USER MANUAL

RIO-47xxx

Manual Rev. 1.0r

Galil Motion Control, Inc.

270 Technology Way
Rocklin, California

916.626.0101
support@galilmc.com
galil.com

04/2016

Contents

Chapter 1 Overview

Introduction

Derived from the same fundamentals used in building 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 for a system and to provide 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

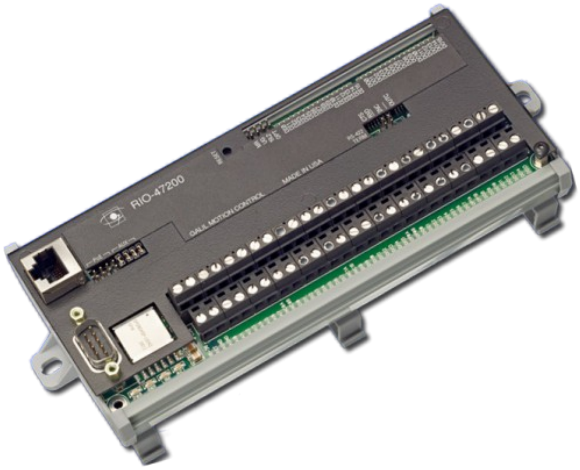
The RIO-47xxx has three distinct packaging types, the RIO-471xx, RIO-4720x, and RIO-47300. Each packaging type has its "base" model where different variations (xx) and additional - Standard Options (yyy) can be ordered. For instance, a full part number would follow the format: RIO-47xxx-yyy, such as RIO-47122-422-HS. Note: multiple - Standard Options (yyy) can be ordered per RIO. Table 1.1 below describes the RIO and its options. For in depth details regarding the - Standard Options (yyy), please see the Ordering Options section in the Appendix.

For full part number information of the RIO product line, see the RIO part number generator:
http://www.galil.com/order/part-number-generator/rio-47xxx#RIO-47100

R I O - 4 7 1 x x - y y y

Base Model	Features	- yyy Standard Options
x x = 0 0	8 0-5V analog inputs 8 0-5V analog outputs 8 500mA sourcing optoisolated digital outputs 8 25mA sinking optoisolated digital outputs 16 optoisolated digital inputs	-422 -DIN
Additional Options	**Additional Features**	**Additional – yyy Options**
x x = 0 2	Base plus: *Expanded Memory*	Base plus: -RTC -(4-20mA) -HS
x x = 2 0	Base plus: Analog I/O upgraded to ±10V configurable	Base plus: -16Bit
x x = 2 2	Base plus: *Expanded Memory* Analog I/O upgraded to ±10V configurable	Base plus: -16Bit -RTC -(4-20mA) -HS -PWM -QUAD, -SSI, and -BiSS
x x = 4 2	Base plus: *Expanded Memory* Analog I/O upgraded to ±10V configurable Dual Ethernet ports, no PoE All 16 outputs upgraded to 500mA sourcing	Base plus: -16Bit -RTC -(4-20mA) -HS -PWM -QUAD, -SSI, and -BiSS -12V option not available. -2LSRC Option not available.

R I O - 4 7 2 x x - y y y



Base Model	Features	- yyy Standard Options
x x = 0 0	Screw-terminal connectors Din-rail mount with metal cover No analog outputs by default (Use – Y Y Y options to add Analog) 16 500mA sourcing optoisolated digital outputs 16 optoisolated digital inputs	-422 -NO DIN - -
Additional Options	**Additional Features**	**Additional – yyy Options**
x x = 0 2	Base plus: *Expanded Memory*	Base plus: -RTC -HS -(4-20mA) -1LSNK/1LSRC & 2LSNK/2LSRC -PWM -(AI_10v12bit), (AI_10v16bit) -(8AO_10v12bit), (8AO_10v16bit), (8AO_5v12bit) -QUAD, -SSI, and -BiSS

R I O - 4 7 3 x x - y y y

Base Model	Features	- yyy Standard Options
x x = 0 0	Screw-terminal connectors Din-rail mount with metal cover 8 ±10V configurable analog inputs 8 ±10V configurable analog outputs 24 500mA sourcing optoisolated digital outputs 24 optoisolated digital inputs *Expanded Memory*	-422 -NO DIN -16Bit -HS -(4-20mA) -PWM -RTC -QUAD, -SSI, and -BiSS -24ExIn & -24ExOut

Table 1.1: RIO-47xxx Part Number Features and – yyy Standard Options

Standard vs. *Expanded Memory*

Feature	Standard	Expanded
Variable Range	± 2 billion	± 2 billion
Variable Resolution	1×10^{-4}	1×10^{-4}
# of array elements	400	1000
# of program lines	202	402
# of variables	126	254
# of labels	62	126
# of control loops	2	6
# of Ethernet handles	3	5
Auto MDIX	NO	YES
10/100 Mbits/s	100 Mbit/s Standard (10 Mbit/s with jumper added)	Auto-negotiated
Real-time Clock	NO	YES (See -RTC for extra capabilities)
Max # of Burn Cycles	10000	10000

Table 1.2: Feature differences between Standard and Expanded Memory options

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 (speed is jumper configurable with RIO-47xx0). The RIO-47142 and RIO-47300 have an integrated switch with dual Ethernet ports.

Status LEDs

There are four status LEDs on the RIO that indicate operating and error conditions on the controller. Figure 1.1 and Figure 1.2 shows a diagram of the LED bank followed by the description of the four lights on the RIO-471xx (except for the RIO-47142) and the RIO-472xx.

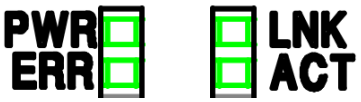


Figure 1.1: Diagram of LED bank on the RIO-471xx

Figure 1.2: Diagram of LED bank on RIO-472xx

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.

RIO-47142/RIO-47300 Status LEDs

There are two status LEDs on the RIO-47142 (PWR and ERR) that indicate operating and error conditions on The PWR and ERR description are identical to that of the RIO products listed above.

On the each Ethernet port there are two LEDs that indicate the status of the port's Ethernet connection.

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. This LED will also blink to show both transmitted and received activity across the connection.

Orange LED (SPD) – The orange LED indicates the speed of the Ethernet connection. It will be illuminated for a 100bT connection, and will be off for a 10bT connection.

Chapter 2 Getting Started

Layout

RIO-4710x and RIO-4712x

The mechanical layout and dimensions are the same for the RIO-4710x and RIO-4712x products (RIO-47100, RIO-47122 etc).

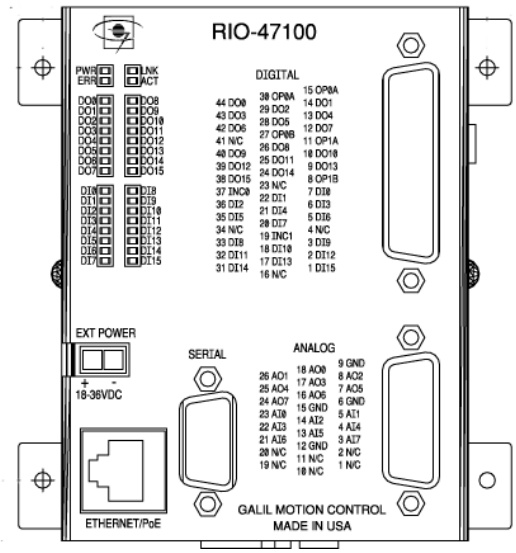


Figure 2.1: Outline of RIO-47100. Dimensions listed in the Appendix under: RIO Dimensions

RIO-47142

The RIO-47142 has similar mechanical dimensions to the RIO-4710x and RIO-4712x products. The main difference is the dual Ethernet switch integrated into the RIO.

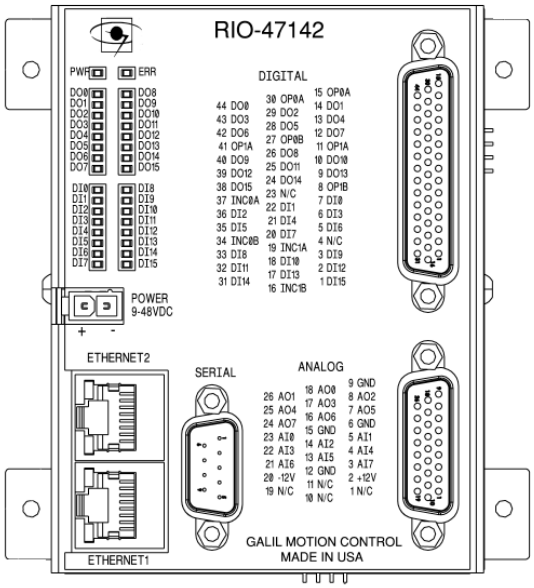


Figure 2.2: Outline of the RIO-47142. Dimensions listed in the Appendix under: RIO Dimensions

