USER MANUAL

RIO-47xxx

Manual Rev. 1.0i

By Galil Motion Control, Inc.

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

Rev Date 05/2010

Contents

Chapter 1 Overview

Introduction

Derived from the same fundamentals used in building the Galil motion controllers, the RIO-47xxx is a programmable remote I/O controller that conveniently interfaces with other Galil boards through its Ethernet port. The RIO is programmed exactly the same way as a DMC (Digital Motion Controller) with the exception of a few revised commands and the removal of all motion-related commands. Communication with the RIO even works the same way as with other Galil controllers, and it utilizes the same software programs. Interrogation commands have been included to allow a user to instantly view the entire I/O status, I/O hardware, or Ethernet handle availability (see the TZ, ID and TH commands).

The purpose of an RIO board is to offer remote I/O in a system and the ability to synchronize complex events. To do this, the RIO consists of two boards – a high speed processor with integrated Ethernet and an I/O board consisting of digital inputs, digital outputs, analog inputs, and analog outputs. If different I/O requirements are required –a custom I/O board can be made to mate up directly with the RIO processor.

Part Numbering Overview

	Highlights	Options
RIO-47100	Metal case 0-5V Analog IO 8 high power optoisolated digital outputs 8 low power optoisolated digital outputs 16 optoisolated digital inputs	-DIN -08-15 SOURCE -HS -422 -4-20mA -HC -PWM
RIO-47102	Same as RIO-47100 but with expanded memory	-DIN -08-15 SOURCE -HS -422 -4-20mA -RTC -HC -PWM
RIO-47120	Same as RIO-47100 but with +-10V analog IO	-DIN -08-15 SOURCE -16 Bit -422 -HS -4-20mA -HC -PWM
RIO-47122	Same as RIO-47120 but with expanded memory	-DIN -08-15 SOURCE -16 Bit -422 -HS -4-20mA -RTC -HC -PWM
RIO-47200	Same as RIO-47100 except -Screw terminals instead of D-Subs -Din rail mount with metal cover -All 16 outputs are high power -No analog outputs (use AO option to add analog)	-422 -0-7 SINK or -0-7 SOURCE -8-15 SINK or -8-15SOURCE -HS -10V12-Bit -10V16-Bit -NO DIN -4-20mA -HC -PWM (8AO_5v_12bit), (8AO_10v12bit),(8AO_10v16bit)

For details on all the options please see Standard Options in the Appendix.

RIO-47xx0 vs. RIO-47xx2

Controller	# of array elements	# of program lines	# of variables	# of labels	# of control loops	# of Ethernet handles
RIO-47xx0	400	200	126	62	2	3
RIO-47xx2	1000	400	256	126	6	5

RIO Functional Elements

Microcomputer Section

The main processing unit of the RIO is a specialized 32-bit Freescale Microcomputer with 32KB SRAM and 256KB of Embedded Flash memory. The SRAM provides memory for variables, array elements and application programs. The flash memory provides non-volatile storage of variables, programs, and arrays; it also contains the RIO firmware. The RIO can process individual Galil Commands in approximately 40 microseconds.

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

Communication

The communication interface with the RIO consists of one RS-232 port (default is 115 kBaud/s) and one 10/100Base-T Ethernet port (jumper configurable).

There are four status LEDs on the RIO that indicate operating and error conditions on the controller. Figure 1-1 shows a diagram of the LED bank followed by the description of the four lights.

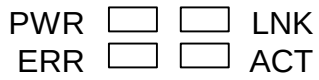


Figure 1-1 - Diagram of LED bank on the RIO

Green Power LED (PWR) - The green status LED indicates that the power has been applied properly to the RIO.

Red Status/Error LED (ERR) - The red error LED will flash on briefly at power up. After the initial power up condition, the LED will illuminate for the following reasons:

1. The reset line on the controller is held low or is being affected by noise.
2. There is a failure on the controller and the processor is resetting itself.
3. There is a failure with the output IC that drives the error signal.

Green Link LED (LNK) – The green LED indicates there is a valid Ethernet connection. This LED will show that the physical Ethernet layer (the cable) is connected.

Activity (ACT) – The amber LED indicates traffic across the Ethernet connection. This LED will show both transmit and receive activity across the connection.

Chapter 2 Getting Started

RIO-471xx

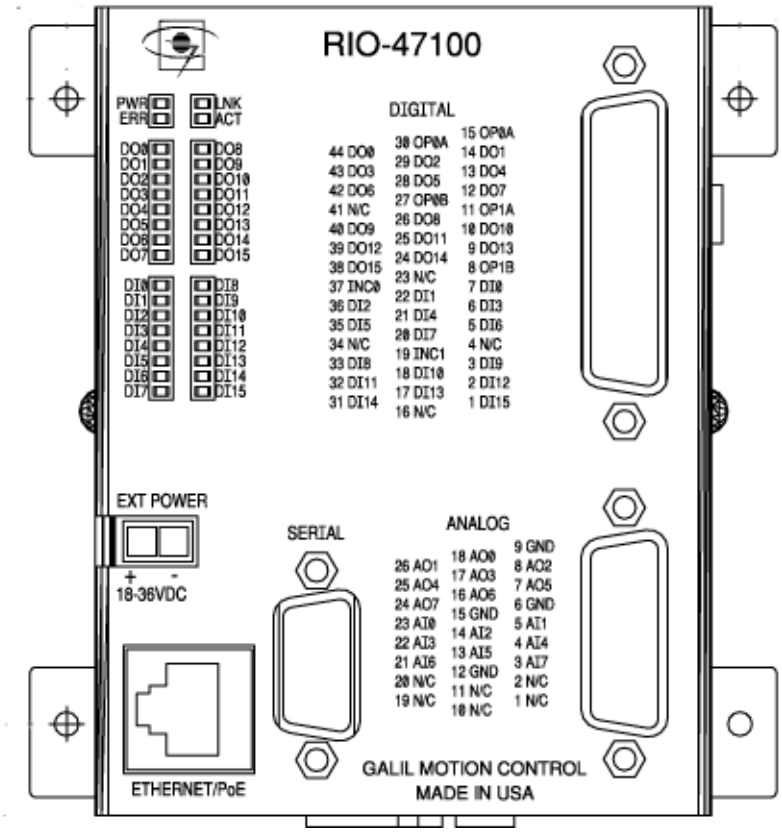


Figure 1: Outline of RIO-471xx (Dimensions listed in the Appendix)

RIO-472xx

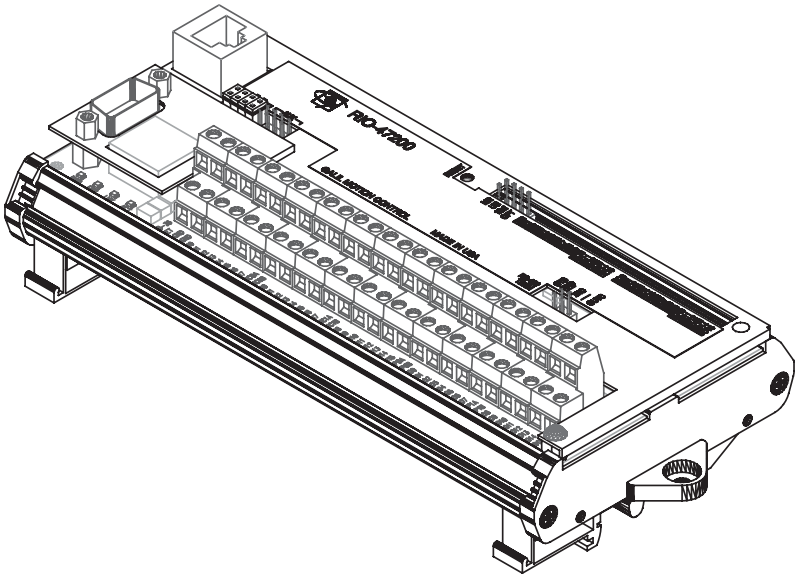


Figure 2: Outline of RIO-472xx (Dimensions listed in the Appendix)

