable value followed by a carriage return. The entered value is assigned to the specified variable name.
The IN command holds up execution of following commands in a program until a carriage return or semicolon is detected. If no value is given prior to a semicolon or carriage return, the previous variable value is kept. Input Interrupts will still be active.

Arguments

IN "m" , n {So} where
"m" is the prompt message. May be letters, numbers, or symbols up to maximum line length and must be placed in quotations. Make sure that maximum line length of 40 characters is not exceeded.
n is the name of variable to hold value returned from input.
{So} specifies string data where o is the number of characters from 1 to 6. Not required if the data is numerical.
Note 1: Do not leave a space between the comma and n.
Note2: The IN command cannot be used over Ethernet, only RS-232.
Note 3: IN command can only be used in thread 0.

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving	Yes
In a Program	Yes
Command Line	No
Controller Usage	All
Default Value	N/A
Default Format	Position Format

Related Commands

N/A

Examples:

```
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
SBN1;    'Set the specified bit high
MG "DONE";   'Print Message
EN;          'End Program
```

IQ

Digital Input Configuration

Full Description

The IQ command sets the active level for each of the 16 digital inputs. A 0 means current flowing through the opto is logic 1, a 1 means current *not* flowing through the opto is logic 1. Current flowing through the opto can be seen on the RIO when the green LED turns ON. The default state is 0 which means that current not flowing through the opto (LED off) is logic 1.

Arguments

IQn where
n is an integer value 0-65535 representing the 16 digital inputs

Operand Usage

_IQn contains the current setting

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	0
Default Format	N/A

Related Commands

Examples:

```
:IQ255;'This sets bank0 inputs to show current not flowing is logic 0
:IQ192;'This sets outputs 6 and 7 to show current not flowing is logic 0
```

JP

Jump to Program Location

Full Description

The JP command causes a jump to a program location on a specified condition. The program location may be any program line number or label. The condition is a conditional statement which uses a logical operator such as equal to or less than. A jump is taken if the specified condition is true.
Multiple conditions can be used in a single jump statement. The conditional statements are combined in pairs using the operands "&" and "|". The "&" operand between any two conditions, requires that both statements must be true for the combined statement to be true. The "|" operand between any two conditions, requires that only one statement be true for the combined statement to be true.

```
Each condition must be placed in parenthesis for proper evaluation by the
controller.
Example:
JP#a,((var0=1)&(var1=2));' valid conditional jump
JP#a,var0=1&var1=2;'       invalid conditional jump
```

Arguments

JP location,condition where
location is a program line number or label
condition is a conditional statement using a logical operator
The logical operators are:
< less than
> greater than
= equal to
<= less than or equal to
>= greater than or equal to
<> not equal to

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	No
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

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

Examples:

```
JP #POS1,(V1<5);'   Jump to label #POS1 if variable V1 is less than 5
JP #A,((V7*V8)=0);' Jump to #A if V7 times V8 equals 0
JP #B,(@IN[1]=9);'  Jump to #B if input 1 = 1
JP #C;'             Jump to #C unconditionally

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

JS

Jump to Subroutine

Full Description

The JS command will change the sequential order of execution of commands in a program. If the jump is taken, program execution will continue at the line specified by the destination parameter, which can be either a line number or label. The line number of the JS command is saved and after the next EN command is encountered (End of subroutine), program execution will continue with the instruction following the JS command. There can be a JS command within a subroutine, up to 16 deep.

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

```
Each condition must be placed in parenthesis for proper evaluation by the
controller.
Example:
JS#a,((var0=1)&(var1=2));' valid conditional jump
JS#a,var0=1&var1=2;'       invalid conditional jump
```

Arguments

JS destination,condition where
destination is a line number or label
condition is an optional conditional statement using a logical operator
The logical operators are:
< less than or equal to
> greater than
= equal to
<= less than or equal to
>= greater than or equal to
<> not equal

Operand Usage

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	No
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

& | - Bitwise Logical Operators AND and OR
EN - End

Examples:

```
JS #SQUARE,(V1<5);' Jump to subroutine #SQUARE if V1 is less than 5
JS #LOOP,(V1<>0);'  Jump to #LOOP if V1 is not equal to 0
JS #A;'             Jump to subroutine #A (unconditionally)
```

KD

Derivative Constant

Full Description

KD designates the derivative constant in the control filter. The filter transfer function is
D(z) = KP + KD(z-1)/z + KIz/2 (z-1)
The derivative gain outputs a voltage based on the rate of change of the error. For further details see the Process Control section in the User's Manual.

Arguments

KD n,n or KDm=n
n is an unsigned numbers in the range 0 to 4095 with a resolution of 1/8.
n = ? Returns the value of the derivative constant for the specified channel.
m=A,B
RIO-47xx2 has 6 control loops. KDn,n,n,n,n,n or m=A,B,C,D,E,F

Operand Usage

_KDm contains the value of the derivative constant for the specified channel.

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	64
Default Format	4.2

Related Commands

KI - Integrator
KP - Proportional

Examples:

```
KD 100,200      Specify KD
KD ?,?          Return KD
:100.00, 200.00
```

KI

Integrator

Full Description

The KI command sets the integral gain of the control loop. It fits in the control equation as follows:
D(z) = KP + KD(z-1)/z + KI z/2(z-1)
The integrator term will reduce the error at rest to zero. For further details see the Process Control section in the User's Manual.

Arguments

KI n,n or KIm=n where
n is an unsigned numbers in the range 0 to 255 with a resolution of 0.001.
n = ? Returns the value for the specified channel.
m=A,B
RIO-47xx2 has 6 control loops. KIn,n,n,n,n,n or m=A,B,C,D,E,F

Operand Usage

_KIm contains the value of the integral gain for the specified channel.

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	0
Default Format	(4.0 for 18x2 & 21x3), (4.4 for 18x6, 40x0 & RIO)

Default Value 0
In a Program Yes Default Format 4.4
Command Line Yes

Related Commands

KP - Proportional Constant
KD - Derivative Constant
IL - Integrator Limit

Examples:

```
KI 12,14        Specify A,B channel integral gain
KI 7            Specify A channel only
KI ,8           Specify B channel only
KI ?,?          Return A,B
:7, 14          KI values
```

KP

Proportional Constant

Full Description

KP designates the proportional constant in the controller filter. The filter transfer function is
D(z) = KP + KD(z-1)/z + KI z/2(z-1)
The proportianal gain outputs a voltage proportional to the amount of error. For further details see the Process Control section in the User's Manual.

Arguments

KP n,n or KPm=n where
n is an unsigned numbers in the range 0 to 1023.875 with a resolution of 1/8.
n = ? Returns the value of the proportional constant for the specified channel.
m=A,B
RIO-47xx2 has 6 control loops. KPn,n,n,n,n,n or m=A,B,C,D,E,F

Operand Usage

_KPm contains the value of the proportional constant for the specified channel.

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	6
Default Format	4.2

Related Commands

KD - Derivative Constant
KI - Integrator Constant
IL - Integrator Limit

Examples:

```
KP 12,14        Specify A,B channel proportional gain
KP 7            Specify A channel only
KP ,8           Specify B channel only
KP ?,?          Return A,B channel
:7, 14          KP values
```

LA

List Arrays

Full Description

The LA command returns a list of all arrays in memory. The listing will be in alphabetical order. The size of each array will be included next to each array name in square brackets.

Arguments

None

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

In a Program Yes
Command Line Yes
Can be Interrogated Yes
Used as an Operand No

Related Commands

LL - List Labels
LS - List Program
LV - List Variable

Examples:

```
: LA    Interrogation Command
CA [10] RIO board returns a list of 4 Arrays
LA [5]
NY [25]
VA [17]
```

LL

List Labels

Full Description

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

Arguments

None

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

In a Program Yes
Command Line Yes
Can be Interrogated Yes
Used as an Operand No

Related Commands

LV - List Variables

Examples:

```
: LL    Interrogation Command
# FIVE  RIO board returns a list of 5 labels
# FOUR
# ONE
# THREE
# TWO
```

LS

List

Full Description

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

Arguments

LS n,m where
n,m are valid numbers from 0 to 199, or labels. n is the first line to be listed, m is the last.

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	No
Command Line	Yes
Controller Usage	All
Default Value	0, Last Line (for DMC)
Default Format	N/A

In a Program No
Command Line Yes
Can be Interrogated No
Used as an Operand No

Related Commands

LA - List Arrays
LL - List Labels
LV - List Variables

Examples:

```
:LS #A,6        List program starting at #A through line 6
002 #A
003 MG "Program A"
004 COUNT=1
005 SB2
006 WT 2000
```

LV

List Variables

Full Description

The LV command returns a listing of all of the program variables in memory (max number of variables is 126 on the RIO-47xx0 and 254 on the RIO-47xx2). The listing will be in alphabetical order.

Arguments

None

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	VF (for 18x6 & 40x0)

In a Program Yes
Command Line Yes
Can be Interrogated Yes
Used as an Operand No

Related Commands

LL - List Labels

Examples:

```
: LV    Interrogation Command
APPLE  = 60.0000        RIO board returns a list of 3 variables
BOY = 25.0000
ZEBRA = 37.0000
```

LZ

Inhibit leading zeros

Full Description

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

Arguments

LZ n where n is
1 to remove leading zeros
0 to disabled the leading zero removal
LZ? Returns the state of the LZ function.
Note: Default value is 1.

Operand Usage

_LZ contains the state of the LZ command

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Details	1
Default Value	N/A

In a Program Yes
Command Line Yes
Can be Interrogated Yes
Used as an Operand Yes

Related Commands

Examples:

```
TB      Tell status
:001
LZ 1    Inhibit leading zeros
TB      Tell status
:1
```

MA

Email Server IP Address

Full Description

The MA command sets the SMTP email server ip address. The default address is 0.0.0.0 which yields no email transactions.

Arguments

MA n,n,n,n where
n is a value from 0 to 254 for each field of the IP address

Operand Usage

_MA

Usage

Usage and Default Details

Usage	Value
While Moving	No
In a Program	Yes
Command Line	Yes
Controller Usage	RIO
Default Value	0,0,0,0
Default Format	3.0

Related Commands

MD - Set Mail Destination Address
MS - Set Mail Source Address

Examples:

MB

Modbus

Full Description

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

Modbus support is for TCP/IP.

Note: For those command formats that have "addr", this is the slave address. The slave address may be designated or defaulted to the device handle letter.
Note: All the formats contain an h parameter. This designates the connection handle letter (A thru H).
Note: Port 502 must be used in the Ethernet handle. See the IH command for more info on how to open a handle with a specific port number.

Level 2 Modbus Function Codes

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

Function codes 3 and 4 can be swapped with the MV command.

By default the Unit ID is equal to the handle number the connection is over (Handle A gives a Unit ID of 0x01)

Arguments

MBh= addr, 1, m, n, array[] where
h is the handle letter
addr is the unit ID
1 is the function code 1, Read Coil Status
m is the address of the first coil
n is the quantity of coils
array[] is the name of the array whose first element will store the response

MBh= addr, 2, m, n, array[] where
h is the handle letter
addr is the unit ID
2 is the function code 2, Read Input Status
m is the address of the first coil
n is the quantity of coils
array[] is the name of the array whose first element will store the response

MBh= addr, 3, m, n, array[] where
h is the handle letter
addr is the unit ID
3 is the function code 3, Read Holding Registers
m is the address of the first Register
n is the quantity of registers

array[] is the name of the array that will store the resulting register data; 2-byte per element

MBh= addr, 4, m, n, array[] where
h is the handle letter
addr is the unit ID
4 is the function code 4, Read Input Registers
m is the address of the first Register
n is the quantity of registers
array[] is the name of the array that will store the resulting register data; 2-byte per element

MBh= addr, 5, m, n, array[] where
h is the handle letter
addr is the unit ID
5 is the function code 5, Force Single Coil
m is the address of the coil
n is a value of 0 or 1 to turn the coil off or on

MBh= addr, 6, m, n where
h is the handle letter
addr is the unit ID
6 is the function code 6, Preset Single Register
m is register address
n is a 16-bit value

MBh= addr, 7, m, n, array[] where
h is the handle letter
addr is the unit ID
7 is the function code 7, Read Exception Status
m is the address of the first Register
n is the quantity of registers
array[] is the name of the array where the response will be stored in the first element

MBh= addr, 15, m, n, array[] where
h is the handle letter
addr is the unit ID
15 is the function code 15, Write Multiple Coils
m is the address of the first register
n is the quantity of registers
array[] is the name of the array that will store the desired register data; 2-byte per element

MBh= addr, 16, m, n, array[] where
h is the handle letter
addr is the unit ID
16 is the function code 16, Write Multiple Registers
m is the address of the first Register
n is the quantity of registers
array[] is the name of the array that will store the desired register data; 2-byte per element

MBh = addr, 17, array[] where
h is the handle letter
addr is the unit ID
17 is the function code 17, Report Slave ID
array[] is where the returned data is stored
(not supported on RIO)

Raw Modbus Packet Send:
MBh= -1,len,array[] where
h is the handle letter
len is the number of bytes in the array,
array[] is the array containing the outgoing data with a dimension of at least len

Note: each element of array[] may contain only one byte, and array[] must contain the entire modbus packet, including transaction identifiers, protocol identifiers, length field, modbus function code, and data specific to that function code.

Raw Modbus Packet Send/Receive:
MBh= -1,len,array[],len2,len3,array2[] where
h is the handle letter
len is the number of bytes in the array,
array[] is the array containing the outgoing data with a dimension of at least len

Note: each element of array[] may contain only one byte, and array[] must contain the entire modbus packet, including transaction identifiers, protocol identifiers, length field, modbus function code, and data specific to that function code.

len2 is the number of bytes in the incoming data to discard
len3 is the number of bytes following the header in the incoming data to keep
array2[] is the incoming data array with a dimension of at least len3

Note: each element of array2[] will contain only one byte.

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

IA - IP Address
MW - Modbus Wait

MV - Modbus Reversal
MI - Modbus Integer

Examples:

```
IHH=>-3;'                           Close handle, if open
IHH=192,168,1,66<502;'              Establish connection on port 502 (Modbus
port)
WT100;'                             Open loop wait (see operands in IH)
MW1;'                               Turn on Modbus Wait
i=2000;'                            Set address variable
MBH=,6,i,$3333;'                    Write to register
EN
```

MD

Email Destination Address

Full Description

The MD command sets the email destination address (max 30 characters) - default is null which yields no email transactions.

Arguments

MD string where
string is the email address to send to

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving	No
In a Program	Yes
Command Line	Yes
Controller Usage	RIO
Default Value	N/A
Default Format	N/A

Default Value null
In a Program Yes Default Format -
Command Line Yes

Related Commands

MA - Set Email Server IP Address
MS - Set Mail Source Address

Examples:

ME

Modbus array write enable

Full Description

The RIO provides array access for reading and writing over Modbus TCP/IP. 1000 elements are available in the RIO-471x2 array table and 400 in the RIO-471x0 and RIO-47200. Each element is accessible as a 16 bit unsigned integer (Modbus registers 1xxx) -OR- as a 32 bit floating point number (Modbus registers 2xxx).

Reading the registers by a Modbus master is always supported, but writing to the registers is not enabled until ME1 is set on the RIO. Writing is disabled by default to prevent unintentional writes to user arrays.

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. For example, a partioned 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 inaccesible from RIO 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[].

For simplicity, Galil recommends that a single, contiguous array be dimensioned with the array name "A". The dimension of the array A[] should be 1000 on the 471x2 and 400 on the 471x0 and 47200.

Modbus Register Map to Galil Array A[]

Modbus Registers:	1000-1xxx	2000-2xxx
Register range for RIO-471x2	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-471x2)	A[0]-A[999]	A[0]-A[999]
References in A[] array (RIO-471x0/200)	A[0]-A[399]	A[0]-A[399]
Number written to A[]	Integer only, fraction not changed	Galil 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[]

Arguments

ME n
Where
n = 0 (default) disables the ability for Modbus masters to write to the RIO's array table. Read is always enabled through function code 3.
n = 1 enables ability for Modbus masters to write integer or float values using function code 16.
n = ? returns the current setting

Operands

N/A

Usage

Usage and Default Details

Usage	Value

In a Program	Yes
Command Line	Yes
Default Value	0

Related Commands

MB - Modbus
DM - Dimension
DA - Deallocate the Variables & Arrays

Examples

```
: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
:
```

```
'This example is written for a Galil modbus master connected to an RIO-471x2.
'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
RIO.