RIO-4720x

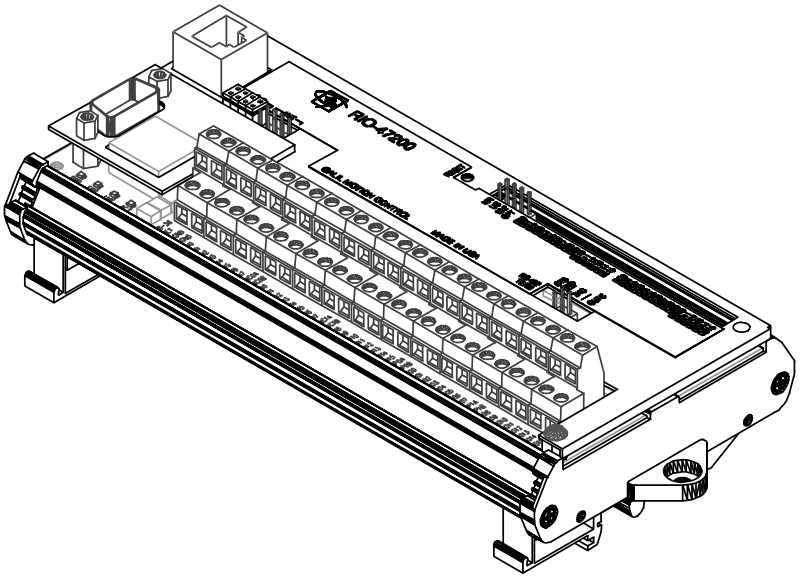


Figure 2.3: Outline of RIO-4720x. Dimensions listed in the Appendix under: RIO Dimensions

RIO-47300

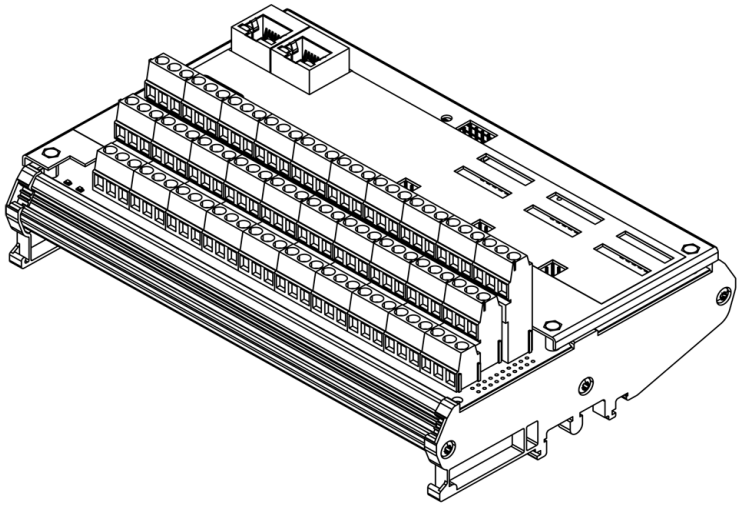


Figure 2.4: Outline of the RIO-47300. Dimensions listed in the Appendix under: RIO Dimensions.

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 (EXT/AUX vs PoE)

The RIO can be powered using either an external DC power input or a PoE (Power over Ethernet) switch to deliver power over the Ethernet cable. By default, the RIO is expecting the use of an external power supply and four jumpers are placed on either the "EXT" or "AUX" pins depending on the RIO model. If PoE is desired instead, move the **four** jumpers from the pins labeled "EXT/AUX" to the pins labeled "PoE." A full description of the "PoE" and "AUX/EXT" jumpers are in the Appendix listed under Jumper Descriptions.

Not all RIO models have the PoE option. For more information PoE options and which DC power supply is appropriate for your model, continue to Step 2. Connect Power to the RIO.

Master Reset and Upgrade Jumper

The MRST jumper is for a master reset. When MRST is jumped, the RIO will perform a master reset either when the board reset button is pressed or the controller is power cycled. Whenever the I/O board has a master reset, all parameters, 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, connect over RS232, and use the update firmware function in the Galil software to re-load the system firmware.

The location for the jumpers are in the Appendix listed under Jumper Descriptions.

Setting the Baud Rate on the RIO

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

The jumper labeled "19.2," 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

The location for the jumpers are in the Appendix listed under Jumper Descriptions.

Step 2. Connect Power to the RIO

Most RIO models can be powered using either an auxiliary DC power supply or a PoE (Power over Ethernet) switch. These power options are selected by the user by placing four jumpers on either the "PoE" or "EXT/AUX" labels. See Step 1. Configure Jumpers for a full description of these jumpers. Once these jumpers are configured and power is properly applied based upon this selection, the green PWR LED will turn on.

PoE configurations will allow the RIO to derive its power directly from the Ethernet cable—no additional connections are necessary for powering. Any PoE style switch can be used, such as the FS108P from Netgear.

In contrast, the "EXT/AUX" configuration will allow the RIO to derive it's power from an auxiliary power source either through a 2-pin Molex connector or designated screw terminals (depending on the model). The power supply used should be capable of delivering 4 Watts of power. For more information on power specifications, see the Appendix for Power Requirements for EXT/AUX Power Option.

Table 2.1 below depicts the different power options, voltage requirements, and DC power connector type. Note: Not all models have PoE capabilities as shown below.

Power Connection Options

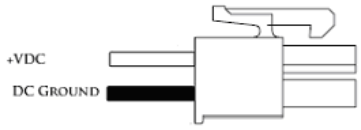
Model	Power over Ethernet (Jumpers on PoE)	DC Power Input[1] (Jumpers on EXT/AUX)	DC Power Connector Type
RIO-47100	YES	18-36 VDC	2-pin Molex [2]
RIO-47102	YES	18-36 VDC	2-pin Molex [2]
RIO-47120	YES	18-36 VDC	2-pin Molex [2]
RIO-47122	YES	18-36 VDC	2-pin Molex [2]
RIO-47142	NO	9-48 VDC	2-pin Molex [2]
RIO-47200	YES	18-36 VDC	Screw Terminals [3]
RIO-47202	YES	18-36 VDC	Screw Terminals [3]
RIO-47300	NO	9-48 VDC	Screw Terminals [3]

Table 2.1: Power Connection Options

[1]**Warning – Damage will occur of improper voltage is applied to the RIO.** Do not supply voltages larger than the indicated maximum. See the Appendix for Power Requirements for EXT/AUX Power Option.

[2]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 found *on the controller.*



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

[3]See the Appendix for your RIO PLC for the appropriate power pin-outs in the Connectors for RIO-47xxx.

Step 3. Install the Communications Software

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

http://www.galil.com/download/manual/galilsuite/galilsuite.pdf

Step 4. Establish Communications between RIO and the Host PC

Ethernet

For non-Auto MDIX RIO models[1], 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's software, or via a serial cable using the IA command. See Chapter 3 Communication for more information on how to establish an IP address. Once an IP address is established, the user can communicate to the controller either using GalilSuite's Terminal or even a simple Windows Telnet session can connect to the controller.

[1]Please refer to Table 3.1 for a full description of your RIO's Ethernet capabilities. Auto-MDIX RIO models can use either a straight-through or cross-over cable.

RS-232

To use serial communication, connect a 9-pin straight-through RS-232 cable (Part number: CABLE-9-PIND) between the serial port of the RIO and the computer or terminal communications port. The computer or terminal must be configured as described in Table 2.2 below. Galil's communication software is already configured for this, and thus, an unnecessary step if using Galil software.

Port Setting	Required RIO Configuration
Data Bits	8
Parity	None
Start Bits	1
Stop Bits	1
Flow Control	Hardware

Table 2.2: Required Port Settings to communicate to an RIO using RS232

Check to insure that the baud rate jumpers (See Jumper Descriptions) have been set to the desired baud rate as you're trying to connect with. Also, the hardware handshake lines (RTS/CTS) need to be connected.

At this point the user can connect either using Galil software or a standard Windows HyperTerminal session.

See Chapter 3 Communication for more information on 'Handshake Modes.'

Sending Test Commands to the Terminal After a Successful Connection

After connecting to a 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 2.5 shows an output of the RIO Web Server.

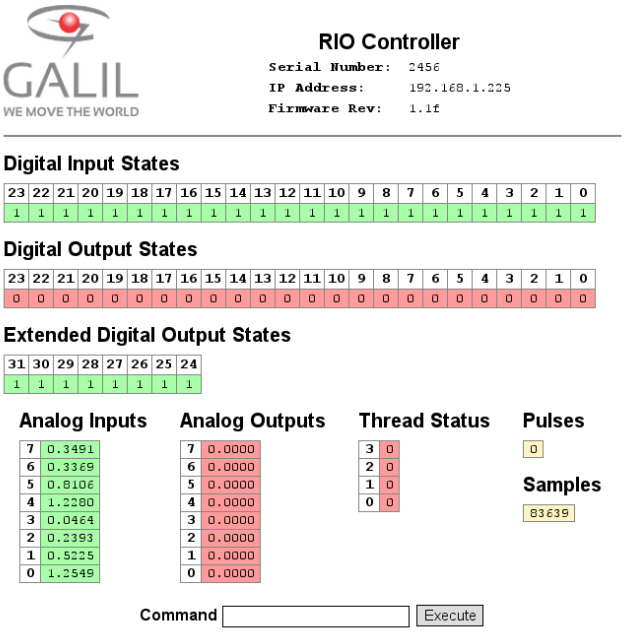


Figure 2.5: RIO Web Server Output

Chapter 3 Communication

Introduction

The RIO has one RS-232 port and one Ethernet port. The RIO is capable of 100bT or a 10bT Ethernet connection. The RIO-47142 and RIO-47300 have dual port Ethernet switches.

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 - Power: J5, 2-pin Molex.