Installing the RIO Board

Installation of a complete, operational RIO system consists of 4 steps:

Step 1. Configure jumpers

Step 2. Connect power to the RIO

Step 3. Install the communications software

Step 4. Establish communications between the RIO and the host PC

Step 1. Configure Jumpers

Power Input Jumpers (AUX vs PoE)

The RIO can be powered using either a 18-36V DC power input or a PoE (Power over Ethernet) switch to deliver power over the Ethernet cable. The default configuration is the 18-36VDC power input. If PoE is used, the **four** jumpers on JP6 for the RIO-471xx and JP1 for the RIO-472xx must be moved from AUX to PoE.

Master Reset and Upgrade Jumper

Jumpers labeled as MRST and UPGD are located at JP5 for the RIO-471xx and JP2 for the RIO-472xx, next to the reset button. The MRST jumper is for a master reset. When MRST is jumpered, the RIO will perform a master reset upon a power cycle to the board or when the board reset button is pushed. Whenever the I/O board has a master reset, all programs, arrays, and variables stored in non-volatile memory will be erased – **this will set the RIO board back to factory defaults**.

The UPGD jumper enables the user to unconditionally update the board firmware. This jumper is not necessary for firmware updates when the RIO board is operating normally, but may be necessary in cases of a corrupted non-volatile memory. non-volatile memory corruption should never occur under normal operating circumstances; however, corruption is possible if there is a power fault during a firmware update. If non-volatile memory corruption occurs, your board may not operate properly. In this case, install the UPGD jumper and use the update firmware function in the Galil software to re-load the system firmware.

Setting the Baud Rate on the RIO

The default baud rate for the RIO is 115K (jumper OFF).

The jumper labeled "19.2," also located at JP5 for the RIO-471xx and JP2 for the RIO-472xx, allows the user to select the serial communication baud rate. The baud rate can be set using the following table:

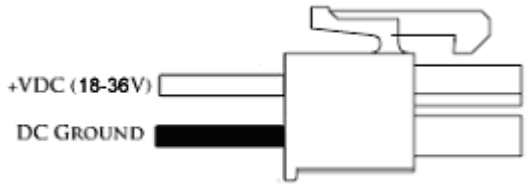
19.2	BAUD RATE
OFF	115k
ON	19.2k

Step 2. Connecting Power to the RIO

Since the RIO can be powered using either a 18-36V DC power input or a PoE (Power over Ethernet) switch, there are two possible connection options shown here:

1) EXT: 18-36VDC power input is the default configuration. The four jumpers on JP6 for the RIO-471xx and JP1 for the RIO-472xx . Apply a DC power supply in the range of 18-36V to the 2-pin molex connector. The power supply should be capable of delivering up to 4 Watts. The RIO uses Molex Pitch Mini-Fit, Jr.™ Receptacle Housing connectors for connecting DC Power. For more information on the connectors, go to http://www.molex.com/.

Note: The part number listed below is the connector that is found on the controller. For more information see the Molex website. http://www.molex.com/



Molex Part Number	Pin Part Number (x2)	Type
39-31-0020	44476-3112	2 Position

Warning: *Damage can occur if a supply larger than 36VDC is connected to the board.*

2) PoE: Power over Ethernet. This configuration needs the four jumpers on JP6 for the RIO-471xx and JP1 for the RIO-472xx to be placed on the side labeled PoE. Once this is done, the controller will derive its power directly from the Ethernet cable. A PoE style switch can be used such as the FS108P from Netgear.

Applying power will turn on the green LED power indicator.

Step 3. Install the Communications Software

After applying power to the computer, install the Galil software that enables communication between the I/O board and your PC. It is strongly recommended to use the Galil software "GalilTools" when communicating to the RIO unit. Please see the GalilTools Manual for a complete description of how to install and connect to Serial or Ethernet controllers.

http://www.galilmc.com/products/software/galiltools.html

Step 4. Establish Communications between RIO and the Host PC

Communicating to the RIO using Galil Software

RS-232:

To use serial communication, connect a 9pin **straight-through RS-232 cable** (CABLE-9-PIND) between the serial port of the RIO and the computer or terminal communications port. The RIO serial port is configured as DATASET.

Ethernet:

Connect the RIO Ethernet port to your computer via a crossover Ethernet cable, or to a network hub with a straight through Ethernet cable.

Using Non-Galil Communication Software

RS-232:

The RIO serial port is configured as DATASET. The computer or terminal must be configured as a for full duplex, no parity, 8 data bits, one start bit and one stop bit. A standard Windows HyperTerminal session can connect to the controller using a straight-through serial cable.

Check to insure that the baud rate jumpers have been set to the desired baud rate as described above. Also, the hardware handshake lines (RTS/CTS) need to be connected. See Chapter 3 for more information on 'Handshake Modes.'

Ethernet:

Connect the RIO Ethernet port to your computer via an Ethernet crossover cable, or to a network hub by a straight through Ethernet cable. An IP address needs to be assigned via a DHCP server, through Galil software, or via a serial cable using the IA command. See Chapter 3 for more information on how to establish an IP address. Once an IP address is established, a standard Windows Telnet session can connect to the controller.

* Note that the RIO-471x2 supports auto-crossover detection (auto MDIX)

Sending Test Commands to the Terminal after a successful Connection

After connecting to the computer or terminal, press <carriage return> or the <enter> key on the keyboard. In response to carriage return {CR}, the controller responds with a colon, :

Now type

TZ {CR}

This command directs the RIO to return the current I/O status. The controller should respond with something similar to the following:

```
: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)
0.0000,0.0000,0.0000,0.0000,0.0037,0.0012,0.0000,0.0000
Analog Outputs(7-0)
0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000
```

RIO Web Server

The RIO has a built-in web server that can be accessed by typing the IP address of the controller into a standard web browser. The controller comes from the factory without any IP address assigned so a user must go through the steps outlined above to establish an IP address before the web-server is accessible. Figure 3 shows an output of the RIO Web Server:

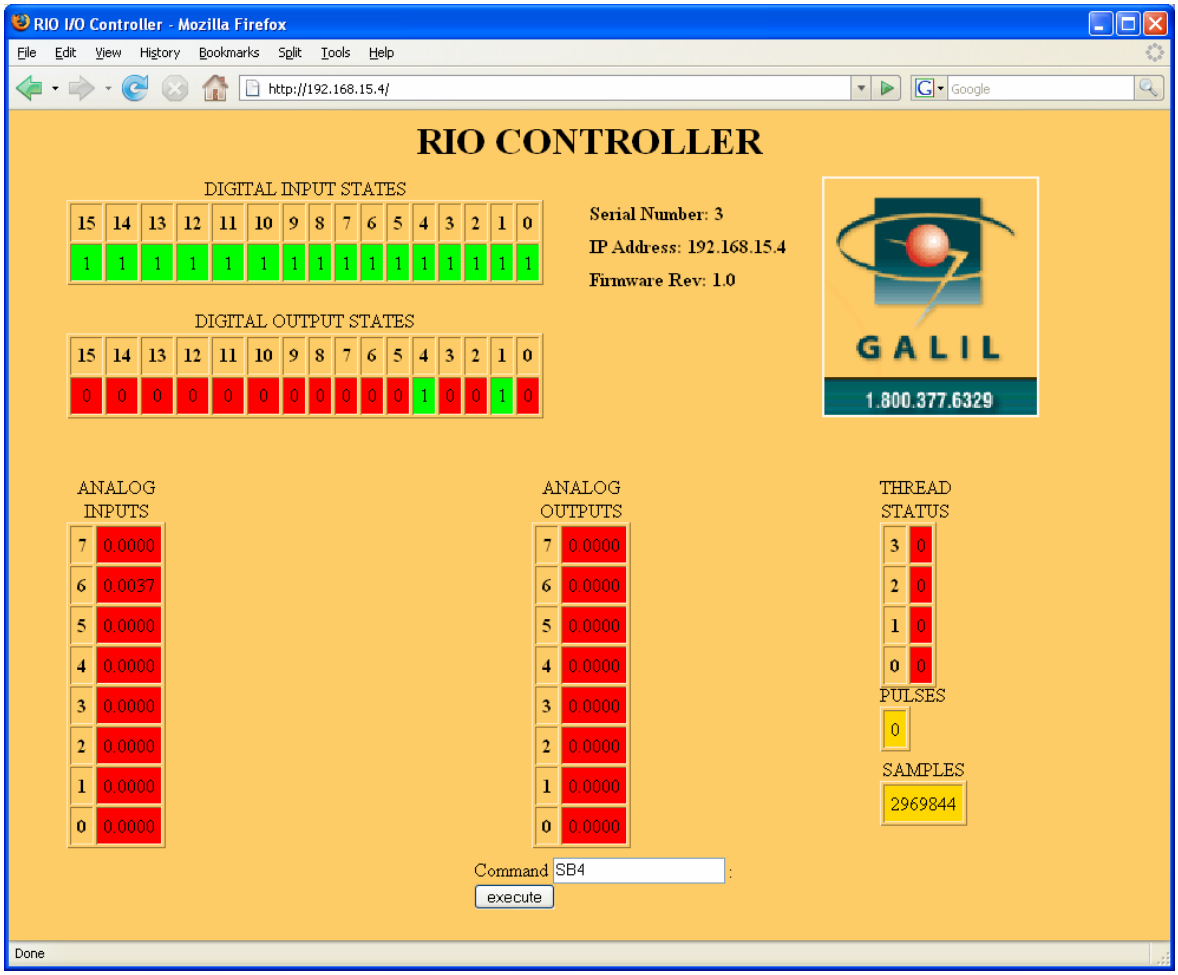


Figure 3: RIO Web Server Output

Chapter 3 Communication

Introduction

The RIO has one RS-232 port and one Ethernet port. The RS-232 port is the data set, and it is a standard serial link with a communication baud rate up to 115kbaud. The Ethernet port is jumper configurable for 10 or 100Base-T (default).

RS232 Port

The RIO board has a single RS232 connection for sending and receiving commands from a PC or other terminal. The pin-outs for the RS232 connection can be found in the Appendix - J5 Power: 2 pin Molex.

RS-232 Configuration

Configure the PC for 8 data bits, no parity, one stop bit, and hardware handshaking. The baud rate for the RS232 communication defaults to 115k baud but can be set to 19.2k baud by placing a jumper on J5. The serial port has a 4 bytes FIFO.

Handshaking Modes

The RS232 port is configured for hardware handshaking. In this mode, the RTS and CTS lines are used. The CTS line will go high whenever the RIO is not ready to receive additional characters. The RTS line will inhibit the RIO board from sending additional characters. **Note:** The RTS line goes high for inhibit. This handshake procedure is required and ensures proper communication especially at higher baud rates.

Ethernet Configuration

Communication Protocols

The Ethernet is a local area network through which information is transferred in units known as packets. Communication protocols are necessary to dictate how these packets are sent and received. The RIO supports two industry standard protocols, TCP/IP and UDP/IP. The board will automatically respond in the format in which it is contacted.

TCP/IP is a "connection" protocol. The master must be connected to the slave in order to begin communicating. Each packet sent is acknowledged when received. If no acknowledgement is received, the information is assumed lost and is resent.

Unlike TCP/IP, UDP/IP does not require a "connection". This protocol is similar to communicating via RS232. If a cable is unplugged, the device sending the packet does not know that the information was not received on the other side. Because the protocol does not provide for lost information, the sender must re-send the packet.

Galil recommends using TCP/IP for standard communication to insure that if a packet is lost or destroyed while in transit, it will be resent. However UDP is recommended in certain situations such as launching Data Record information to a host for graphing or data collection.

Each packet must be limited to 470 data bytes or less. This is not an issue when using Galil software as the Galil Ethernet driver will take care of the low level communication requirements.

The IK command blocks the controller from receiving packets on Ethernet ports lower than 1000 except for ports 0, 23, 25, 68, 80 and 502. To receive packets on all ports, set IK to 0.

NOTE: In order not to lose information in transit, Galil recommends that the user wait for an acknowledgement of receipt of a packet before sending the next packet.

Jumper Configuration for 10BaseT

If 10BaseT communication is required, a jumper must be placed on the pins labeled OPT. The default is no jumper which is 100BaseT Ethernet communication.

Addressing

There are three levels of addresses that define Ethernet devices. The first is the MAC or hardware address. This is a unique and permanent 6 byte number. No other device will have the same MAC address. The RIO MAC address is set by the factory and the last two bytes of the address are the serial number of the board. To find the Ethernet MAC address for a RIO unit, use the TH command. A sample is shown here with a unit that has a serial number of 3:

Sample MAC Ethernet Address: 00-50-4C-28-00-03

The second level of addressing is the IP address. This is a 32-bit (or 4 byte) number that usually looks like this: 192.168.15.1. The IP address is constrained by each local network and must be assigned locally. Assigning an IP address to the RIO board can be done in a number of ways.

The first method for setting the IP address is using a DHCP server. The DH command controls whether the RIO board will get an IP address from the DHCP server. If the unit is set to DH1 (default) and there is a DHCP server on the network, the controller will be dynamically assigned an IP address from the server. Setting the board to DH0 will prevent the controller from being assigned an IP address from the server.

The second method to assign an IP address is to use the BOOT-P utility via the Ethernet connection. The BOOT-P functionality is only enabled when DH is set to 0. Either a BOOT-P server on the internal network or the Galil software may be used. When opening the Galil Software, it will respond with a list of all RIO boards and controllers on the network that do not currently have IP addresses. The user must select the board and the software will assign the specified IP address to it. This address will be burned into the controller (BN) internally to save the IP address to the non-volatile memory. Note: if multiple boards are on the network – use the serial numbers to differentiate them.

CAUTION: Be sure that there is only one BOOT-P or DHCP server running. If your network has DHCP or BOOT-P running, it may automatically assign an IP address to the RIO board upon linking it to the network. In order to ensure that the IP address is correct, please contact your system administrator before connecting the I/O board to the Ethernet network.