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
RIO-471x2

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
RIO-471x2
'note that register 2000 would have stepped on the integer memory written at
1000

DM read[2];'                           Dimension an array for holding read data

MBH=0,3,1000,1,read[];'                Read the integer at register 1000
MG"Integer=",read[0];'                 Print the read integer

MBH=0,3,2001,2,read[];'                Read the float at register 1000
float=(read[0]*$10000) + read[1];'    Construct the float. Shifting necesary
for high bytes
MG"Float=", float{$8.0};'              Print the float in hex
```

Hint: A useful utility for determining the 32 bit floating point value for a given fractional number can be found here: http://www.purl.oclc.org/vickery/IEEE-754/Decimal.html

MG

Message

Full Description

The MG command transmits data from the RIO board. This can be used to alert an operator, send instructions or return a variable value.

Arguments

MG "m", "m" {Ex}, {^n}, V {Fm.n or $m,n} {N} {Sn} {M}
"m" is a text message including letters, numbers, symbols or <ctrl>G. Make sure that maximum line length of 40 characters is not exceeded.
{Ex}for ethernet and 'x' specifies the ethernet handle (A, B or C) for sending the message through. {Ex} can be replaced by {P1} to send the message through the serial port. These settings are only needed when the user wants to specify another return port than the one set by the CF command.
{^n} is an ASCII character specified by the value n
V is a variable name or array element, where the following specifiers can be used for formatting:
{Fm.n} Display variable in decimal format with m digits to left of decimal, and n to the right.
{$m,n} Display variable in hexadecimal format with m digits to left of decimal, and n to the right.
{N} Suppress carriage return line feed.
{Sn} Display variable as a string of length n where n is 1 thru 6
{M} emails the message out through an SMTP email server (see MA, MS, and MD commands)
Note: Multiple text, variables, and ASCII characters may be used, each must be separated by a comma. Formatting fields are optional.
Note: The order of arguments is not important.

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	Variable Format

In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand No

Related Commands

Examples:

```
Case 1:  Message command displays ASCII strings
        MG "Good Morning"       Displays the string
Case 2:  Message command displays variables or arrays
        MG "The Answer is", Total {F4.2}  Displays the string with the content
of variable TOTAL in local format of 4 digits before and 2 digits after the
decimal point.
Case 3:  Message command sends any ASCII characters to the port.
        MG {^13}, {^30}, {^37}, {N}  Sends carriage return, characters 0 and 7
followed by no carriage return line feed command to the port.
```

MI

Modbus Integer

Full Description

The MI command specifies whether the RIO as a Modbus slave will respond to Analog I/O related Modbus requests in volts represented by 32-Bit Floating Point notation or in counts represented as a 16-Bit Integer. The MI command affects the way the RIO responds to function codes 3, 4 and 16. Some modbus software packages may require the data to be in integer format.

When MI is set to 1 the range of the 16-bit integer depends on the AQ and DQ setting. Please see the RIO user manual for details.

Arguments

MI n where
n = 0 enables 32-Bit Floating Point notation (volts)
n = 1 enables data returned as a 16-bit integer (counts)
n = ? returns the state of the MI command (0 or 1)

Operand Usage

_MI can not be used as an operand

Usage

Usage and Default Details

Usage	Value
While Moving	No
In a Program	Yes
Command Line	Yes
Controller Usage	RIO
Default Value	0
Default Format	N/A

Related Commands

MB - Modbus
MW - Modbus Wait
MV - Modbus Reversal

Examples:

```
MI1     Set slave to respond to Analog I/O related Modbus requests in counts
```

MS

Email Source Address

Full Description

The MS command sets the email source address (max 30 characters) - default is null.

Arguments

MS string where
string is the email source address

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving	No
In a Program	
Command Line	Yes
Controller Usage	
Default Value	N/A
Default Format	N/A

Default Value null
In a Program Yes Default Format -
Command Line Yes

Related Commands

MA - Set Email Server IP Address
MD - Set Mail Destination Address

Examples:

```
MA 10,0,0,1;                  example smtp email server ip
MD someone@example.com        sample email address
MG "Testing Email"{M}         Message to send via Email
```

MV

Modbus Reversal

Full Description

Enabling the MV command causes the RIO 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.
Note: Use of the Galil commands @AN[], @AO[] require MV to be set to 0.

Arguments

MV n where
n = 0 The RIO responds to function code 3 with analog input information, and function code 4 with analog output information
n = 1 The RIO responds to function code 3 with analog output information, and function code 4 with analog input information
n = ? Returns the state of the MV command (0 or 1)

Operand Usage

_MV can not be used as an operand

Usage

Usage and Default Details

Usage	Value
While Moving	No
In a Program	Yes
Command Line	Yes
Controller Usage	RIO
Default Value	0
Default Format	N/A

Related Commands

MB - Modbus
MW - Modbus Wait

Examples:

```
MV1     RIO to respond to code 3 with analog output and code 4 with analog
input information
```

MW

Modbus Wait

Full Description

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

Arguments

MWn where
n = 0 Disables the Modbus Wait function
n = 1 Enables the Modbus Wait function (default)
n = ? returns the state of the Modbus Wait (0 or 1)

Operand Usage

MW? contains the state of the Modbus Wait.
_MW0 returns last function code received
_MW1 returns modbus error code (see User Manual for details)

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	1 (0 on 21x3)
Default Format	1.0

Related Commands

MB - Modbus

Examples:

```
MW1     Enables Modbus Wait
SB1001  Set Bit 1 on Modbus Handle A
CB1001  Clear Bit 1 on Modbus Handle A
Hint:  The MW command ensures that the command that was sent to the Modbus
device was successfully received before continuing program execution.  This
prevents the controller from sending multiple commands to the same Modbus
device before it has a chance to execute them
```

NO,'

No Operation

Full Description

The NO command performs no action in a sequence, but can be used as a comment in a program. After the NO, characters can be given to form a program comment up to the maximum line length of the controller, which is 40. This helps to document a program.
An apostrophe ' may also be used instead of the NO to document a program.

Arguments

NO m where
m is any group of letters and numbers

Operand Usage

_NO returns a bit field indicating which threads are running. For example, 0 means no threads are running, 1 means only thread 0 is running, 3 means threads 0 and 1 are running, and 255 means all 8 threads are running).

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand No

Related Commands

Examples:

```
#A;'                    Program A
NO                      No Operation
NO This Program         No Operation
' Does Absolutely       No Operation
' Nothing               No Operation
EN;'      End of Program
```

OB

Output Bit

Full Description

The OB n, logical expression command defines output bit n = 0 through 15 as either 0 or 1 depending on the result from the logical expression. Any non-zero value of the expression results in a one on the output.

Arguments

OB n, expression where
n is 0 through 15, denoting output bit
expression is any valid logical expression, variable or array element.

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand No
RELATED COMMAND:
"CB" Clear Bit
"SB" Set Bit

Related Commands

Examples:

```
OB 1, POS=1     If POS 1 is non-zero, Bit 1 is high.
        If POS 1 is zero, Bit 1 is low
OB 2, @IN[5]&@IN[6]     If Input 5 and Input 6 are both high, then
        Output 2 is set high
OB 3, COUNT[1]  If the element 1 in the array is zero, clear bit 3, otherwise
set bit 3
OB N, COUNT[1]  If element 1 in the array is zero, clear bit N
```

OF

Offset

Full Description

The OF command sets a bias voltage on the control output. AF/AZ must be set prior to changing OF.

Arguments

OF n,n or OFm=n where
n is a signed number whose range is -2.5V to 2.5V on the RIO-4710x and is determined by the DQ command on the RIO-4712x. Resolution for both is 0.0001.
n = ? Returns the offset for the specified channel.
m=A,B
RIO-47xx2 has 6 control loops. OFn,n,n,n,n,n or m=A,B,C,D,E,F

Operand Usage

_OFm contains the offset value for the specied axis

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	0
Default Format	1.4 (1.0 for 18x2)

Related Commands

Examples:

```
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
```

OP

Output Port

Full Description

The OP command sends data to the output ports of the controller. You can use the output port to control external switches and relays. The arguments of the OP command are the decimal representation of the general output bits 0 through 7. For example, OP255,0 sets all outputs in bank 0 (bits 0-7) high and all outputs in bank 1 (bits 8-15) low.

Arguments

OP m,n where
m and n are integers in the range 0 to 255 decimal, or $00 to $FF hexadecimal.
Arguments Blocks Bits Description
m 0 0-7 General Outputs (bank 0)
n 1 8-15 General Outputs (bank 1)
m or n = ? returns the value of the argument

Operand Usage

_OP0 contains the value of the first argument, m
_OP1 contains the value of the second argument, n

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	0
Default Format	3.0

Related Commands

SB (Binary EA) - Set output bit
CB - Clear output bit
OB - Output Byte

Examples:

```
OP 0    Clear Output Port -- all bits
OP $85  Set outputs 1,3,8; clear the others
MG _OP0 Returns the first parameter "m"
MG _OP1 Returns the second parameter "a"
```

PC

Pulse Counter Enable

Full Description

Enables the use of a pulse counter on input DI3. When turned on, input DI3 will not be available as a general purpose input accessible with IF or @IN[].

When the -HC (high speed counter) option is ordered with the RIO, DI2 will be the differential input for the counter input. DI2 will not be available as a general purpose input. See the -HC section in the RIO-47xxx User Manual for more information.

Arguments

PCn where
n=0 (default) input DI3 is a general purpose input
n=1 sets input DI3 to be a rising edge pulse counter (also clears the pulse counter)
n=-1 sets input DI3 to be a falling edge pulse counter (also clears the pulse counter)
n=? returns the status of the pulse counter (0 if disabled, 1 if enabled)

Operand Usage

_PC contains the number of pulses counted. _PC will return an integer in the range of 2147483647 to -2147483648

Usage

Usage and Default Details

Usage	Value
While Moving	N/A
In a Program	Yes
Command Line	Yes
Controller Usage	RIO
Default Value	0
Default Format	N/A

Related Commands

Examples:

```
:PC1
:MG_PC
 214
:MG_PC
 515
:t=_PC
:MG t
 718
```

PM

PWM output enable

Full Description

PM enables the PWM output feature for the RIO-471xx. Outputs 14 and 15 can be configured for PWM output with configurable frequency and duty cycle.

The -PWM option is necessary for full resolution of the frequency. See the RIO user manual for more information.

Arguments

PM n,n

Where
First argument corresponds to output 14, second to output 15.

n = 0 PWM output disabled for corresponding output.
n = 1 PWM output enabled with high polarity
high polarity with DY 100 (100% duty cycle) is the same as the "CB" state with PM0
n = -1 PWM output enabled with low polarity
low polarity with DY 100 (100% duty cycle) is the same as the "SB" state with PM0

Operands

N/A

Usage

Usage and Default Details

Usage	Value
In a Program	Yes
Command Line	Yes
Default Value	0,0

Related Commands

DY - PWM output duty cycle
FQ - PWM output frequency

Examples

```
:FQ100,200   Set output 14 to 100 Hz, and output 15 to 100 Hz
:DY50,25     Set output 14 to 50%, output 15 to 25%
:PM1,1       Turn PWM mode on for outputs 14 and 15
:
```

PS

Control Setpoint

Full Description

The control setpoint command is used in a process control loop to specify the analog value that will be used as the setpoint. The range and resolution for the setpoint is dependant on the RIO hardware configuration. See the AQ command for the Analog Input range.
Note: The AF and AZ commands must be set before the use of PS.

Arguments

PS n,n or PSm=n
n is a signed value from 0V to 5v on the RIO-4710x or within the DQ range on the RIO-4712x. n=? returns the control set point
Resolution for both is 0.0001.
m=A,B
RIO-47xx2 has 6 control loops. PSn,n,n,n,n,n or m=A,B,C,D,E,F

Operand Usage

_PSm returns setpoint for the specified axis

Usage

Usage and Default Details

Usage	Value
While Moving	No
In a Program	Yes
Command Line	Yes
Controller Usage	RIO
Default Value	0
Default Format	1.4

Default Value 0
In a Program Yes Default Format 1.4
Command Line Yes

Related Commands

Examples:

```
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
```

PW

Password

Full Description

The password can be set with the command PW password,password where the password can be up to 8 alphanumeric characters. The default value after master reset is a null string. The password can only be changed when the controller is in the unlocked state (^L^K). The password is burnable but cannot be interrogated. If you forget the password you must master reset the controller to gain access.

Arguments

PW n,n where
n is a string from 0 to 8 characters in length

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes (No for 40x0)
Command Line	Yes
Controller Usage	All
Default Value	"" (null string)
Default Format	N/A

Related Commands

<control>L<control>K - Lock/Unlock
ED - Edit program
UL - Upload program
LS - List program
TR - Trace program

Examples:

```
:PWtest,test            Set password to "test"
:^L^K test,1            Lock the program
:ED                     Attempt to edit program
```

QD

Download Array

Full Description

The QD command transfers array data from the host computer to the RIO.
QD array[],start,end requires that the array name be specified along with the first element of the array and last element of the array. The array elements can be separated by a comma (,) or by <CR><LF>. The downloaded array is terminated by a \.

Arguments

QD array[],start,end where
"array[]" is a valid array name
"start" is the first element of the array (default=0)
"end" is the last element of the array (default=last element)

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	No
Command Line	Yes
Controller Usage	All
Default Value	start=0, end=size-1
Default Format	N/A (Position Format for 18x2)

In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand No

Related Commands

QU - Upload array

Examples:

```
Using Galil terminal software, the command can be used in the following manner:
1.      Define the array with the DM command
2.      Set the timeout to 0 (this disengages the timeout)
3.      Send the command QD
3a.  Use the send file command to send the data file.
OR
3b.  Enter data manually from the terminal.  End the data entry with the
character '\'
4.      Set the timeout back to a positive number
```

QR

I O Data Record

Full Description

The QR command causes the RIO to return a record of information regarding the I/O status back to the host PC. 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 user manual.

Arguments

QR
Mapping for DataRecord:
header UL
sample number UW
error code UB
general status UB
analog out 0 (counts) UW
analog out 1 (counts) UW
analog out 2 (counts) UW
analog out 3 (counts) UW
analog out 4 (counts) UW
analog out 5 (counts) UW
analog out 6 (counts) UW
analog out 7 (counts) UW
analog in 0 (counts) UW
analog in 1 (counts) UW
analog in 2 (counts) UW
analog in 3 (counts) UW
analog in 4 (counts) UW
analog in 5 (counts) UW
analog in 6 (counts) UW
analog in 7 (counts) UW
output state UW
input state UW
pulse count UL
ZC data SL
ZD data SL

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	
Default Value	N/A
Default Format	N/A

Related Commands

QZ - Return DMA / Data Record information
Note: The Galil windows terminal will not display the results of the QR command since the results are in binary format.

QZ - Return Data Record information
Note: The Galil windows terminal will not display the results of the QR command since the results are in binary format.

Examples:

QU

Upload Array

Full Description

The QU command transfers array data from the RIO to a host computer. QU requires that the array name be specified along with the first element of the array and last element of the array. The uploaded array will be followed by a <control>Z as an end of text marker.

Arguments

QU array[],start,end,delim where
"array[]" is a valid array name
"start" is the first element of the array (default=0)
"end" is the last element of the array (default = last element)
"delim" specifies the character used to delimit the array elements. If delim is 1, then the array elements will be separated by a comma. Otherwise, the elements will be separated by a carriage return.

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	0
Default Format	Position Format

In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand No

Related Commands

QD - Download array

Examples:

QZ

Return Data Record information

Full Description

The QZ command is an interrogation command that returns information regarding the Data Record. The RIO board's response to this command will be the return of 4 integers separated by commas.
The first field returns the number of control loops
The second field returns the number of bytes in the general data block of the QR record. This is always 4 for the RIO.
Third field returns the number of bytes in the I/O block of the QR record
Fourth field returns 0, representing nothing

Arguments

QZ

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand No

Related Commands

QR - Data Record

Examples:

```
QZ       Enter Command
 2, 4, 48, 0
```

RA

Record Array

Full Description

The RA command selects one or two arrays for automatic data capture. The selected arrays must have been dimensioned by the DM command. The data to be captured is specified by the RD command and time interval by the RC command.

Arguments

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

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

DM - Dimension Array
RD - Record Data
RC - Record Interval

Examples:

```
#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.
```

RC

Record

Full Description

The RC command begins recording for the Automatic Record Array Mode (RA). RC 0 stops recording.

Firmware Note: Do not allocate or deallocate arrays (DM,DA) while the Automatic Record Array Mode is running.

GalilTools Note: Do not download arrays from GalilTools, or call the arrayDownload() or arrayDownloadFile() apis while Automatic Record Array Mode is running.

Arguments

RC n,m where
n is an integer 1 thru 8 and specifies 2^n samples between records. RC 0 stops recording.
m is optional and specifies the number of records to be recorded. If m is not specified, the array bounds will be used. A negative number for m causes circular recording over array addresses 0 to m-1.

n = ? Returns status of recording. '1' if recording, '0' if not recording.

Note: The address for the array element for the next recording can be interrogated with _RD.

Operand Usage

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

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	0

Related Commands

DM - Dimension Array
RD - Record Data
RA - Record Array Mode

Examples:

```
#RECORD;'              Record label
DM Torque[1000];'      Define Array
RA Torque[];'          Specify Array to record data
RD _TTA;'              Specify Data Type
RC 2;'                 Begin recording and set 4 msec between records
JG 1000;BG;'           Begin motion
#A;JP #A,_RC=1;'       Loop until done
MG "DONE RECORDING";'  Print message
EN;'                   End program
```

RD

Record Data

Full Description

The RD command specifies the data type to be captured for the Record Array (RA) mode. The command type includes:

DATA TYPE MEANING
_TI Input status (0-15)
_OP Output status (0-15)
_AFn Analog inputs status (0-7)
_AOn Analog outputs (0-7)

Arguments

RD m1, m2 where
the arguments are the data type to be captured using the record array feature. The order is important. Each of the two data types corresponds with the array specified in the RA command.

Operand Usage

_RD contains the address for the next array element for recording.

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand Yes

Related Commands

DM - Dimension Array
RA - Record Array
RC - Record Interval

Examples:

```
See example for RA command.
```

RE

Return from Error Routine

Full Description

The RE command is used to end the error handling subroutine #TCPERR. An RE at the end of this routine causes a return to the main program. Care should be taken to be sure the error condition no longer occurs to avoid re-entering the subroutine. If the program sequencer was waiting for a trippoint to occur, prior to the error interrupt, the trippoint condition is preserved on the return to the program if RE1 is used. RE0 clears the trippoint. To avoid returning to the main program on an interrupt, use the ZS command to zero the subroutine stack.

Arguments

RE n where
n = 0 Clears the interrupted trippoint
n = 1 Restores state of trippoint
no argument clears the interrupted trippoint

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	No
In a Program	Yes
Command Line	No
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

#TCPERR - Error Subroutine

Examples:

```
#L
          MG {EA} "L"
          WT1000
       JP#L
        #TCPERR
          MG {P1} "TCPERR.  Dropped handle", _IA4
        RE
```

REM

Remark

Full Description

REM is used for comments. The REM statement is NOT a controller command. Rather, it is recognized by Galil PC software, which strips away the REM lines before downloading the DMC file to the controller. REM differs from NO (or ') in the following ways:
(1) NO (or ') comments are downloaded to the controller and REM comments aren't
(2) NO (or ') comments take up execution time and REM comments don't; therefore, REM should be used for code that needs to run fast.
(3) REM comments cannot be recovered when uploading a program but NO (or ') comments are recovered. Thus the uploaded program is less readable with REM.
(4) NO (or ') comments take up program line space and REM lines don't.
(5) REM comments must be the first and only thing on a line, whereas NO (or ') can be used to place comments to the right of code (after a semicolon) on the same line

NO (or ') should be used instead of REM unless speed or program space is an issue.

Arguments

REM string where
string is a text comment

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	No
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

NO (' apostrophe also accepted) - No operation (comment)

Examples:

```
REM This comment will be stripped when downloaded to the controller
'This comment will be downloaded and takes some execution time
PRX=1000; 'this comment is to the right of the code
```

RI

Return from Interrupt Routine

Full Description

The RI command is used to end the interrupt subroutine beginning with the label #ININTn. If the program sequencer was interrupted while waiting for a trippoint (such as WT), RI1 restores the trippoint on the return to the program. RI0 clears the trippoint. The second field of the RI command either restores or disables the input interrupt feature. This field has been added as a special feature of the RIO. To avoid returning to the main program on an interrupt, use the command ZS to zero the subroutine stack. This turns the jump subroutine into a jump only.

Arguments

RI m,n where
m = 0 or 1
0 clears trippoint (e.g. AI trippoint)
1 restores trippoint
n = 0 (or no field) or 1
0 restores interrupt
1 disables interrupt

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving	No
In a Program	Yes
Command Line	No
Controller Usage	All
Default Value	N/A
Default Format	N/A

In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand No

Related Commands

#ININT - Input interrupt subroutine
II - Enable input interrupts

Examples:

```
#A;II1,0,3      Program label; enable interrupt on input 3
AI5;MG"DONE";EN AI trippoint on input 5; end of program
#ININT1 Begin interrupt subroutine
MG "IN[3] INTERRUPTED"  Print Message
CB 3    Set output line 1
RI 1,1  Return to the main program, restore AI trippoint and disable interrupt
on input 3
HINT:  An applications program must be executing for the #ININTn subroutine to
function.
```

RO

Realtime Offset

Full Description

Available on RIO-471x2 only. RO Sets the hour offset from GMT time for the realtime clock feature. RO is used when synching the realtime clock on the RIO with a TIME server. The TIME protocol provides GMT time, and RO allows for localization of time.

RO is burnable with BN.

Arguments

ROn
where
n is an integer +/-12 corresponding to the hourly time zone offset from GMT.

Operands

_RO0 contains the current GMT offset
_RO1 contains the value last received from a network TIME protocol server (See IH and RT)

Usage

Usage and Default Details

Usage	Value
In a Program	Yes
Command Line	Yes
Default Value	0

Related Commands

RT - Real Time
RY - Real Year (Calender Feature)

Examples