RS-232 Configuration

Configure the PC for 8 data bits, no parity, one stop bit, and hardware handshaking as shown in Table 2.2. 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 upon connection.

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 acknowledgment 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 acknowledgment of receipt of a packet before sending the next packet.

Ethernet Capabilities by Model

Model	Auto MDIX	Dual-Port	10/100 Mbits/s
RIO-47100	NO	NO	100 Mbits standard, 10 Mbits w/jumper installed
RIO-47102	YES	NO	Auto-negotiate
RIO-47120	NO	NO	100 Mbits standard, 10 Mbits w/jumper installed
RIO-47122	YES	NO	Auto-negotiate
RIO-47142	YES	YES	Auto-negotiate
RIO-47200	NO	NO	100 Mbits standard, 10 Mbits w/jumper installed
RIO-47202	YES	NO	Auto-negotiate
RIO-47300	YES	YES	Auto-negotiate

Table 3.1: Ethernet Capabilities by RIO Part Number

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; an example of such is: 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 3 different ways.

1. 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.

2. The second method to assign an IP address is to use Galil Software. This procedure will vary depending on the software package. Refer to the user manual for the different software packages that support the RIO-47xxx.
3. 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) (e.g. IA 124,51,29,31). 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.

Note: if multiple boards are on the network – use the serial numbers to differentiate them.

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 and RIO-47300. 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 communication 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.

The RIO can communicate through its different channels, which can be tightly controlled. When a device queries the RIO, it will receive the response unless it explicitly tells the RIO to send it to another device. When a command that generates an unsolicited 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 multiple 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

An additional protocol layer is available for speaking to I/O devices. Modbus is an RS-485 protocol that packages information in binary packets that are sent as part of a TCP/IP packet.

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.

The Modbus protocol has a set of commands called function codes. As a Modbus Class 1 device, the RIO-47xxx supports the 10 major function codes:

Function Code	Modbus Description
1	Read Coil Status
2	Read Input Status
3	Read Holding Registers
4	Read Input Registers
5	Force Single Coil
6	Preset Single Register
7	Read Exception Status
15	Force Multiple Coils
16	Preset Multiple Registers

Table 3.2: Supported Modbus function codes and descriptions.

All modbus function codes listed in Table 3.2 are supported by the RIO when it operates as a master (client) or when it operates as a slave (server).

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

As a Modbus Master

The RIO-47xxx provides three method of Modbus communication as a master:

1. The first method of Modbus communication uses standard Galil commands. The following commands support Modbus:

   @IN[], @AN[], @OUT[], SB, CB, OB, and AO

   See the RIO command reference individually for each command on how to address a Modbus slave.

2. The second method uses the MB command that requires the user to enter a select a few key parameters and allows the controller to build the Modbus packet. The formats vary depending on the function code that is called. For more information refer to the MB command in Command Reference

3. The final method allows the user to send raw Modbus packets. This gives the user complete control over the creation of their Modbus packet including transaction identifiers, protocol identifiers, length field, Modbus function code, and data specific to that function code.. To send a raw Modbus packet the user must set the MB command with a function code of –1. For more information refer to the MB command in Command Reference

The following sections provide a full description of each method.

Example #1 – Using Methods 1 and 2 to set Outputs

This example shows how a RIO master might connect to another slave RIO and toggle it's outputs on using Method 1 and Method 2 described above.

Method 1

1. Begin by opening a connection to the slave assuming it has the IP address 192.168.1.120. This command would be issued to the RIO Master:

```
IHB=192,168,1,120<502>2
```

This command opens a Modbus connection to the slave on handle “B”.

2. Set the outputs using the SB command, these commands are issued to the RIO Master:

```
SB 2001;SB 2003;SB 2005;SB 2007
```

Note that the “2000” designates the command is to the Modbus slave connected through handle “B” and that slave's outputs 1, 3, 5, and 7 are toggled “on” by the SB command from the master.

Method 2

Assume that the handle is still open from the “IHB” command above, but that the outputs are now turned off. It is desired to set the same outputs but instead using the MB command.

1. Dimension an array to store the commanded values. Set array element 0 equal to 170. Note that array element 0 configures digital outputs 7-0. The following commands would be sent to the RIO master:

```
DM myarray[2]
myarray[0]= 170
```

Note: 170 is 10101010 in binary meaning outputs 1, 3, 5, and 7 would be toggled “on” where the remaining would be turned “off” or remain “off”.

2. Send the appropriate MB command. Use function code 15 for setting outputs. The following command would be issued to the RIO master:

```
MBB=,15,0,16,myarray[]
```

Both 1 and 2 will result in the same outputs being toggled. The only difference is that Method 2 uses a single command to toggle all outputs both *on and off*; Whereas, SB can only toggle and individual output on at a time and CB is used to turn those outputs off.

Example #2 – Using Method 2 to read analog inputs

Assume an RIO master is connecting as a Modbus master to a 3rd party PLC. The RIO will read the value of analog inputs 3 and 4 on the PLC located at addresses 40006 and 40008, respectively. Address information can be found in the PLC manufacturers documentation. Assume the PLC stores values as 32-bit floating point numbers, which is common.

1. Begin by opening a connection to the PLC which has an IP address of 192.168.1.10. This command is issued on the RIO master:

```
IHB=192,168,1,10<502>2
```

2. Dimension an array to store the results, this command is issued to the RIO master:

```
DM myanalog[4]
```

3. Send the appropriate MB command: Use function code 4 (this is specified by the PLC as well). Start at address 40006. Retrieve 4 Modbus registers. Note: There are *two* Modbus registers per a single analog input using 32-bit numbers.

MBB=,4,40006,4,myanalog[]

Array elements 0 and 1 will make up the 32-bit floating point value for analog input 3 and array elements 2 and 3 will combine for the value of analog input 4. The array myanalog[] will store these values in decimal format that requires converting to get it into volts. Assume the following array elements return with these respective decimal values:

Analog input	Array Element	Value Stored [decimal]	Hex Conversion	Full 32-bit value
Analog input 3	myanalog[0]	16412	$401C	$401CCCCD
	myanalog[1]	52429	$CCCD	
Analog input 4	myanalog[2]	49347	$C0C3	$C0C33333
	myanalog[3]	13107	$3333	

The 32-bit hex value must then be converted into volts:

Analog input 3 = $401CCCCD = 2.45 V

Analog input 4 = $C0C33333 = -6.1 V

As a Modbus Slave

Function Code	RIO Slave Description
1	Read Digital Outputs
2	Read Digital Inputs
3	Read Analog Inputs[1]
4	Read Analog Outputs[1]
5	Write Digital Output
6	Write Digital Outputs
7	Read Digital Outputs
15	Write Digital Outputs
16	Write Analog Outputs

[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. See MV command in the RIO command reference for further details.

Function Code 1 ($01) - Read Coils

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

The RIO will accept the following range requests:

Starting Address Range: $0000-$0017

Quantity of Coils Range: $0001-$0018

The RIO will respond to a request with function code $01 followed by a byte count (in hex), which describes the number of bytes of digital outputs being returned. The byte count can be calculated as follows:

$$byte\ count = \frac{quanitity\ of\ outputs}{8}$$

Equation 3.1:

If the remainder of this equation is not 0, the byte count is instead calculated by:

$$byte\ count = \frac{quanitity\ of\ outputs}{8} + 1$$

Equation 3.2:

After the byte count, the RIO will respond with a coil status using the amount of bytes calculated above with values ranging from $000000-$FFFFFF. 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	Addresses	Coil	Addresses	Coil	Addresses
0	Digital Output 0	8	Digital Output 8	16	Digital Output 16
1	Digital Output 1	9	Digital Output 9	17	Digital Output 17
2	Digital Output 2	10	Digital Output 10	18	Digital Output 18
3	Digital Output 3	11	Digital Output 11	19	Digital Output 19
4	Digital Output 4	12	Digital Output 12	20	Digital Output 20
5	Digital Output 5	13	Digital Output 13	21	Digital Output 21
6	Digital Output 6	14	Digital Output 14	22	Digital Output 22
7	Digital Output 7	15	Digital Output 15	23	Digital Output 23

Table 3.3: Digital output coil map

Function Code 2 ($02) - Read Discrete Inputs

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

The RIO will accepts the following range request:

Starting Address Range: $0000-$0017 (referencing digital inputs 0-15)

Quantity of Inputs Range: $0001-$0018

The RIO will respond to a request with function code $02 followed by the byte count (in hex), which describes the number of bytes of digital inputs being returned as calculated by Equation 3.1 or Equation 3.2 on pg 26.

After the byte count, the RIO will respond with a input status using the amount of counts calculated by Equation 3.1 or Equation 3.2 with values ranging from $000000-$FFFFFF. 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	Addresses	Coil	Addresses	Coil	Addresses
0	Digital Input 0	8	Digital Input 8	16	Digital Input 16
1	Digital Input 1	9	Digital Input 9	17	Digital Input 17
2	Digital Input 2	10	Digital Input 10	18	Digital Input 18
3	Digital Input 3	11	Digital Input 11	19	Digital Input 19
4	Digital Input 4	12	Digital Input 12	20	Digital Input 20
5	Digital Input 5	13	Digital Input 13	21	Digital Input 21
6	Digital Input 6	14	Digital Input 14	22	Digital Input 22
7	Digital Input 7	15	Digital Input 15	23	Digital Input 23

Table 3.4: Digital input coil map

Function Code 3 ($03) - Read Holding Registers

Modbus function code $03 is a request to read holding registers. The default configuration is to respond to this command with analog input register information.