The third method for setting an IP address is to send the IA command through the RS-232 port. (Note: The IA command is only valid if DH0 is set). The IP address may be entered as a 4 byte number delimited by commas (industry standard uses periods) or a signed 32 bit number (e.g. IA 124,51,29,31 or IA 2083724575). Type in BN to save the IP address to the RIO non-volatile memory.

NOTE: Galil strongly recommends that the IP address selected is not one that can be accessed across the Gateway. The Gateway is an application that controls communication between an internal network and the outside world.

The third level of Ethernet addressing is the UDP or TCP port number. The Galil board does not require a specific port number. The port number is established by the client or master each time it connects to the RIO board. Typical port numbers for applications are:

Port 23: Telnet
Port 502: Modbus
Port 80: HTTP

Email from the RIO

If the RIO is on a network with a SMTP Mail Server, the RIO is capable of sending an email message using the MG command. There are three configuration commands necessary to send an email from the RIO unit – MA, MS and MD. MA sets the smtp email server IP address. MS sets the email source or "from" address and MD sets the destination or "to" address. There is a maximum character limit for the MS and MD commands of 30 characters. An example of this is shown here:

```
MA 10,0,0,1;                ‘example SMTP Email Server IP address
MD someone@example.com;     ‘sample destination email address
MS me@example.com;          ‘sample source address
MG "Testing Email"{M};      ‘Message to send via Email
```

Please contact your system administrator for information regarding email settings.

Note: it is strongly recommended that the email messaging frequency is limited so as not to overload the email server.

Communicating with Multiple Devices

The RIO is capable of supporting multiple masters or slaves. A typical scenario would be connecting a PC (a master) and a motion controller (a 2nd master) that can both send commands to the RIO board over Ethernet on different handles.

Note: The term "master" is equivalent to the Internet "client" and the term "slave" is equivalent to the Internet "server".

An Ethernet handle is a communication resource within a device. The RIO-47xx0 can have a maximum of 3 Ethernet handles open at any time. This number is increased to 5 Ethernet handles on the RIO-47xx2. If all handles are in use and another device tries to connect, it will be sent a "reset packet" showing that the RIO cannot establish any new connections.

NOTE: A reset will cause the Ethernet connection to be lost. There are a number of ways to reset the board. Hardware resets (push reset button or power down RIO board) and software resets (through Ethernet or RS232 by entering the RS command).

When the RIO acts as the master, the IH command is used to assign handles and connect to its slaves. The IP address may be entered as a 4 byte number separated with commas (industry standard uses periods) or as a signed 32 bit number. A port number may also be specified, but if it is not, it will default to 1000. The protocol (TCP/IP or UDP/IP) to use must also be designated at this time. Otherwise, the board will not

connect to the slave. (Ex: IHB=151,25,255,9<179>2. This will open handle #2 and connect to the IP address 151.25.255.9, port 179, using TCP/IP)

Once the IH command is used to connect to slaves, the user can communicate to these slaves by sending commands to the master. The SA command is used for this purpose, and it has the following syntax.

SAh= "command string"

Here "command string" will be sent to handle h. For example, SAA=”XQ” command will send an XQ command to the slave/server on handle A. A more flexible form of the command is

SAh= field1,field2,field3,field4 ... field8

where each field can be a string in quotes or a variable.

When the Master/client sends an SA command to a Slave/server, it is possible for the master to determine the status of the command. The response _IHh4 will return the number 1 to 4. 1 indicates waiting for the acknowledgement from the slave. 2 indicates a colon (command accepted) has been received. 3 indicates a question mark (command rejected) has been received. 4 indicates the command timed out.

If a command generates multiple responses (such as the TE command), the values will be stored in _SAh0 thru _SAhn where n is the last field. If a field is unused, its _SA value will be -2^31.

See the Command Reference for more information on the SA command.

Which devices receive what information from the RIO depends on various things. If a device queries the RIO, it will receive the response unless it explicitly tells the RIO to send it to another device. If the command that generates a response is part of a downloaded program, the response will route to whichever port is specified by the CF command (either a specific Ethernet handle or the RS232 port). If the user wants to send the message to a port other than what is specified by the CF command, add an {Eh} or {P1} to the end of the command (Ex. MG{EB}"Hello" will send the message "Hello" to handle #2 and MG{P1}”Hello” will send it to the serial port).

Handling Communication Errors

A reserved automatic subroutine which is identified by the label #TCPERR can be used to catch communication errors. If an RIO has an application program running and the TCP communication is lost, the #TCPERR routine will automatically execute. The #TCPERR routine should be ended with the RE command.

Multicasting

A multicast may only be used in UDP/IP and is similar to a broadcast (where everyone on the network gets the information) but specific to a group. In other words, all devices within a specified group will receive the information that is sent in a multicast. There can be many multicast groups on a network and are differentiated by their multicast IP address. To communicate with all the devices in a specific multicast group, the information can be sent to the multicast IP address rather than to each individual device IP address. All Galil devices belong to a default multicast address of 239.255.19.56. This multicast IP address can be changed by using the IA>u command.

Unsolicited Message Handling

Unsolicited messages are any messages that are sent from the controller that are not directly requested by the host PC. An example of this is a MG or TP command inside of a program running on the controller. Error messages are also “unsolicited” because they can come out at any time. There are two software commands that will configure how the controller handles these unsolicited messages: CW and CF.

The RIO has 3 Ethernet handles as well as 1 serial port where unsolicited messages may be sent. The CF command is used to configure the controller to send these messages to specific ports. In addition, the Galil

software has various options for sending messages using the CF command. For more information, see the CF command description in the Command Reference.

The CW command has two data fields that affect unsolicited messages. The first field configures the most significant bit (MSB) of the message. A value of 1 will set the MSB of unsolicited messages, while a value of 2 suppresses the MSB. Programs like HyperTerminal or Telnet need to use a setting of CW2 for the unsolicited messages to be readable in standard ASCII format. However, the Galil software needs a value of CW1 to be set so that it can differentiate between solicited and unsolicited messages. If you have difficulty receiving characters from the controller, or receive garbage characters instead of messages, check the status of the CW command.

The second field of the CW command controls whether the product should pause while waiting for the hardware handshake to enable the transmission of characters over RS-232 (CW,0), or continue processing commands and lose characters until the hardware handshake allows characters to be sent (CW,1).\

Other Protocols Supported

Galil supports DHCP, ARP, BOOT-P, and Ping, which are utilities for establishing Ethernet connections. ARP is an application that determines the Ethernet (hardware) address of a device at a specific IP address. BOOT-P is an application that determines which devices on the network do not have an IP address and assigns the IP address you have chosen to it. Ping is used to check the communication between the device at a specific IP address and the host computer.

The RIO can communicate with a host computer through any application that can send TCP/IP or UDP/IP packets. A good example of this is Telnet, a utility that comes standard with the Windows operating system.

When using DHCP and a DNS (Domain Name Server), the DNS will assign the name “RIO47100-n” to the controller where n is the serial number of the unit.

Modbus with the RIO

The RIO-47xxx supports Modbus/TCP, and requires an Ethernet connection between its master or slave devices.