```
'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
```

```
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};'Second
EN
'SAMPLE OUTPUT:
' 11: 23: 16
' 10: 23: 16
```

RS

Reset

Full 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.

RS-1 Soft master reset. Restores factory defaults without erasing EEPROM. To restore saved EEPROM settings use RS with no arguments.

Arguments

Operand Usage

_RS returns the state of the processor on its last power-up condition. The value returned is the decimal equivalent of the 4 bit binary value shown below.
Bit 3 For master reset error (there should be no program to execute)
Bit 2 For program check sum error
Bit 1 For parameter check sum error
Bit 0 For variable check sum error

Usage

Usage and Default Details

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

In a Program No
Command Line Yes
Can be Interrogated Yes
Used as an Operand Yes

Related Commands

Examples:

```
RS       Reset the hardware
```

RT

Real Time

Full Description

Available on RIO-471x2 only. RT provides the ability to set and querry the current time on the RIO realtime clock.

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. Provides calendar function (See RY).

RT can be set manually, or automatically with the TIME protocol (See IH).

Arguments

RTs, m, h

Where
s is current seconds value (0-59)
m is current minutes value (0-59)
h is the current 24 hour clock hour value (0-23)

Operands

_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 (RIO-471x2 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 occured, _RT3 contains the number of days since last power cycle. With the precision RTC upgrade, use _RY0 for the day of week counter.

Usage

Usage and Default Details

Usage	Value
In a Program	Yes
Command Line	Yes
Default Value	0,0,0

Related Commands

RY - Real Year Calendar Function
RO - Realtime Offset
IH - Open IP Handle

Examples

```
:RT 30,25,22     Set time to 10:25:30 PM
:
```

```
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},":"{N};'print hour
```

```
MG_RT1{F2.0},":"{N};'print minutes
MG_RT0{F2.0};'print seconds
EN
'Sample Output:
'$CF943C7B
' 13: 28: 43
```

RY

Real Year Calendar Function

Full Description

Available only on RIO-471x2 with realtime chip upgrade. RY provides a calender 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 calender function is leap-year compliant.

Arguments

RY dw, dm, m, y

Where
dw is the current day of the week (1=Sunday, 2=Monday, 3=Tuesday, 4=Wednesday, 5=Thursday, 6=Friday, 7=Saturday)

dm is the current day of the month (1-31)

m is the current month of the year (1=January, 2=February, 3=March, 4=April, 5=May, 6=June, 7=July, 8=August, 9=September, 10=October, 11=November, 12=December).

y is the current year (0-99 corresponding to 2000 through 2099)

Operands

_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

Usage

Usage and Default Details

Usage	Value
In a Program	Yes
Command Line	Yes
Default Value	0,0,0,0

Related Commands

RY - Real Year Calendar Function
RO - Realtime Offset
IH - Open IP Handle

Examples

```
:RY6,19,2,10     Set to Fri, February 19th, 2010
:
```

```
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
```

SA

Send Command

Full Description

SA sends a command from the RIO to another Galil Ethernet controller (or another RIO unit). Any command can be sent to a slave device and will be interpreted by the controller as a "local" command.
Note: A wait statement (e.g. WT5) must be inserted between successive calls to SA.

Arguments

SAh= arg or SAh=arg,arg,arg,arg,arg,arg,arg,arg where
h is the handle being used to send commands to another Galil Ethernet controller (A, B, or C on the RIO-47xx0 or A,B,C,D,E on the RIO-47xx2)
arg is a number, a controller operand, variable, mathematical function, or string; The range for numeric values is 4 bytes of integer (2^31)followed by two bytes of fraction (+/-2,147,483,647.9999). The maximum number of characters for a string is 6 characters. Strings are identified by quotations.
Typical usage would have the first argument as a string such as "OQ" and the subsequent arguments as the arguments to the command: Example SAA= "KD",20,40 would send the command "KD 20,40".

Operand Usage

_SAhn gives the value of the response to the command sent with an SA command. The h value represents the handle A, B or C and the n value represents the specific field returned from typically a multi-channel controller. The n value gets defaulted to 1 if unspecified in the operand. If a specific field is not used by the controller (e.g. fifth field on a four axes controller), the operand will be -2^31. Response from another RIO board has usually just one field; therefore the n value does not need to be included in the operand.

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand Yes

Related Commands

IH - Set Internet Handles

Examples:

```
SAA="KI",1,2    Sends the command to handle A (e.g. a controller):  KI 1,2
WT5
SAA="TE"        Sends the command to handle A (controller):  TE
MG _SAA         Display the content of the operand _SAA (first response to TE
command)
: 132
MG _SAA2        Display the content of the operand _SAA (2nd response to TE
command)
: 12
Note:  The SA command does not wait for a response from the slave controller
before continuing code execution.  Therefore, a WTxx is required between two SA
```

commands or between an SA command and querying the response using _SAhn. There is a 38 character maximum string length for the SA command. It is helpful for timing to keep the SA command query as short as possible.

SB

Set Bit

Full Description

The SB command sets a particular output bit, setting the output to logic 1. The SB and CB (Clear Bit) instructions can be used to control the state of output lines. The SB command can also be used with modbus devices to toggle off-board outputs.
Note 1: When using Modbus devices, the I/O points of the modbus devices are calculated using the following formula:
n = (HandleNum*1000) + ((Module-1)*4) + (Bitnum-1)
HandleNum is the handle specifier from A,B or C (1 to 3).
Module is the position of the module in the rack from 1 to 16.
BitNum is the I/O point in the module from 1 to 4.

Arguments

SB n where
n is an integer in the range 0 to 15 decimal.

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A
Related Command	CB Clear Bit

In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand No

Related Commands

CB - Clear Bit
OB - Output Bit

Examples:

```
SB 3    Set output line 3
SB 1    Set output line 1
```

SL

Single Step

Full Description

For debugging purposes. Single Step through the program after execution has paused at a breakpoint (BK). Optional argument allows user to specify the number of lines to execute before pausing again. The BK command resumes normal program execution.

Arguments

SL n where
n is an integer representing the number of lines to execute before pausing again

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	No
Command Line	Yes
Controller Usage	All
Default Value	1
Default Format	

Default Value 1
In a Program No
Command Line Yes

Related Commands

BK - Breakpoint
TR - Trace

Examples:

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

SM

Subnet Mask

Full Description

The SM command assigns a subnet mask to the controller. All packets sent to the controller whose source IP address is not on the subnet will be ignored by the controller. For example, for SM 255, 255, 0, 0 and IA 10, 0, 51, 1, only packets from IP addresses of the form 10.0.xxx.xxx will be accepted.

Arguments

SM sm0, sm1, sm2, sm3 or SM n where
sm0, sm1, sm2, sm3 are 1 byte numbers (0 to 255) separated by commas and represent the individual fields of the subnet mask.
n is the subnet mask for the controller, which is specified as an integer representing the signed 32 bit number (two's complement).
SM? will return the subnet mask of the controller

Operand Usage

_SM0 contains the IP address representing a 32 bit signed number (Two's complement)

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	
Default Value	SM 0, 0, 0, 0
Default Format	N/A

Related Commands

IH - Internet Handle
IA - IP address

Examples:

```
SM 255, 255, 255, 255   Ignore all incoming Ethernet packets
SM 0, 0, 0, 0   Process all incoming Ethernet packets
```

TB

Tell Status Byte

Full Description

The TB command returns status information from the RIO as a decimal number. Each bit of the status byte denotes the following condition when the bit is set (high):

BIT	STATUS
Bit 7	Executing program
Bit 6	N/A
Bit 5	N/A
Bit 4	N/A
Bit 3	Input interrupt enabled in thread 0
Bit 2	Executing input interrupt routine in thread 0
Bit 1	0 (Reserved)
Bit 0	Echo on

Arguments

None

Operand Usage

_TB contains the status byte.

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	3.0

In a Program	Yes
Command Line	Yes
Can be Interrogated	Yes

Related Commands

Examples:

```
TB      Tell status information from the RIO board
129     Executing program and echo on  (2^7 + 2^0 = 128 + 1 = 129)
```

TC

Tell Error Code

Full Description

The TC command returns a number between 1 and 255. This number is a code that reflects why a command was not accepted by the controller. This command is useful when the controller halts execution of a program or when the response to a command is a question mark. After TC has been read, the error code is set to zero.

TC1 will return the error code, along with a human-readable description of the code.

Tell Code List

Tell Code Number	Description	Notes
1	Unrecognized command	
2	Command only valid from program	
3	Command not valid in program	
4	Operand error	
5	Input buffer full	
6	Number out of range	
7	Command not valid while running	not valid for RIO
8	Command not valid while not running	not valid for RIO
9	Variable error	
10	Empty program line or undefined label	
11	Invalid label or line number	
12	Subroutine more than 16 deep	
13	JG only valid when running in jog mode	not valid for RIO
14	EEPROM check sum error	
15	EEPROM write error	
16	IP incorrect sign during position move or IP given during forced deceleration	not valid for RIO
17	ED, BN and DL not valid while program running	
18	Command not valid when contouring	not valid for RIO
19	Application strand already executing	
20	Begin not valid with motor off	not valid for RIO
21	Begin not valid while running	not valid for RIO
22	Begin not possible due to Limit Switch	not valid for RIO
24	Begin not valid because no sequence defined (no RIO)	
25	Variable not given in IN command	
28	S operand not valid	not valid for RIO
29	Not valid during coordinated move	not valid for RIO
30	Sequenct Segment Too Short	not valid for RIO
31	Total move distance in a sequence > 2 billion	not valid for RIO
32	Segment buffer full	not valid for RIO
33	VP or CR commands cannot be mixed with LI commands	not valid for RIO
39	No time specified	not valid for RIO
41	Contouring record range error	not valid for RIO
42	Contour data being sent too slowly	not valid for RIO
46	Gear axis both master and follower	not valid for RIO
50	Not enough fields	
51	Question mark not valid	
52	Missing " or string too long	

53	Error in {}	
54	Question mark part of string	
55	Missing [or []	
56	Array index invalid or out of range	
57	Bad function or array	
58	Bad command response (i.e._GNX)	
59	Mismatched parentheses	
60	Download error - line too long or too many lines	
61	Duplicate or bad label	
62	Too many labels	
63	IF statement without ENDIF	
65	IN command must have a comma	
66	Array space full	
67	Too many arrays or variables	
71	IN only valid in thread #0	
80	Record mode already running	
81	No array or source specified	
82	Undefined Array	
83	Not a valid number	
84	Too many elements	
90	Only A B C D valid operand	not valid for RIO
96	SM jumper needs to be installed for stepper motor operation (no Accelera, no RIO)	
97	Bad Binary Command Format	
98	Binary Commands not valid in application program	
99	Bad binary command number	
100	Not valid when running ECAM	not valid for RIO
101	Improper index into ET	not valid for RIO
102	No master axis defined for ECAM	not valid for RIO
103	Master axis modulus greater than 256 EP value	not valid for RIO
104	Not valid when axis performing ECAM	not valid for RIO
105	EB1 command must be given first	not valid for RIO
106	Privilege Violation	not valid for Econo, Optima
110	No hall effect sensors detected	not valid for RIO
111	Must be made brushless by BA command	not valid for RIO
112	BZ command timeout	not valid for RIO
113	No movement in BZ command	not valid for RIO
114	BZ command runaway	not valid for RIO
118	Controller has GL1600 not GL1800	not valid for RIO
119	Not valid for axis configured as stepper	
120	Bad Ethernet transmit	not valid for PCI
121	Bad Ethernet packet received	not valid for PCI
122	Ethernet input buffer overrun	DMC-21x3 only
123	TCP lost sync	not valid for PCI
124	Ethernet handle already in use	not valid for PCI
125	No ARP response from IP address	not valid for PCI
126	Closed Ethernet handle	not valid for PCI
127	Illegal Modbus function code	not valid for PCI
128	IP address not valid	not valid for PCI
131	Serial Port Timeout	not valid for PCI
132	Analog inputs not present	

133	Command not valid when locked / Handle must be UDP	not valid for PCI
134	All motors must be in MO for this command	not valid for RIO
135	Motor must be in MO	not valid for RIO
136	Invalid Password	not valid for Econo, Optima
137	Invalid lock setting	not valid for Econo, Optima
138	Passwords not identical	not valid for Econo, Optima

Arguments

TC n where
n = 0 Returns numerical code only
n = 1 Returns numerical code and human-readable message
n = ? Returns the error code

Operand Usage

_TC contains the value of the error code.

Usage

Usage Details

Usage	Value
While Moving	Yes (No RIO)
In a Program	Yes
Not in a program	Yes
Default Value	N/A
Default Format	3.0

Related Commands

Examples:

```
:GF32   Bad command
?TC1    Tell error code
1       Unrecognized command
```

TE

Tell Error

Full Description

:
This command returns the current error of the analog signal in the control loop. For further details see the Process Control section in the User's Manual.

Arguments

TE or TEn where
n is A or B
No argument will provide the position error for all channels

Operand Usage

_TEn contains the current position error value for the specified channel.

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	0
Default Format	Position Format

Default Value 0
In a Program Yes
Command Line Yes

Related Commands

CL - Control Loop update rate
DB - Deadband
PS - Control Loop Setpoint
AF - Analog Feedback Select
AZ - Analgo Output Select

Examples:

```
TE      Return all position errors
:1,-2
Error =_TEA     Sets the variable, Error, with the channel 1 position error
```

TH

Tell Ethernet Handle

Full Description

This 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

N/A

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (no RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	Ethernet Only
Default Value	-
Default Format	-

Related Commands

HS - Handle Swap
IA - IP address
IH - Internet Handle
WH - Which Handle

Examples:

```
:TH
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
```

TI

Tell Inputs

Full Description

This command returns the state of inputs in banks of 8. Response is a decimal number which when converted to binary represents the status of the digital inputs for the specified bank. TI0 specifies inputs 0-7 (bank 0) and TI1 specifies inputs 8-15 (bank 1) - the response will be a number from 0 to 255.

Arguments

TI n where
n ranges from 0 thru 1.

Operand Usage

_TIn contains the status of the input module for bank n.
Note: The operand can be masked to return only specified bit information - see section on Bitwise operations.

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	3.0

In a Program Yes
Command Line Yes
Can be Interrogated Yes
Used as an Operand Yes

Related Commands

Examples:

```
TI1     Tell input state on bank 1
8       Bit 3 on slot 1 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
```

TIME

Time Operand (Keyword)

Full Description

*The TIME operand contains the value of the intenal 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.
The keyword, TIME, does not require an underscore (_) as with the other operands.

Arguments

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	
In a Program	Yes
Command Line	Yes
Controller Usage	
Default Value	
Default Format	

In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand Yes

Related Commands

Examples:

```
MG TIME Display the value of the internal clock
T1=TIME Sets the variable, T1, with the TIME value
```

TR

Trace

Full Description

The TR command causes each instruction in a program to be sent out the communications port prior to execution. TR1 enables this function and TR0 disables it. The trace command is useful in debugging programs.

Note: The lines following the Line Continuation Character (`) will not be displayed in the trace output.