Function code 3 has very different behaviors depending on the configuration commands set. A synopsis and brief list of commands that effect how the RIO responds to function code 3 is in Table 3.5 below:

Command	Description
MV	Swaps the response of function code 3 and 4
MI	Configures if the RIO will respond with a 16-bit integer or a 32-bit float
ME	Enables a master to write and read to arrays to array locations of an RIO slave

Table 3.5: Important configuration commands for Function Code 3

The RIO will accept different starting address ranges for a read holding registers request depending on the state of the MI command:

MI 0 is set (register data is 16-bit integer [counts])

Address range: $0000-$000E

Quantity of Registers: up to $0008

MI 1 is set (register data is 32-bit floating [volts])

Address range: $0000-$0007

Quantity of Registers: up to $0010

The RIO will respond to a request with function code $03 followed with a byte count (in hex) as calculated:

$$Byte\ count = 2 \times (Number\ of\ Analog\ Inputs \times n)$$

Equation 3.3:

Where, $Number\ of\ Analog\ Inputs$ is equal to the number of analog inputs the master is trying to query

$n = 1$ if MI 0 is set

$n = 2$ if MI 1 is set

The RIO will respond with the register value data the size of the byte count field calculated in Equation 3.3 above.

Function code 3 also has a secondary capability: The ability to read from array data on the RIO[1]. Up to 1000 elements are available. Each element is accessible as a 16-bit unsigned integer (Modbus register 1xxx) *or* as a 32 bit floating point number (Modbus registers 2xxx). This capability is enabled by setting the ME command, see the RIO Command Reference for further details.

[1] Only RIO firmware revisions Rev D and later support this capability ME.

Register Address	32-bit floating [volts]	16-bit integer [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		

Table 3.6: Analog input register address map

Function Code 4 ($04) - Read Input Registers

Modbus function code $04 is a request to read input registers. In its default configuration the RIO responds to this command with analog output register information.

Function code 4 has very different behaviors depending on the configuration commands set. A synopsis and brief list of commands that effect how the RIO responds to function code 4 is in Table 3.7 below:

Command	Description
MV	Swaps the response of function code 3 and 4
MI	Configures if the RIO will respond with a 16-bit integer or a 32-bit float

Table 3.7: Important configuration commands for Function Code 4

The RIO will accept different address ranges for a read input registers request depending on the state of the MI command:

MI 0 is set (register data is 16-bit integer [counts])

Address range: $0000-$000E

Quantity of Registers: up to $0008

MI 1 is set (register data is 32-bit floating [volts])

Address range: $0000-$0007

Quantity of Registers: up to $0010

The RIO will respond to a request with function code $04 followed with a byte count (in hex) as calculated by Equation 3.3 on page 27.

The RIO will respond with an input registers field data the size of the byte count field calculated in Equation 3.3. The data is in ascending order from the analog output referenced in the address.

Register Address	32-bit floating [volts]	16-bit integer [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		

Table 3.8: Analog output register address map

Function Code 5 ($05) – Write Single Coil

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.

The RIO will accepts the following range request:

Starting Address Range: $0000-$0017

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

Same as Coil Map as in Table 3.3, pg 26.

Function Code 6 ($06) – Preset Single Register

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

The RIO will accepts the following range request:

Starting Address: $0000

Register value range : $0000 - $FFFF

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

Same as Coil Map as in Table 3.3, pg 26.

Function Code 7 ($07) – Read Exception Status

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.

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	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

Table 3.9: Digital output coil for function code 7

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

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

The RIO will accepts the following range request:

Starting Address: $0000-$000F (referencing 16 digital outputs 0-15)

Register value range : $0001-$0010

The RIO will respond with function code $0F followed by quantity of outputs (in hex) which matches the quantity of outputs field of the request packet.

Same as Coil Map as in Table 3.3, pg 26.

Function Code 16 ($10) – Write Multiple Registers

Modbus function code ($10) is a request to write multiple registers. This will write multiple analog outputs to an RIO configured as a slave.

Command	Description
MI	Configures if the RIO will respond with a 16-bit integer or a 32-bit float
ME	Enables a master to write and read to arrays to array locations of an RIO slave

Table 3.10: Important configuration commands for Function Code 16

The RIO will accept different starting address ranges for a write multiple registers request depending on the state of the MI command.

MI 0 is set (register data is 16-bit integer [counts])

Address range: $0000-$000E

MI 1 is set (register data is 32-bit floating [volts])

Address range: $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.

Function code 3 also has a secondary capability: The ability to write array data to the RIO[1]. Up to 1000 elements are available. Each element is accessible as a 16-bit unsigned integer (Modbus register 1xxx) *or* as a 32 bit floating point number (Modbus registers 2xxx). This capability is enabled by setting the ME command, see the RIO Command Reference for further details.

[1] Only RIO firmware revisions Rev D and later support this capability ME.

Same Register Map as Table 3.7, pg 28.

Modbus Exceptions

As a Master

An RIO-47xxx configured as a master can query the function code of the last response it received using the _MW0 operand. In addition, the _MW0 operand can be used to determine if an exception has occurred.

The _MW1 operand can be used to determine why the exemption occurred.

See MW in the Command Reference for more details.

As a Slave

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.

Galil Modbus Packet Structure

The following section provides examples in order to explain Galil's implementation of Modbus. The following examples assume that a Galil unit is a Master and that another Galil unit is the slave connected over port <502> using some Ethernet handle m. In addition assume all Modbus information are stored in some array, array[].

Function Code 1 ($01)

Assume the status digital outputs on the Galil slave in descending order from 15-0 are as follows:

0111001100110111

And that the command issued from the Galil Master is MBm=,1,2,10,array[]

The Modbus request and response packets would form as follows:

Master Request		Slave 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

As a result, the Galil master's would have the following information stored in it's arrays:

array[0]=205
array[1]=12

Function Code 2 ($02)

Assume the status digital inputs on the Galil slave in descending order from 15-0 are as follows:

0111001100110111

And that the command issued from the Galil Master is MBm=,2,2,12,array[]

The Modbus request and response packets would form as follows:

Master Request		Slave 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

As a result, the Galil master's would have the following information stored in it's arrays:

array[0]=205
array[1]=12

Function Code 3 ($03)

Assume the status analog inputs on the Galil slave in descending order from 0-7 are as follows:

.4822, .9753, 1.4673, 1.9629, 2.4622, 2.9675, 3.4583, 3.9600

And that the command issued from the Galil Master is MBm=,3,2,4,array[]

The Modbus request and response packets would form as follows. Both responses for a slave configured for MI 0 and MI 1 are shown:

Master Request		**Slave Response**					
		32-bit Floating Point (MI 0)			16-bit Integer (MI 1)		
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
		Reg Val3 Low	$00		RegVal6 Low	$18	

With the slave MI 0 set, the Galil master's would have the following information stored in it's arrays:

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

With the slave MI 1 set, the Galil master's would have the following information stored in it's arrays:

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

Function Code 4 ($04)

Assume the status analog outputs on the Galil slave in descending order from 0-7 are as follows:

.5, 1, 1.5, 2, 2.5, 3, 3.5, 4

And that the command issued from the Galil Master is MBm=,4,2,4,array[]

The Modbus request and response packets would form as follows. Both responses for a slave configured for MI 0 and MI 1 are shown:

Master Request		Slave Response					
		32-bit Floating Point (MI 0)			16-bit Decimal (MI 1)		
Field Name	**Hex**	**Field Name**	**Hex**	**Volts**	**Field Name**	**Hex**	**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
		Reg Val3 Low	$00		RegVal6 Low	$99	

With the slave MI 0 set, the Galil master's would have the following information stored in it's arrays:

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

With the slave MI 1 set, the Galil master's would have the following information stored in it's arrays:

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

Function Code 5 ($05)

Assume that the command issued from the Galil Master is MBm=,5,7,1

The Modbus request and response packets would form as follows:

Master Request		Slave 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, the Galil slave will have output 7 turned on.

Function Code 6 ($06)

Assume that the command issued from the Galil Master is MBm= ,6,0,$55AA

The Modbus request and response packets would form as follows:

Master Request		Slave 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)

Assume the status digital outputs on the Galil slave in descending order from 15-0 are as follows:

0101010110101010

And that the command issued from the Galil Master is MBm= ,7,array[]

The Modbus request and response packets would form as follows:

Master Request		Slave Response	
Field Name	**Hex**	**Field Name**	**Hex**
Function	$07	Function	$07
		Output Data	$AA

As a result, array[0] on the master Galil master will equal 170.

Function Code 15 ($0F)

Assume the desired setting for digital outputs on the Galil slave in descending order from 15-0 are as follows:

0101010110101010

And that the command issued from the Galil Master is MBm= ,15,0,16,array[], where

array[0]=$AA
array[1]=$55

Note: $AA55 results in the binary value of 1's and 0's, representing the desired outputs as listed above.