As a Modbus class 1 device, the RIO supports the following Modbus function codes:

Function Code	Modbus Description	Galil Description
1	Read Coil Status	Read Digital Outputs
2	Read Input Status	Read Digital Inputs
3	Read Holding Registers	Read Analog Inputs
4	Read Input Registers	Read Analog Outputs
5	Force Single Coil	Write Digital Output
6	Preset Single Register	Write Digital Outputs
7	Read Exception Status	Read Digital Outputs
15	Force Multiple Coils	Write Digital Outputs
16	Preset Multiple Registers	Write Analog Outputs

Of the Modbus function codes the RIO supports, all are supported by the RIO when it operates as a master (also known as a client) or when it operates as a slave (server).

Note 1: By default the RIO uses function code 3 for analog inputs and function code 4 for analog outputs. For a majority of Modbus devices this functionality is inverted. Use the MV command to switch the functionality. Please see the command reference for details.

Note 2: The remainder of this document uses the '$' symbol to signify that numbers are in hexadecimal notation.

Setup

Modbus/TCP requires an Ethernet connection between master and slave. Modbus/TCP also requires that all slaves communicate with their masters over port 502. See the IH command to setup port communication for the RIO.

Raw Modbus Send/Receive

Firmware revisions Rev D and newer support raw Modbus read/write functionality. This provides the user with the most flexibility for interfacing to modbus devices. Specifying a -1 for the Modbus function code enables the raw read/write of Modbus functions.

See the MB command in the RIO Command Reference for further details.

Modbus Read/Write to Array Table

Firmware revisions Rev D and newer support the ability to read from and write to array data on the RIO. Up to 1000 elements are available in the RIO-471x2 and 400 in the RIO-47xx0. Each element is accessible as a 16 bit unsigned integer (Modbus register 1xxx) -OR- as a 32 bit floating point number (Modbus registers 2xxx).

See the ME command in the RIO Command Reference for further details.

Sending Modbus Packets

The RIO programming language provides 3 ways of issuing Modbus packets as a master.

1) Issue the MB command of type Mbh = -1,len,array[]

> This Galil command allows the user complete control over the creation of their Modbus packet. len is the number of bytes to be included in the packet, and array[ ] is the name of the array containing the Modbus packet. 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.

2) Issue the MB command of type Mbh = addr, x, m, n, array[]

> This Galil command allows the user to send a Modbus command easily by allowing the user to select a few key parameters, and allowing the controller to do the rest. addr is the Unit ID field, which if not set, Galil will automatically set to the value of the handle the communication is over (Handle A=$01, B=$02, etc). Also, as a slave the RIO ignores the Unit ID field. x is the function code of the Modbus command. m is the address at which to begin reading or writing. n is either the number of coils or the number of registers to read/write. array[] is the array in which data from a read gets stored or where data to write is stored. See individual function code descriptions in the command reference for specifics of this command.

3) Issue another Galil command that supports Modbus

> The following Galil commands support Modbus, and are an easy way to use the Modbus protocol: SB,CB,AO,OB,@IN[],@OUT[],@AN[],@AO[]. The I/O number (variable) to use with these commands when using Modbus can be calculated as follows:
>
> I/O Number = (HandleNum*1000) + (bitNum)

Modbus Exceptions

An RIO configured as a slave will return an exception response if it receives an invalid request (e.g. An invalid function code, or a communication error). As a class 1 Modbus device the RIO-47xxx can respond with exception codes $01 or $02. Exception code $01 is returned when a request referencing an Illegal

Function is received. Exception code $02 is returned when a request referencing an Illegal Data Address is received.

When an Exception Response occurs, the function code of the response is $80 added to the original function code (e.g. Improper use of function code $01 will result in the exception response $81)

An RIO-47xxx configured as a master can query the function code of the last response it received using the _MW command (see command reference). The _MW command can be used to determine if an exception has occurred. The _MW1 command (see the command reference) can be used to query the exception code.

Function Code 1 ($01) - Read Coils

Description

Modbus function code $01 is a request to read coils. This will read digital outputs from an RIO configured as a slave.

Operating as a master

The function code of the response can be queried with the _MW command. If an exception occurred, the exception code of the response can be queried with _MW1.

Example:

Normal Response	Exception Response
_MW results in $01	_MW results in $81
	_MW1 contains $01 or $02

When using the MB command with Modbus function code 1, response data will be stored in the array referenced in the command line. When using @OUT[], @OUT[] contains the response data, which can either be stored to a variable or transmitted via serial port or ethernet.

Ways to use function code $01 with Galil commands:

1. MB command in raw packet mode
2. MB command with Modbus function code1
3. @OUT[] (see @OUT[] in the command reference)

Operating as a slave

The RIO will accept a read coils request with a starting address ranging from $0000-$000F, referencing digital outputs 0-15. The RIO will accept a request for up to all 16 of its digital outputs, with the quantity of coils ranging from $0001-$0010. The RIO will respond with function code $01 followed by a byte count of either $01 or $02, which describes the number of bytes of digital outputs being returned (byte count = quantity of outputs/8; if the remainder is not 0, byte count = quantity of outputs/8 +1). The RIO will respond with a coil status of 1 or 2 bytes (equal to the byte count) ranging from $0001-$FFFF, with each bit representing the state of a digital output (1 or 0). The LSB of the first coil status byte refers to the output addressed by the request packet.

Coil Mapping

Coil	Addresses	Coil	Addresses
0	Digital Output 0	8	Digital Output 8
1	Digital Output 1	9	Digital Output 9
2	Digital Output 2	10	Digital Output 10
3	Digital Output 3	11	Digital Output 11
4	Digital Output 4	12	Digital Output 12
5	Digital Output 5	13	Digital Output 13
6	Digital Output 6	14	Digital Output 14
7	Digital Output 7	15	Digital Output 15

Examples:

MBA= ,1,2,12,array[]	Request the status of coils 2-13 (result is stored in array[])
MG@OUT[1002]	Requests the status of coil 2 (result is transmitted via serial port or ethernet)

Packets

The command MBA=,1,2,10,array[] results in the following packets being sent, when one RIO is the master, and another RIO is the slave, communicating over handle A, port 502(Modbus). Assume digital outputs, in descending order from 15-0 are: 0,1,1,1,0,0,1,1,0,0,1,1,0,1,1,1

Request		**Response**	
Field Name	(hex)	Field Name	(hex)
Function	01	Function	01
Starting Address High	00	Byte Count	02
Starting Address Low	02	Outputs Status 9-2	CD
Quantity of Outputs High	00	Outputs Status 13-10	0C
Quantity of Outputs Low	0C		

1st Byte of Response Word

bit	7	6	5	4	3	2	1	0
Coil #	9	8	7	6	5	4	3	2
Value	1	1	0	0	1	1	0	1

2nd Byte of Response Word

bit	15	14	13	12	11	10	9	8
Coil #	X	X	X	X	13	12	11	10
Value	0	0	0	0	1	1	0	0

Note: bits in the response marked 'X' are not valid coil response data, but are instead 0's that fill the remainder of the byte

On the master RIO, array[0]=205 and array[1]=12 after the MBA= ,2,2,12,array[]command is issued

Function Code 2 ($02) - Read Discrete Inputs

Description

Modbus function code $02 is a request to read discrete inputs. This will read digital inputs from an RIO configured as a slave.

Operating as a master

The function code of the response can be queried with the _MW command. If an exception occurred, the exception code of the response can be queried with _MW1.