```
#A
a=123`
456;'not displayed with TR1 output
EN
```

Arguments

TR n where
n = 0 or 1
0 disables function
1 enables function

Usage

In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand No

Operand Usage

Related Commands

` - Line Continuation Character

Examples:

TZ

Tell I O Configuration

Full Description

This command returns a list of information pertaining to the status of the RIO's I/O. 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

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

In a Program Yes
Command Line Yes
Can be Interrogated Yes
Used as an Operand No
RELATED COMMAND:
"TH" Tell Ethernet Handle
"TQ" Tell Thread Execution

Related Commands

TI - Tell Inputs
SB/CB - Set/Clear output bits
OP - Output port
CO - Configure I/O

Examples:

```
:TZ
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
```

UL

Upload

Full Description

The UL command transfers data from the RIO to a host computer. Programs are sent without line numbers. In the Galil software, the UL command is not necessary because the UL command is handled by the graphical interface (Upload Program). In a terminal utility such as HyperTerminal or Telnet, the UL command will bring the uploaded program to screen. From there, the user can copy it and save it to a file. In Hyper Terminal, the UL command will be followed by a <control>Z or a \ as an end of Text marker.

Arguments

None

Operand Usage

When used as an operand, _UL gives the number of available variables. The total number of variables is 126 on the RIO-47xx0 or 254 on the RIO-47xx2

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	No
Command Line	Yes
Controller Usage	All
Default Value	0
Default Format	N/A

In a Program No
Command Line Yes
Can be Interrogated No
Used as an Operand Yes

Related Commands

DL - Download

Examples:

```
UL;      Begin upload
#A;'     Line 0
SB1;'    This is an Example Line 1
CB1;'    Program Line 2
EN;'     Line 3
{cntrl}Z         Terminator
```

VF

Variable Format

Full Description

The VF command formats the number of digits to be displayed when interrogating the RIO board.
If a number exceeds the format, the number will be displayed as the maximum possible positive or negative number (i.e. 999.99, -999, $8000 or $7FF).

Arguments

VF m.n where
m and n are unsigned numbers in the range 0<m<10 and 0<n<4.
m represents the number of digits before the decimal point. A negative m specifies hexadecimal format. When in hexadecimal, the string will be preceded by a $ and Hex numbers are displayed as 2's complement with the first bit used to signify the sign.
n represents the number of digits after the decimal point.
m = ? Returns the value of the format for variables and arrays.

Operand Usage

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

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	10.4
Default Format	2.1

In a Program Yes
Command Line Yes
Can be Interrogated Yes
Used as an Operand Yes

Related Commands

Examples:

```
VF 5.3  Sets 5 digits of integers and 3 digits after the decimal point
VF 8.0  Sets 8 digits of integers and no fractions
VF -4.0 Specify hexadecimal format with 4 bytes to the left of the decimal
```

WH

Which Handle

Full Description

The WH command is used to identify the handle in which the command is executed on. The command returns IHA, IHB, or IHC to indicate on which handle the command was executed on in the RIO-47xx0 or IHA,IHB,IHC,IHD,IHE on the RIO-47xx2. The command returns RS232 if using serial communication.

Arguments

None

Operand Usage

_WH contains the numeric representation of the handle in which a command is executed. Handles A - Eare indicated by the value 0 - 4, while a -1 indicates the serial port.

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes (No on 21x3)
Command Line	Yes
Controller Usage	All
Default Value	N/A
Default Format	N/A

Related Commands

HS - Handle Swap
IA - IP address
IH - Internet Handle
TH -Tell Handles

Examples:

```
:WH      Request handle identification
IHB      Command executed in handle B
:WH      Request handle identification
RS232    Command executed in RS232 port
```

WT

Wait

Full Description

The WT command is a trippoint used to time events. When this command is executed, the controller will wait for the number of miliseconds specified before executing the next command.

Arguments

WT n where
n is an unsigned even number in the range 0 to 2000000000 (2 Billion)

Operand Usage

N/A

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In A Program	Yes
Command Line	No
Controller Usage	ALL
Default Value	-
Default Format	-

Related Commands

AT - At Time
TIME - Time Operand

Examples:

```
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
```

XQ

Execute Program

Full Description

The XQ command begins execution of a program residing in the program memory of the RIO board. Execution will start at the label or line number specified. Up to 4 programs may be executed simultaneously to perform multitasking.

Arguments

XQ #A,n XQm,n where
A is a program name of up to seven characters
m is a line number
n is the thread number (0 thru 3) for multitasking
NOTE: The arguments for the command, XQ, are optional. If no arguments are given, the first program in memory will be executed as thread 0.

Operand Usage

_XQn contains the current line number of execution for thread n, and -1 if thread n is not running.

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	0
Default Format	N/A

In a Program Yes
Command Line Yes
Can be Interrogated No
Used as an Operand Yes

Related Commands

HX - Halt execution

Examples:

```
XQ #Apple,0     Start execution at label Apple, thread zero
XQ #data,1      Start execution at label data, thread one
XQ 0    Start execution at line 0
```

ZC

User Variable, ZC

Full Description

ZC sets the first user variable. This variable provides a method for specific board information to be passed using the data record.

Arguments

ZCn where
n can be a number, RIO board operand, variable, mathematical function, or string; The range for numeric values is from -2,147,483,648 to +2,147,483,647. The maximum number of characters for a string is 4 characters. Strings are identified by quotations.
NOTE: If n is a '?,' the decimal value of ZC will be returned.

Usage

In a Program Yes
Command Line Yes
Can be Interrogated Yes
Used as an Operand Yes

Operand Usage

_ZC contains the decimal value of the user variable.

Related Commands

"ZD" Set second user variable

Examples:

```
ZC 2343 Sets the first user variable to a number (2343)
```

ZD

User Variable, ZD

Full Description

ZD sets the second user variable. This variable provides a method for specific board information to be passed using the data record.

Arguments

ZDn where
n can be a number, RIO board operand, variable, mathematical function, or string; The range for numeric values is from -2,147,483,648 to +2,147,483,647. The maximum number of characters for a string is 4 characters. Strings are identified by quotations.
NOTE: If n is a '?,' the decimal value of ZD will be returned.

Operand Usage

_ZD contains the decimal value of the user variable.

Usage

Usage and Default Details

Usage	Value
While Moving	No
In a Program	Yes
Command Line	Yes
Controller Usage	All
Default Value	
Default Format	

In a Program Yes
Command Line Yes
Can be Interrogated Yes
Used as an Operand Yes

Related Commands

ZC - Set first user variable

Examples:

```
ZD "INPT"       Sets the second user variable to the string "INPT"
```

ZS

Zero Subroutine Stack

Full Description

The ZS command is only valid in an application program and is used to avoid returning from an interrupt (either input or error). ZS alone returns the stack to its original condition. ZS1 adjusts the stack to eliminate one return. This turns the jump to subroutine into a jump. Do not use RI (Return from Interrupt) when using ZS. To re-enable interrupts, you must use II command again.
The status of the stack can be interrogated with the operand _ZSn - see operand usage below.

Arguments

ZS n where
n = 0 Returns stack to original condition
n = 1 Eliminates one return on stack

Operand Usage

_ZSn contains the stack level for the specified thread where n = 0 thru 3. The response, an integer between zero and sixteen, indicates zero for beginning condition and sixteen for the deepest value.

Usage

Usage and Default Details

Usage	Value
While Moving (No RIO)	Yes
In a Program	Yes
Command Line	No
Controller Usage	All
Default Value	0
Default Format	3.0

Related Commands

Examples:

```
II0,1,7;'       Input Interrupt on 7
#A;JP #A;EN;'   Main program
#ININT0;'       Input Interrupt
MG "INTERRUPT";'Print message
S=_ZS;'         Interrogate stack before ZS
S=;'            Print stack
ZS;'            Zero stack
S=_ZS;'         Interrogate stack after ZS
S=;'            Print stack
EN
```