The Modbus request and response packets would form as follows:

Master Request		Slave 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

Example 1

Assume the desired setting for digital outputs on the Galil slave in descending order from 15-0 are as follows:

0101010110101010

And that the command issued from the Galil Master is MBm= ,16,2,4,array[], where array[] contains [$40A0,$0000,$4040,$0000].

The Modbus request and response packets would form as follows if MI 0 is set:

Master Request		**Slave Response**	
32-Bit Floating Point			
Field Name	**Hex**	**Field Name**	**Hex**
Function	$10	Function	$10
Starting Address Hi	$00	Starting Address Hi	$00
Starting Address Lo	$02	Starting Address Lo	$02
Quantity Outputs Hi	$00	Quantity of Registers Hi	$00
Quantity Outputs Lo	$04	Quantity of Registers Lo	$04
Byte Count	$08		
RegVal0 High	$40		
	$A0		
	$00		
RegVal0 Low	$00		
RegVal1 High	$40		
	$40		
	$00		
RegVal1 Low	$00		

As a result, the Galil slave will have analog output 1 set to 5V and analog output 2 set to 3V.

Example 2

Assume the desired setting for digital outputs on the Galil slave in descending order from 15-0 are as follows:

0101010110101010

And that the command issued from the Galil Master is MBm= ,16,2,2,array[], where array[] contains [$FFFF,$9999,$6666,$3333].

The Modbus request and response packets would form as follows if MI 1 is set:

Master Request		**Slave Response**	
32-Bit Floating Point			
Field Name	**Hex**	**Field Name**	**Hex**
Function	$10	Function	$10
Starting Address Hi	$00	Starting Address Hi	$00
Starting Address Lo	$02	Starting Address Lo	$02
Quantity Outputs Hi	$00	Quantity of Registers Hi	$00
Quantity Outputs Lo	$02	Quantity of Registers Lo	$02
Byte Count	$04		
RegVal0 High	$FF		
RegVal0 Low	$FF		
RegVal1 High	$99		
RegVal1 Low	$99		

As a result, the Galil slave will have analog output 2 set to 5V and analog output 3 set to 3V.

Analog I/O Ranges

The analog inputs and outputs range from different values depending on the configuration of the RIO. This information is specifically important when using the RIO to communicate as a modbus slave and MI is set to 1. With your part number, see Table 1.1 or Table 4.4 to find what analog option you have.

0-5V Analog I/O Option

Analog Inputs

AQ x,m (see command reference for details)

m	Analog Range	Counts Range(decimal)	Counts Range(hex)
0	0-5V	0-32572	0x0000 - 0x7FF0
1	0-5V	0-32572	0x0000 - 0x7FF0

Analog Outputs

Analog Range	Counts Range(decimal)	Counts Range(hex)
0-5V	0-65520	0x0000 - 0xFFF0

±10V Configurable Analog I/O Options (12- or 16-bit versions)

Analog Inputs

AQ x,m (see command reference for details)

m	Analog Range	Counts Range(decimal)	Counts Range(hex)
1	±5V	-32768 to 32767	0x8000 - 0x7FFF
2	±10V	-32768 to 32767	0x8000 - 0x7FFF
3	0-5V	0-65535	0x0000 - 0xFFFF
4	0-10V	0-65535	0x0000 - 0xFFFF

Analog Outputs

DQ x,m (see command reference for details)

m	Analog Range	Counts Range(decimal)	Counts Range(hex)
1	0-5V	0-65535	0x0000 - 0xFFFF
2	0-10V	0-65535	0x0000 - 0xFFFF
3	±5V	0-65535	0x0000 - 0xFFFF
4	±10V	0-65535	0x0000 – 0xFFFF

Data Record

QR and DR Commands

The RIO can provide a block of status information back to the host computer in a single Ethernet packet using either the QR or DR commands. The QR command returns the Data Record as a single response. The DR command causes the controller to send a periodic update of the Data Record out a dedicated UDP Ethernet handle. The Data Record response packet contains binary data that is a snapshot of the controller's I/O status.

Since the Data Record response contains all information in binary format; the result of this command cannot be displayed in a Galil terminal.

The QR and DR commands will return 4 bytes of header information, followed by an entire data record. A data record map is provided below.

Data Record Map Key	
Acronym	**Meaning**
UB	Unsigned byte
UW	Unsigned word
SL	Signed long
UL	Unsigned long

RIO Data Record

RIO-471xx/472xx Data Record

DATA TYPE	ITEM
UB	1st byte of header
UB	2nd byte of header
UB	3rd byte of header
UB	4th byte of header
UW	Sample number
UB	Error Code
UB	General Status
UW	Analog Out Channel 0 (counts)
UW	Analog Out Channel 1 (counts)
UW	Analog Out Channel 2 (counts)
UW	Analog Out Channel 3 (counts)
UW	Analog Out Channel 4 (counts)
UW	Analog Out Channel 5 (counts)
UW	Analog Out Channel 6 (counts)
UW	Analog Out Channel 7 (counts)
UW[1]	Analog In Channel 0 (counts)
UW[1]	Analog In Channel 1 (counts)
UW[1]	Analog In Channel 2 (counts)
UW[1]	Analog In Channel 3 (counts)
UW[1]	Analog In Channel 4 (counts)
UW[1]	Analog In Channel 5 (counts)
UW[1]	Analog In Channel 6 (counts)
UW[1]	Analog In Channel 7 (counts)
UW	Output State
UW	Input State
UL	Pulse Count
SL	ZC data – user configurable variable

SL	ZD data – user configurable variable

RIO-47300 Data Record

DATA TYPE	ITEM
UB	1st byte of header
UB	2nd byte of header
UB	3rd byte of header
UB	4th byte of header
UW	Sample number
UB	Error Code
UB	General Status
UW	Analog Out Channel 0 (counts)
UW	Analog Out Channel 1 (counts)
UW	Analog Out Channel 2 (counts)
UW	Analog Out Channel 3 (counts)
UW	Analog Out Channel 4 (counts)
UW	Analog Out Channel 5 (counts)
UW	Analog Out Channel 6 (counts)
UW	Analog Out Channel 7 (counts)
UW[1]	Analog In Channel 0 (counts)
UW[1]	Analog In Channel 1 (counts)
UW[1]	Analog In Channel 2 (counts)
UW[1]	Analog In Channel 3 (counts)
UW[1]	Analog In Channel 4 (counts)
UW[1]	Analog In Channel 5 (counts)
UW[1]	Analog In Channel 6 (counts)
UW[1]	Analog In Channel 7 (counts)
UW	Digital outputs 0-15
UW	Digital outputs 16-23
UW	Digital inputs 0-15
UW	Digital inputs 16-23
UL	Pulse Count
SL	ZC data – user configurable variable
SL	ZD data – user configurable variable

-QUAD/-BISS/-SSI

The data record for the RIO-47122, RIO-47142, RIO-47202, or RIO-47300 ordered with an encoder option (-QUAD, -BISS, or -SSI) contains the following information appended to the standard data record for that PLC.

DATA TYPE	ITEM
SL	Encoder channel 0
SL	Encoder channel 1
SL	Encoder channel 2
SL	Encoder channel 3

-24EXIN/-24EXOUT

The data record for the RIO-47300 ordered with an extended I/O option (-24EXIN or -24EXOUT) contains the following information appended to the standard data record for that PLC.

DATA TYPE	ITEM
UW	Digital outputs 24-39
UW	Digital outputs 40-47
UW	Digital inputs 24-39
UW	Digital inputs 40-47

[1]These may be signed or unsigned words depending on the AQ setting on the RIO-4712x. For example, if the bytes received from the data record packet for analog input 0 were 00 80, it could have the following meaning, depending on AQ

Little Endian	AQ 0,1	AQ 0,2	AQ 0,3	AQ 0,4
80 00	-5 Volts	-10 Volts	2.5 Volts	5 Volts

This data can be broken up into sections. The **Data Record Map** includes the 4 bytes of header. The **General Data Block** consists of the sample number, the error code, and the general status. **The I/O Data Block** includes all the other items in the above table.

Explanation of Status Information

Header Information –

Bytes 0, 1 of Header:

The first two bytes of the data record provide the header information.

BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
1	N/A	N/A	N/A	N/A	N/A	N/A	N/A
BIT 7	**BIT 6**	**BIT 5**	**BIT 4**	**BIT 3**	**BIT 2**	**BIT 1**	**BIT 0**
N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Bytes 2, 3 of Header:

Bytes 2 and 3 make up a word, which represents the Number of bytes in the data record, including the header. Byte 2 is the low byte, and byte 3 is the high byte.

Note: The header information of the data records is formatted in little endian.

General Status Information (1 Byte)

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
Program Running	N/A	N/A	N/A	N/A	Waiting for input from IN command	Trace On	Echo On

ZC and ZD Commands

Another important feature of the data record is that it contains two variables that can be set by the user. The ZC and ZD commands are responsible for these variables. Each variable can be a number, a mathematical equation, or a string. See the Command Reference for more information on the ZC and ZD commands.

Chapter 4 I/O

Introduction

Each RIO comes with a different set of default outputs types and quantity. Use Table 1.1 and Table 4.1 below to find out what default outputs come with your specific model. The interrogation command, **TZ**, allows the user to get a quick view of the I/O configuration and bit status.

Specifications

Access to I/O points is made through either the High Density D-Sub connectors on the top of the unit or through screw-terminal points, depending on your model. Pin-outs for the Connectors for RIO-47xxx are listed in the Appendix.