Example:

Normal Response	Exception Response
_MW results in $02	_MW results in $82
	_MW1 contains $01 or $02

When using the MB command with Modbus function code $02, response data will be stored in the array referenced in the command line. When using @IN[], @IN[] contains the response data, which can either be stored to a variable or transmitted via serial port or ethernet.

Ways to use function code 2 with Galil commands:

1. MB command in raw packet mode
2. MB command with Modbus function code 2
3. @IN[] (see @IN[] in the command reference)

Operating as a slave

The RIO will accept a read discrete inputs request with a starting address ranging from $0000-$000F, referencing digital inputs 0-15. The RIO will accept a request for up to all 16 of its digital inputs, with a quantity of inputs range of $0001-$0010.

The RIO will respond with a byte count of either $01 or $02, which describes the number of bytes of digital inputs being returned (byte count = quantity of inputs/8; if the remainder is not 0, byte count = quantity of inputs/8 +1). The RIO will respond with a input status of 1 or 2 bytes (equal to the byte count) ranging from $0001-$FFFF, with each bit representing the state of a digital input (1 or 0). The LSB of the first input status byte refers to the input addressed by the request packet.

Coil Mapping

Coil	Addresses	Coil	Addresses
0	Digital Input 0	8	Digital Input 8
1	Digital Input 1	9	Digital Input 9
2	Digital Input 2	10	Digital Input 10
3	Digital Input 3	11	Digital Input 11
4	Digital Input 4	12	Digital Input 12
5	Digital Input 5	13	Digital Input 13
6	Digital Input 6	14	Digital Input 14
7	Digital Input 7	15	Digital Input 15

Examples:

MBA= ,2,2,12,array[]	Request the status of discrete inputs 2-13 (result is stored in array[])
MG@IN[1002]	Requests the status of input 2 (result is transmitted via serial port or ethernet)

Packets:

The command MBA=,2,2,12,array[] results in the following packets being sent, when one RIO is the master, and another RIO is the slave, communicating over handle A, port 502(Modbus). Assume digital inputs, in descending order from 15-0 are: 0,1,1,1,0,0,1,1,0,0,1,1,0,1,1,1.

Request		**Response**	
Field Name	(hex)	Field Name	(hex)
Function	02	Function	02
Starting Address High	00	Byte Count	02
Starting Address Low	02	Inputs Status 9-2	CD
Quantity of Inputs High	00	Inputs Status 13-10	0C
Quantity of Inputs Low	0C		

1st Byte of Response Word

bit	7	6	5	4	3	2	1	0
Input #	9	8	7	6	5	4	3	2
Value	1	1	0	0	1	1	0	1

2nd Byte of Response Word

bit	15	14	13	12	11	10	9	8
Input #	X	X	X	X	13	12	11	10
Value	0	0	0	0	1	1	0	1

Note: bits in the response marked 'X' are not valid input response data, but are instead 0's that fill the remainder of the byte. Inputs report back a 0 when active and a 1 when inactive

On the master RIO, array[0]=205 and array[1]=12 after the MBA= ,2,2,12,array[] command is issued

Function Code 3 ($03) - Read Holding Registers

Description

Modbus function code $03 is a request to read holding registers. In its default configuration the RIO-471x0 responds to this command with analog input register information. To configure the RIO to respond to a function code 3 request with analog output information see the MV command in the command reference.

Operating as a master

The function code of the response can be queried with the _MW command. If an exception occurred, the exception code of the response can be queried with _MW1.

Example:

Normal Response	Exception Response
_MW results in $03	_MW results in $83
	_MW1 contains $01 or $02

When using the MB command with Modbus function code $03, response data will be stored in the array referenced in the command line. When using @AN[], @AN[] contains the response data, which can either be stored to a variable or transmitted via serial port or ethernet.

Ways to use function code 3 with Galil commands:

1. MB command in raw packet mode
2. MB command with Modbus function code 3
3. @AN[] (see @AN[] in the command reference)

Operating as a slave

The RIO will accept different starting address ranges for a read holding registers request depending on the state of the MI command. If MI is set to 0 (register data is volts in 32-bit floating point), the RIO will accept a read holding registers request with an address range of $0000-$000E. If MI is set to 1 (register data is counts in 16-bit decimal), The RIO will accept a read holding registers request with an address range of $0000-$0007. The RIO will accept a request with a quantity of registers field up to $0008 if MI is set to 0, and $00010 if MI is set to 1.

The RIO will respond with a byte count ranging from $0000 to $0020 if MI is 0, and from $0000 to $0010 if MI is 1 (Byte Count = 2*NumberOfRegisters, where NumberOfRegisters is equal to the number of analog inputs you are trying to read multiplied by 2 if MI is 0, or 1 if MI is 1). The RIO will respond with a byte count field equal to the byte count field in the request packet. The RIO will respond with a register value field consisting of either 2 bytes (counts) or 4 bytes (32-bit floating point) per analog input in ascending order from the analog input referenced in the address.

Galil Register Map

Register Address	32-Bit Floating Point	Counts
0	Analog Input 0	Analog Input 0
1		Analog Input 1
2	Analog Input 1	Analog Input 2
3		Analog Input 3
4	Analog Input 2	Analog Input 4
5		Analog Input 5
6	Analog Input 3	Analog Input 6
7		Analog Input 7
8	Analog Input 4	
9		
10	Analog Input 5	
11		
12	Analog Input 6	
13		
14	Analog Input 7	
15		

Examples:

MBA= ,3,2,4,array[]	Request the status of holding registers 2-5 (AN1 and AN2 if MI0, or AN2, AN3, AN4, AN5 if MI1). The response is stored in array[]
MG@AN[1002]	Requests the status of analog input 2 (result is transmitted via serial port or ethernet).

Packets:

The command MBA=,3,2,4,array[] results in the following packets being sent, when one RIO is the master, and another RIO-47100 is the slave, communicating over handle A, port 502(Modbus). When MI is set to 0 the response is given as volts in 32-bit Floating Point. When MI is set to 1 the response is given as counts in 16-bit decimal notation. Assume analog inputs in ascending order from 0-7 are: .4822, .9753, 1.4673, 1.9629, 2.4622, 2.9675, 3.4583, 3.9600

		Slave MI0			*Slave MI1*		
Request		**Response**			**Response**		
		32-bit Floating Point		Real Value	16-bit Integer		Real Value
Field Name	(hex)	Field Name	(hex)	(volts)	Field Name	(hex)	(counts)
Function	03	Function	03		Function	03	
Starting Address High	00	Byte Count	08		Byte Count	08	
Starting Address Low	02	RegVal2 High	3F	0.9753	RegVal2 High	25	9600
Quantity of Registers High	00		79		RegVal2 Low	80	
Quantity of Registers Low	04		B0		RegVal3 High	32	12904
		RegVal2 Low	00		RegVal3 Low	68	
		RegVal3 High	3F	1.4673	RegVal4 High	3F	16160
			BB		RegVal4 Low	20	
			D0		RegVal5 High	4C	19480
		RegVal3 Low	00		RegVal5 Low	18	

With the slave MI set to 0, the master RIO's arrays will look like this:

array[0]=16249
array[1]=45056
array[2]=16315
array[3]=53248

With the slave MI set to 1, the master RIO's arrays will look like this:

array[0]=9600
array[1]=12904
array[2]=16160
array[3]=19480

Function Code 4 ($04) - Read Input Registers

Description

Modbus function code $04 is a request to read input registers. In its default configuration the RIO-471x0 responds to this command with analog output register information. To configure the RIO to respond to a function code 4 request with analog input information see the MV command in the command reference.