Digital Outputs

Make sure to check the configuration of your RIO before wiring the digital outputs (labeled DO). Table 4.1 shows the default output ratings for Bank 0 (DO[7:0]), Bank 1 (DO[15:8]), and Bank 2 (DO[23:16]) for each model. Table 4.1 also lists whether or not the product has the OUTC jumpers available. The OUTC jumpers are used to bypass optoisolation by using the RIO's internal +5V, see OUTC jumpers for details.

Model	Bank 0, DO[7:0]	Bank 1, DO[15:8]	Bank 2, DO[23:16]	OUTC Jumpers
RIO-47100	500mA Sourcing	25mA Sinking	–	Yes
RIO-47102	500mA Sourcing	25mA Sinking	–	Yes
RIO-47120	500mA Sourcing	25mA Sinking	–	Yes
RIO-47122	500mA Sourcing	25mA Sinking	–	Yes
RIO-47142	500mA Sourcing	500mA Sourcing	–	No
RIO-47200	500mA Sourcing	500mA Sourcing	–	Yes
RIO-47202	500mA Sourcing	500mA Sourcing	–	Yes
RIO-47300	500mA Sourcing	500mA Sourcing	500mA Sourcing	No

Table 4.1: Default RIO Output Configurations

For wiring and electrical information, see the individual sections below which individually describes each type of output: 500mA Sourcing, 25mA Sinking, and 25mA Sourcing. Each of these are wired differently and have separate constraints, so read each section carefully before wiring.

Note: For the following sections, "n" will denote the bank of interest representing either 0, 1, or 2 representing Bank 0, Bank 1, or Bank 2 respectively.

500mA Sourcing Outputs (HSRC)

The 500mA sourcing option, referred to as high power sourcing (HSRC), is capable of sourcing up to 500mA per output and up to 3A per **bank**. The voltage range for the outputs is 12-24 VDC. These outputs are capable of driving inductive loads such as solenoids or relays. The outputs are configured for hi-side (sourcing) only.

Electrical Specifications

Output PWR (OPnA) Max Voltage	24 VDC
Output PWR (OPnA) Min Voltage	12 VDC
Max Drive Current per Output	0.5 A (not to exceed 3A per Bank)

Wiring Information

With this configuration, the output power supply will be connected to Output PWR (labeled OPnA) and the power supply return will be connected to Output GND (labeled OPnB) , where n denotes 0, 1, or 2 referring to Bank 0, Bank 1, Bank 2 respectively. Note that the load is wired between DO and Output GND. The wiring diagram for Bank 0 is shown in Figure 4.1, Bank 1 in Figure 4.2, and Bank 2 in Figure 4.3. Refer to Connectors for RIO-47xxx in the Appendix for pin-out information.

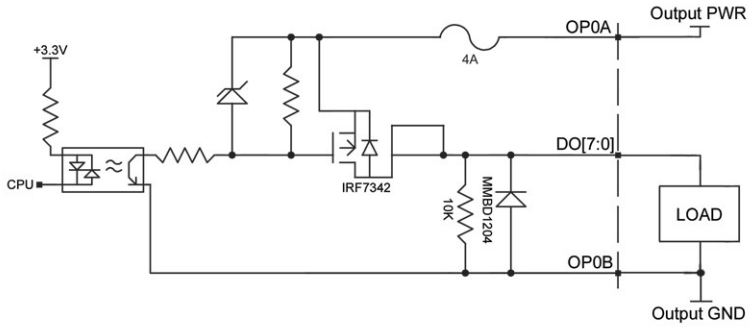


Figure 4.1: 500mA Sourcing wiring diagram for Bank 0, DO[7:0]

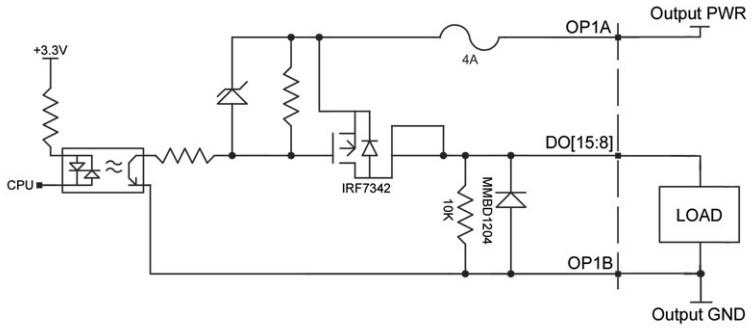


Figure 4.2: 500mA Sourcing wiring diagram for Bank 1, DO[15:8]

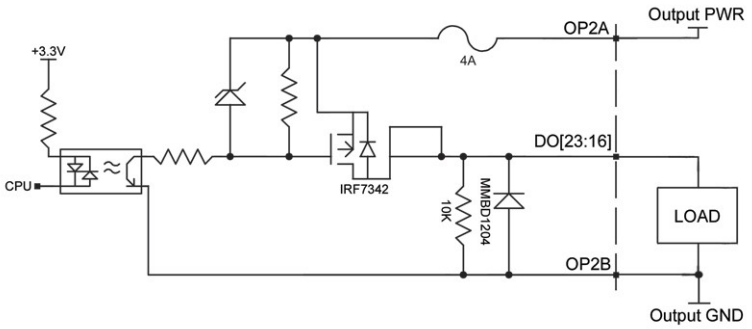


Figure 4.3: 500mA Sourcing wiring diagram for Bank 2, DO[23:16]

25mA Low Power Sinking Outputs (LSNK)

The 25mA sinking option, referred to as lower power sinking (LSNK), are capable of sinking up to 25mA per output. The voltage range for the outputs is 5-24 VDC. These outputs should not be used to drive inductive loads directly.

Electrical Specifications

Output PWR (OPnB) Max Voltage	24 VDC
Output PWR (OPnB) Min Voltage	5 VDC
ON Voltage (No Load, Output PWR= 5 VDC)	1.2 VDC
Max Drive Current per Output	25 mA, sinking

Wiring Information

The output power supply will be connected to Output PWR (labeled OPnB) and the power supply return will be connected to Output GND (labeled OPnA), where n denotes 0, 1, or 2 referring to Bank 0, Bank 1, and Bank 2 respectively. Note that the load is wired between Output PWR and DO. The wiring diagram for Bank 0 is shown in Figure 4.4 , Bank 1 in Figure 4.6, and Bank 2 in Figure 4.5. Refer to Connectors for RIO-47xxx in the Appendix for pin-out information.

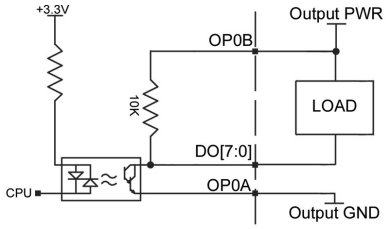


Figure 4.4: 25mA Sinking wiring diagram for Bank 0, DO[7:0]

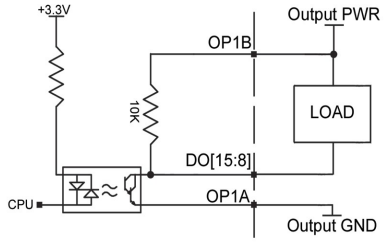


Figure 4.5: 25mA Sinking wiring diagram for Bank 1, DO[15:8]

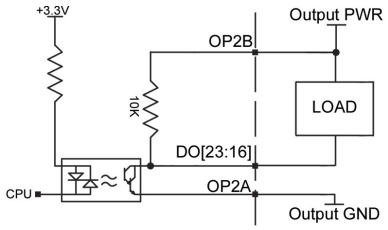


Figure 4.6: 25mA Sinking wiring diagram for Bank 2, DO[23:16]

25mA Low Power Sourcing Outputs (LSRC)

The 25mA sourcing option, referred to as lower power sourcing (LSRC), are capable of sourcing up to 25mA per output. The voltage range for the outputs is 5-24 VDC. These outputs should not be used to drive inductive loads directly.

Electrical Specifications

Output PWR(OPnA) Max Voltage	24 VDC
Output PWR (OPnA) Min Voltage	5 VDC
Max Drive Current per Output	25 mA, sourcing

Wiring Information

With this configuration, the output power supply will be connected to Output PWR (labeled OPnA) and the power supply return will be connected to Output GND (labeled OPnB), where n denotes 0, 1, or 2 referring to Bank 0, Bank 1, and Bank 2 respectively. Note that the load is wired between DO and Output GND. The wiring diagram for Bank 0 is shown in Figure 4.7, Bank 1 in Figure 4.8, and Bank 2 in Figure 4.9. Refer to Connectors for RIO-47xxx in the Appendix for pin-out information.

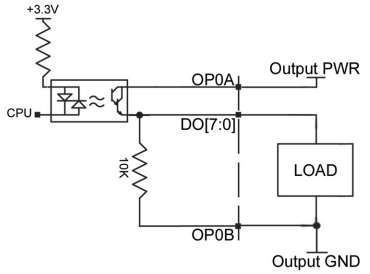


Figure 4.7: 25mA Sourcing wiring diagram for Bank 0, DO[7:0]

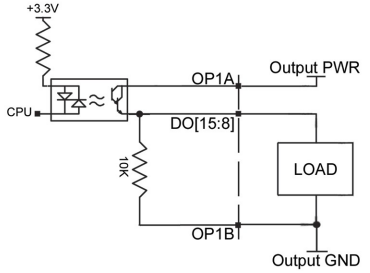


Figure 4.8: 25mA Sourcing wiring diagram for Bank 1, DO[15:8]

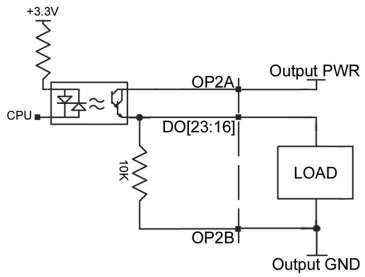


Figure 4.9: 25mA Sourcing wiring diagram for Bank 2, DO[23:16]

OUTC jumpers

The OUTC jumpers can be used when an external power supply is not desired for digital outputs 8-15. These low power outputs can use the internal +5V from the RIO instead of an external supply. To do this, place a jumper on the pins labeled OUTC as shown in Figure 4.10.

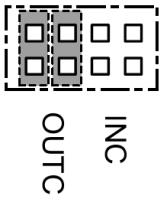


Figure 4.10: OUTC jumpers for RIO

1. These jumpers DO NOT supply power to high power digital outputs, an external supply is required for those outputs.
2. With the RIO-472xx, the OUTC jumpers are only available when LSRC or LSNK options are ordered from the factory.
3. OUTC jumpers are not available on the RIO-47142 or RIO-47300 (all high power outputs).

PWM Outputs

With firmware revisions Rev D and newer, Digital Outputs 14 and 15 can be setup independently as PWM outputs using the DY, FQ and PM commands. The standard optoisolated outputs found on the RIO-47xxx will have a limited bandwidth (50Hz) that will not allow the full range of frequency and duty cycle available from the DY, FQ and PM commands. The RIO can be ordered with a –PWM option that will bypass the optoisolated outputs and provide buffered outputs for DO[15:14]. See the -PWM section in the Appendix for more information.

Digital Inputs

The RIO-47xxx has 16-24 optoisolated inputs (depending on model). These inputs can be read individually using the function `@IN[x]` where x specifies the input number (0 thru 23). These inputs are uncommitted and can allow the user to create conditional statements related to events external to the PLC.

This can be accomplished by connecting a voltage in the range of +5V to +24V into INCOM of the input circuitry from a separate power supply.

	Input Common
RIO-471xx	
Bank 0, DI[7:0]	INC0
Bank 1, DI[15:8]	INC1
RIO-472xx	
Bank 0, DI[7:0]	INC0A
Bank 1, DI[15:8]	INC1A
RIO-47300	
Bank 0, DI[7:0]	INC0A
Bank 1, DI[15:8]	INC1A
Bank 2, DI[23:16]	INC2A

Table 4.2: List of Input Commons for each Bank given the RIO model.

Although rare, it is sometimes desired that optoisolation is bypassed. This can be done by using the "INC" jumpers on the RIO allowing the inputs to be powered by the RIO's +5V internal reference voltage. In addition, this requires a ground reference voltage as supplied by the Input Reference Ground pins. For more details, see the INC jumpers section below.

Electrical Specifications

INCOM Max Voltage	24 V_{DC}
INCOM Min Voltage	0 V_{DC}
Minimum current to turn on Inputs	1.2 mA
Minimum current to turn off Inputs once activated (hysteresis)	0.5 mA
Maximum current per input[1]	11 mA
Internal resistance of inputs	2.2 kΩ

[1]See the Input Current Limitations section below for details.

Wiring the Digital Inputs

To take full advantage of optoisolation, an isolated power supply should be used to provide the voltage at the input common connection. Connecting the ground of the isolated power to the ground of the controller will bypass optoisolation and is not recommended if true optoisolation is desired.

Banks of inputs can be used as either active high or low. Connecting $+V_s$ to INCOM will configure the inputs for active low as current will flow through the diode when the inputs are pulled to the isolated ground. Connecting the isolated ground to INCOM will configure the inputs for active high as current will flow through the diode when the inputs are pulled up to $+V_s$.

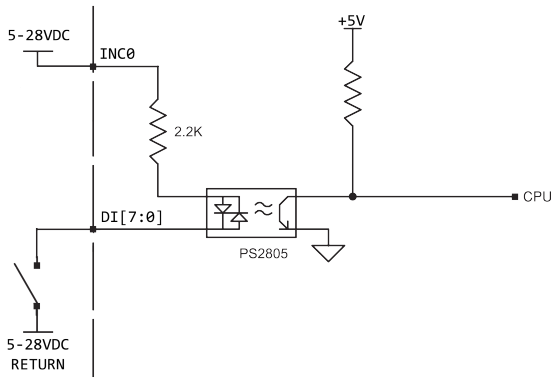


Figure 4.11: Digital Input wiring for Bank 0, DI[7:0]

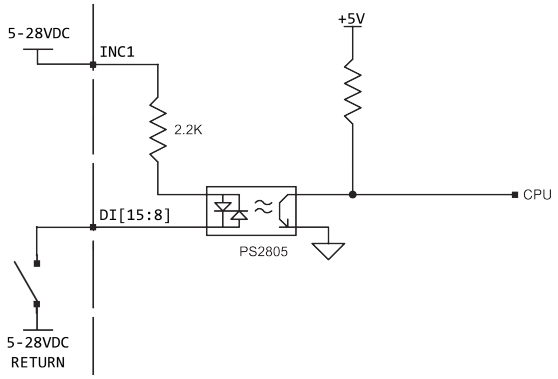


Figure 4.12: Digital Input wiring for Bank 1, DI[15:8]

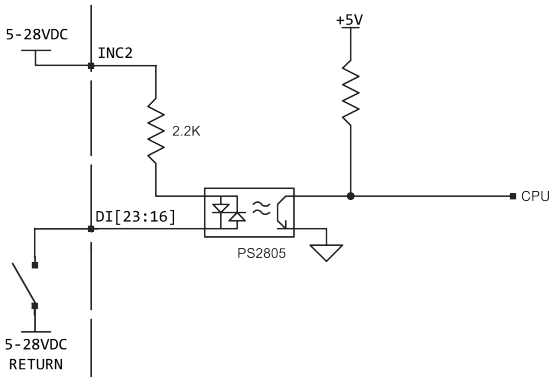


Figure 4.13: Digital Input wiring for Bank 2, or DI[23-16]

Input Current Limitations

The current for an optoisolated input shall not exceed 11mA. Some applications may require the use of an external resistor (R) to limit the amount of current for an input. These external resistors can be placed in series between the inputs and their power supply (Vs). To determine if an additional resistor (R) is required, follow Equation 3.4 below for guidance.

$$1\,mA < \frac{Vs}{R + 2200\,\Omega} < 11\,mA$$

Equation 3.4: Current limitation requirements for each input.

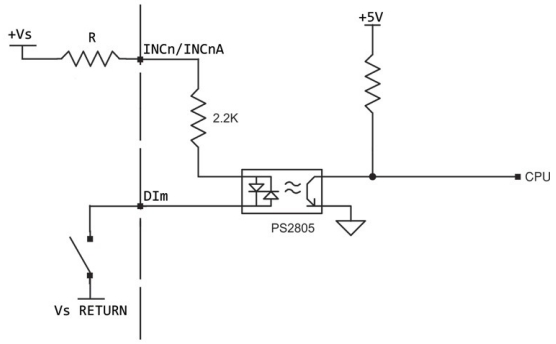


Figure 3.14: Wiring diagram showing how to put R in series between Vs and INCn/INCnA to limit current through the bank.

Where n= 0, 1, and 2 representing input banks INC0/INC0A, INC1/INC1A, or INC2A

m= [7:0], [15:8], and [23:16] depending on the bank of INCn/INCnA

INC jumpers

The INC jumpers can be used when an external power supply is not desired to power the digital inputs. When INC jumpers are installed, example shown in Figure 4.15, the Input Common pins are internally connected to the RIO +5V reference signal. Each RIO model has a slightly different labeling scheme for these jumpers, so use Table 4.3 as a reference for the INC Jumper Labels for your model. In addition to installing the INC jumpers, the digital inputs must have a reference ground. This reference comes from the Input Reference Ground pins as shown in Table 4.3.

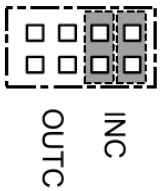


Figure 4.15: INC Jumpers installed on a RIO-47100

	INC Jumper Labels [2]	**No Connect** [3] (+5V internal reference)	**Input Reference Ground**
RIO-471xx			
Bank 0, DI[7:0]	INC	INC0	INC0B[1]
Bank 1, DI[15:8]	INC	INC1	INC1B[1]
RIO-472xx			
Bank 0, DI[7:0]	INC	INC0A	INC0B
Bank 1, DI[15:8]	INC	INC1A	INC1B
RIO-47300			
Bank 0, DI[7:0]	INC0A, INC0B	INC0A	INC0B
Bank 1, DI[15:8]	INC1A, INC1B	INC1A	INC1B
Bank 2, DI[23:16]	INC2A, INC2B	INC2A	INC2B

Table 4.3: Listing of INC Jumpers and Input Reference Ground by model

[1] Labeled "N/C" see RIO-471xx - 44 pin D-Sub Connector for correct pin-outs.