Operating as a master

The function code of the response can be queried with the _MW command. If an exception occurred, the exception code of the response can be queried with _MW1.

Example:

Normal Response	Exception Response
_MW results in $04	_MW results in $84
	_MW1 contains $01 or $02

When using the MB command with Modbus function code $04, response data will be stored in the array referenced in the command line. When using @AO[], @AO[] contains the response data, which can either be stored to a variable or transmitted via serial port or ethernet.

Ways to use function code1 with Galil commands:

1. MB command in raw packet mode
2. MB command with Modbus function code 4
3. @AO[] (see @AO[] in the command reference)

Operating as a slave

The RIO will accept different address ranges for a read input registers request depending on the state of the MI command. If MI is set to 0 (register data is volts in 32-bit floating point), the RIO will accept a read input registers request with an address range of $0000-$000E. If MI is set to 1 (register data is counts in 16-bit decimal), The RIO will accept a read input registers request with an address range of $0000-$0007. The RIO will accept a request with a quantity of registers field up to $0008 if MI is set to 0, and $00010 if MI is set to 1. The RIO will respond with a byte count ranging from $0000 to $0010 if MI is 1, and from $0000 to $0020 if MI is 0 ((byte count = 2*NumberOfRegisters, where NumberOfRegisters is equal to the number of analog outputs you are trying to read multiplied by 2 if MI is 0, or 1 if MI is 1). The RIO will respond with an input registers field consisting of either 2 bytes (counts) or 4 bytes (32-bit floating point) per analog output register in ascending order from the analog output referenced in the address.

Galil Register Map

Register Address	32-Bit Floating Point	Counts
0	Analog Output 0	Analog Output 0
1		Analog Output 1
2	Analog Output 1	Analog Output 2
3		Analog Output 3
4	Analog Output 2	Analog Output 4
5		Analog Output 5
6	Analog Output 3	Analog Output 6
7		Analog Output 7
8	Analog Output 4	
9		
10	AnalogOutput 5	
11		
12	Analog Output 6	
13		
14	Analog Output 7	
15		

Examples:

MBA= ,4,2,4,array[]	Request the status of Registers 2-5 (AO1 and AO2 if MI0, and AO2, AO3, AO4, AO5 if MI1). The response is stored in array[]
MG@AO[1002]	Requests the status of analog output 2 (result is transmitted via ethernet or serial)

Packets:

The command MBA=,4,2,4,array[] results in the following packets being sent, when one RIO is the master, and another RIO-47100 is the slave, communicating over handle A, port 502(Modbus). When MI is set to 0 the response is given as volts in 32-bit Floating Point. When MI is set to 1 the response is given as counts in 16-bit decimal notation. Assume analog outputs in ascending order from 0-7 are: .5, 1, 1.5, 2, 2.5, 3, 3.5, 4

		Slave MI0			*Slave MI1*		
Request		**Response**			**Response**		
Field Name	(hex)	32-bit Floating Point Field Name	(hex)	Real Value (volts)	16-bit decimal Field Name	(hex)	Real Value (counts)
Function	04	Function	04		Function	04	
Starting Address High	00	Byte Count	08		Byte Count	08	
Starting Address Low	02	RegVal2 High	3F	1.0000	RegVal2 High	4C	19661
Quantity of Registers High	00		80		RegVal2 Low	CD	
Quantity of Registers Low	04		00		RegVal3 High	66	26214
		RegVal2 Low	00		RegVal3 Low	66	
		RegVal3 High	3F	1.5000	RegVal4 High	80	32768
			C0		RegVal4 Low	00	
			00		RegVal5 High	99	39321
		RegVal3 Low	00		RegVal5 Low	99	

With the slave MI set to 0, the master RIO's arrays will look like this:

array[0]=16256
array[1]=0
array[2]=16320
array[3]=0

With the slave MI set to 1, the master RIO's arrays will look like this:

array[0]=19661
array[1]=26214
array[2]=32768
array[3]=39321

Function Code 5 ($05) - Write Single Coil

Description

Modbus function code $05 is a request to write a single coil. This will write a digital output of an RIO configured as a slave.

Operating as a master

The function code of the response can be queried with the _MW command. If an exception occurred, the exception code of the response can be queried with _MW1.

Example:

Normal Response	Exception Response
_MW results in $05	_MW results in $85
	_MW1 contains $01 or $02

Ways to use Function Code 5 with Galil commands:

1. MB command in raw packet mode
2. MB command with Modbus function code 5
3. SB
4. CB
5. OB

Operating as a slave

The RIO will accept a write single coil request with a starting address ranging from $0000-$000F, referencing digital outputs 0-15.

The RIO will respond with a Modbus packet that is identical to the packet it received.

Coil Mapping

Coil	Addresses	Coil	Addresses
0	Digital Output 0	8	Digital Output 8
1	Digital Output 1	9	Digital Output 9
2	Digital Output 2	10	Digital Output 10
3	Digital Output 3	11	Digital Output 11
4	Digital Output 4	12	Digital Output 12
5	Digital Output 5	13	Digital Output 13
6	Digital Output 6	14	Digital Output 14
7	Digital Output 7	15	Digital Output 15

Examples:

For the following example, array[] contains [0,0,0,0,0,6,1,5,0,7,$FF,$00]

MBA= -1,12,array[] Request to set digital output 7 high

MBA=,5,7,1 Request to set digital output 7 high

SB1007 Request to set digital output 7 high

OB1007,@IN[1000] Request to set digital output 7 high if digital output 0 is high

Packets:

The command MBA=,5,7,1 results in the following packets being sent, when one RIO is the master, and another RIO is the slave, communicating over handle A, port 502(Modbus).

Request		**Response**	
Field Name	(hex)	Field Name	(hex)
Function	05	Function	05
Starting Address High	00	Starting Address High	00
Starting Address Low	07	Starting Address Low	07
Output Value High	FF	Output Value High	FF
Output Value Low	00	Output Value Low	00

As a result of the MB command above, the slave RIO will have output 7 turned on.

Function Code 6 ($06) - Preset Single Register

Description

Modbus function code $06 is a request to write to a single register. This will write all 16 digital outputs of an RIO configured as a slave.

Operating as a master

The function code of the response can be queried with the _MW command. If an exception occurred, the exception code of the response can be queried with _MW1.

Example:

Normal Response	Exception Response
_MW results in $06	_MW results in $86
	_MW1 contains $01 or $02

Ways to use function code 6 with Galil commands:

1. MB command in raw packet mode
2. MB command with Modbus function code 6

Operating as a slave

The RIO will accept a preset single register request with a starting address of $0000. The register values can range from 0x0000 to 0xFFFF and correspond to a binary representation of the 16 digital outputs. The RIO will respond with a Modbus packet that is identical to the packet it received.

Coil Mapping

Coil	Addresses	Coil	Addresses
0	Digital Output 0	8	Digital Output 8
1	Digital Output 1	9	Digital Output 9
2	Digital Output 2	10	Digital Output 10
3	Digital Output 3	11	Digital Output 11
4	Digital Output 4	12	Digital Output 12
5	Digital Output 5	13	Digital Output 13
6	Digital Output 6	14	Digital Output 14
7	Digital Output 7	15	Digital Output 15

Examples:

For the following example, array[] contains [0,0,0,0,0,6,1,6,0,0,$55,$AA]

MBA= -1,12,array[] Request to write digital outputs 15-0 to $55AA

MBA= ,6,0,$55AA Request to write digital outputs 15-0 to $55AA

Note: writing digital outputs 15-0 to $55AA results in digital outputs 15-0 in descending order, being 0,1,0,1,0,1,0,1,1,0,1,0,1,0,1,0.

Packets:

The command MBA= ,6,0,$55AA results in the following packets being sent, when one RIO is the master, and another RIO is the slave, communicating over handle A, port 502(Modbus).

Request		**Response**	
Field Name	(hex)	Field Name	(hex)
Function	06	Function	06
Starting Address High	00	Starting Address High	00
Starting Address Low	00	Starting Address Low	00
Register Value High	55	Register Value High	55
Register Value Low	AA	Register Value Low	AA

Function Code 7 ($07) – Read Exception Status

Description

Modbus function code $07 is a request to read the 8 exception status outputs. This will read digital outputs 0-7 of an RIO configured as a slave.

Operating as a master

The function code of the response can be queried with the _MW command. If an exception occurred, the exception code of the response can be queried with _MW1.

Example:

Normal Response	Exception Response
_MW results in $07	_MW results in $87
	_MW1 contains $01 or $02

When using the MB command with Modbus function code $07, response data will be stored in the array referenced in the command line.

Ways to use function code 7 with Galil commands:

1. MB command in raw packet mode
2. MB command with Modbus function code 7.

Operating as a slave

The RIO will accept a read exception status request. The RIO will respond with function code $07, and will return 1 byte of output data ranging from $00 to $FF, with each bit representing the state of a digital output (1 or 0). The LSB of the output data byte is digital output 0, and the MSB of the output data byte is digital output 7.

Coil Mapping

Coil	Addresses
0	Digital Output 0
1	Digital Output 1
2	Digital Output 2
3	Digital Output 3
4	Digital Output 4
5	Digital Output 5
6	Digital Output 6
7	Digital Output 7

Examples:

MBA= ,7,array[] Request to read exception status

Packets:

The command MBA= ,7,array[] results in the following packets being sent, when one RIO is the master, and another RIO is the slave, communicating over handle A, port 502(Modbus). Assume digital outputs, in descending order from 15-0 are:0,1,0,1,0,1,0,1,1,0,1,0,1,0,1,0. ($55AA)

Request		**Response**	
Field Name	(hex)	Field Name	(hex)
Function	07	Function	07
		Output Data	AA

array[0] on the master RIO will equal 170 in this example.

Function Code 15 ($0F) – Write Multiple Coils

Description

Modbus function code ($0F) is a request to write multiple coils. This will write multiple digital outputs to an RIO configured as a slave.

Operating as a master

The function code of the response can be queried with the _MW command. If an exception occurred, the exception code of the response can be queried with _MW1.

Example:

Normal Response	Exception Response
_MW results in $0F	_MW results in $8F
	_MW1 contains $01 or $02

Ways to use function code 15 with Galil commands:

1. MB command in raw packet mode
2. MB command with Modbus function code 15

Operating as a slave

The RIO will accept a write multiple coils request with a starting address ranging from $0000-$000F, referencing digital outputs 0-15. The RIO will accept a request for up to all 16 of its digital outputs, or $0001-$0010.

The RIO will respond with function code $0F, a starting address field which matches the starting address field of the request packet, and a quantity of outputs which matches the quantity of outputs field of the request packet.

Coil Mapping

Coil	Addresses	Coil	Addresses
0	Digital Output 0	8	Digital Output 8
1	Digital Output 1	9	Digital Output 9
2	Digital Output 2	10	Digital Output 10
3	Digital Output 3	11	Digital Output 11
4	Digital Output 4	12	Digital Output 12
5	Digital Output 5	13	Digital Output 13
6	Digital Output 6	14	Digital Output 14
7	Digital Output 7	15	Digital Output 15

Examples:

For the following example, array[] contains [0,0,0,0,0,9,1,15,0,0,0,16,2,$AA,$55]

MBA= -1,15,array[] Request to write $AA55 to digital outputs 15-0

For the following example, array[] contains [$AA55]

MBA= ,15,0,16,array[] Request to write $AA55 to digital outputs 15-0

Packets:

The command MBA= ,15,0,16,array[] (when array contains [$AA55]) results in the following packets being sent, when one RIO is the master, and another RIO is the slave, communicating over handle A, port 502(Modbus). The slave RIO's outputs 15-0 will be set to the following (1 is on 0 is off):

Output	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	0	1	0	1	0	1	0	1	1	0	1	0	1	0	1	0

Request		**Response**	
Field Name	(hex)	Field Name	(hex)
Function	15	Function	15
Starting Address High	00	Starting Address High	00
Starting Address Low	00	Starting Address Low	00
Quantity of Outputs High	00	Quantity of Outputs High	00
Quantity of Outputs Low	10	Quantity of Outputs Low	10
Byte Count	02		
Outputs Value High	AA		
Outputs Value Low	55		

Function Code 16 ($10) – Write Multiple Registers

Description

Modbus function code ($10) is a request to write multiple registers, also known as analog outputs

Operating as a master

The function code of the response can be queried with the _MW command. If an exception occurred, the exception code of the response can be queried with _MW1.

Example:

Normal Response	Exception Response
_MW results in $10	_MW results in $90
	_MW1 contains $01 or $02

Ways to use function code 16 with Galil commands:

1. MB command in raw packet mode
2. MB command with Modbus function code 16
3. AO[x] See command reference for details

Note: The RIO acting as a master can write up to 123 registers at a time with function code 16 per the Modbus specification.

The Modbus transaction results are available with the _MW and _MW1 commands.

Operating as a slave

The RIO will accept different starting address ranges for a write multiple registers request depending on the state of the MI command. If MI is set to 0 (register data is volts in 32-bit floating point), the RIO will accept an address range of $0001-$000E. If MI is set to 1 (register data is count in 16-bit decimal), the RIO will accept a write multiple registers request with an address range of $0000-$0007. The RIO will respond with function code 16, a 2 byte starting address field identical to the starting address field of the request packet, and a 2 byte quantity of registers field identical to the quantity of registers field of the request packet.

Galil Register Map

Register Address	32-Bit Floating Point	Counts
0	Analog Output 0	Analog Output 0
1		Analog Output 1
2	Analog Output 1	Analog Output 2
3		Analog Output 3
4	Analog Output 2	Analog Output 4
5		Analog Output 5
6	Analog Output 3	Analog Output 6
7		Analog Output 7
8	Analog Output 4	
9		
10	Analog Output 5	
11		
12	Analog Output 6	
13		
14	Analog Output 7	
15		

Examples:

For the following example, array[] contains [0,0,0,0,0,15,1,16,0,2,0,4,8,64,160,0,0,64,64,0,0]

MBA= -1,21,array[] Request to write 5V to analog output 1 and 3V to analog output 2

For the following example, array[] contains [$40A0, $0000, $4040, $0000] ($40A00000 is 32-bit Floating Point for 5.0000 decimal and $40400000 is 32-bit Floating Point for 3V decimal)

MBA= ,16,2,4,array[] Request to write 5V to analog output 1 and 3V to analog output 2

AO1001,5 Request to write 5V to analog output 1