[2] Location of the jumpers are in the Appendix listed under Jumper Descriptions.

[3] Usually this can remain a No Connect. Alternatively, these pins can be used as a +5V internal reference that can only supply 10mA total. Note: if any noise is introduced on this pin, it can have a deleterious effect on the analog signals.

WARNING	Do not connect any power to the Input Common pins when INC jumpers are installed, damage will occur to the unit.

Figure 4.16, Figure 4.17, and Figure 4.18 shows (for the RIO-47100, -47200, and -47300 respectively) how the INC jumpers effect the internal wiring as well as how to externally wire inputs when these jumpers are in use.

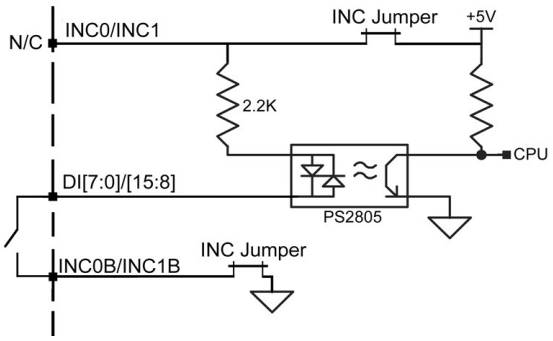


Figure 4.16: Wiring diagram with INC jumpers installed on the RIO-47100

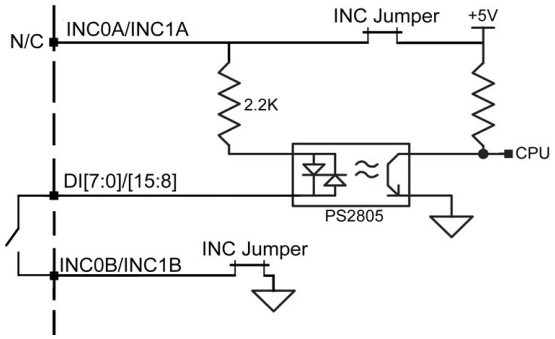


Figure 4.17: Wiring diagram with INC jumpers installed on the RIO-47200

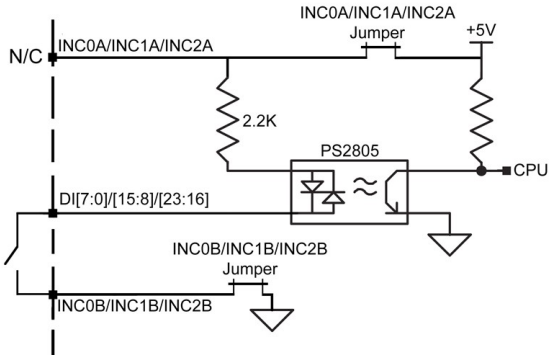


Figure 4.18: Wiring diagram with INCnA/INCnB jumpers installed on the RIO-47300

Pulse Counter Input

Digital input 3 (DI3) is a special purpose input that (when enabled) is used to count pulses coming in. To enable the pulse counter, the PC command must be issued with the following syntax:

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)

When the PC command is enabled, input DI3 will count high or low going edges. The operand _PC is used to report back the number of pulses counted. The maximum frequency of the input is limited by the optocouplers to 300 Hz (50% duty cycle). If a higher frequency is needed order the -HS option in the Appendix.

-HS Option Required with *Expanded Memory* RIO's

The *Expanded Memory* models of the RIO cannot use input 3 as the Pulse Counter Input unless the -HS option is ordered. To see if your model requires the -HS option to use this feature, see table Table 1.1 to find out if you have an *Expanded Memory* RIO.

Analog Outputs

The RIO product line has two main types of analog outputs available for the different models. There are 0-5V and ±10V configurable analog output options. The ±10V configurable option can be ordered with 16-bit resolution and are 12-bit by default. Table 4.4 shows the models and available analog output configurations.

By default the RIO-472xx does not include analog outputs. It can be ordered with analog outputs, see AO Option (SCB-48608) in the Appendices for ordering information.

Model	0-5 V	±10V Configurable	±10V Configurable - 16 bit
RIO-4710x	Yes	No	No
RIO-4712x	No	Yes	Yes
RIO-47142	No	Yes	Yes
RIO-472xx	SCB-48608 (8AO_5v12bit)	SCB-48608 (8AO_10v12bit)	SCB-48608 (8AO_10v16bit)
RIO-47300	No	Yes	Yes

Table 4.4: RIO Analog Output Configurations

If you are unsure of what analog output options you have—compare Table 4.4 with Table 1.1 which also lists the default analog options for each standard RIO part number.

0-5V Analog Outputs

Analog outputs 0-7 found on the basic RIO products have a 0-5V range and 12bit resolution.

Electrical Specifications

Maximum Output Voltage	5V
Minimum Output Voltage	0V
Resolution	12 bit (0-5V range)
Maximum Current Output	4mA (sink/source)

+/-10V Configurable Analog Outputs

Analog outputs 0-7 found on the RIO-4712x and other models have a configurable voltage range that is set using the DQ command. The default outputs have a 12bit DAC resolution (order -16Bit for 16 bit resolution). **See the DQ command in the Command Reference for a full explanation.**

DQ	Analog Output Range
DQ 0,1	Sets AO0 to 0-5VDC
DQ 1,2	Sets AO1 to 0-10VDC
DQ 2,3	Sets AO2 to +/-5VDC
DQ 3,4	Sets AO3 to +/-10VDC

Electrical Specifications

Maximum Output Voltage	10V
Minimum Output Voltage	-10V
Resolution	12-bit default, 16-bit optional
Maximum Current Output	4mA (sink/source)

Analog Inputs

The RIO product line has two main types of analog inputs available for the different models. There is a 0-5V analog input, and a ±10V configurable analog input. The ±10V configurable inputs can be ordered with 16 bit resolution. Table 4.5 shows the models and available analog input configurations.

By default the RIO-472xx has 0-5V analog inputs. It can be ordered with the ±10V Configurable Analog Inputs, see -(AI_10v12Bit) and -(AI_10v16Bit) in the Appendices for ordering information.

Model	0-5 V	±10V Configurable	±10V Configurable - 16 bit
RIO-4710x	Yes	No	No
RIO-4712x	No	Yes	Yes
RIO-47142	No	Yes	Yes
RIO-472xx	Yes - Default	With (AI_10v12bit) Option	With (AI_10v16bit) Option
RIO-47300	No	Yes	Yes

Table 4.5: RIO Analog Output Configurations

0-5V Analog Inputs

0-5V analog inputs have 12-bit ADC (a resolution of approximately 1.22mV) with a 100k input impedance.

Electrical Specifications

Input Impedance 100kΩ

Differential Mode

The 0-5V analog inputs can be set for a differential mode. See the AQ command in the command reference for more information. Note: The AQ command is also used for the ±10V Configurable Analog Inputs, but as a different range for the parameters. Table 4.6 covers the AQ settings for the 0-5V Analog Inputs.

AQ	Differential Pairs
AQ 0,1	Input 0 & Input 1
AQ 2,1	Input 2 & Input 3
AQ 4,1	Input 4 & Input 5
AQ 6,1	Input 6 & Input 7

Table 4.6: Differential Analog Input Channels on RIOs with the 0-5V analog input option

±10V Configurable Analog Inputs

±10V configurable voltage range is set using the AQ command. The default inputs have a 12-bit DAC resolution (order -16Bit for 16-bit resolution). See the AQ command in the Command Reference for a full explanation.

Electrical Specifications

Input Impedance (12 and 16 bit) –

Unipolar (0-5V, 0-10V) 42 kΩ

Bipolar (±5V, ±10V) 31 kΩ

Setting Range with AQ

Use the AQ command to specify the analog input range.

AQ	Input Range
AQ 0,1	Set input 0 to have ±5V input range
AQ 1,2	Set input 1 to have ±10V input range
AQ 2,3	Set input 2 to have 0-5V input range
AQ 3,4	Set input 3 to have 0-10V input range

Table 4.7: Setting Input Ranges on the RIOs with the ±10V configurable option

Setting Differential Mode

The AQ command also allows the RIO to change the configuration from the default 8 single ended analog inputs to 4 differential analog inputs.

AQ	Differential Pairs
AQ 0,-1	Input 0 & Input 1 and ±5V input range
AQ 2,-2	Input 2 & Input 3 and ±10V input range
AQ 4,-3	Input 4 & Input 5 and 0-5V input range
AQ 6,-4	Input 6 & Input 7 and 0-10V input range

Table 4.8: Differential analog input channels on RIO's with the ±10V configurable option

Chapter 5 Programming

Overview

The RIO provides a versatile programming language that allows users to customize the RIO board for their particular application. Programs can be downloaded into the RIO memory, freeing up the host computer for other tasks. However, the host computer can send commands to the RIO at any time, even while a program is being executed.

In addition to commands that handle I/O, the RIO provides commands that allow it to make decisions. These commands include conditional jumps, event triggers, and subroutines. For example, the command JP#LOOP, n<10 causes a jump to the label #LOOP if the variable n is less than 10.

For greater programming flexibility, the RIO provides user-defined variables, arrays, and arithmetic functions. The following sections in this chapter discuss all aspects of creating applications programs. The RIO-47xx0 program memory size is 200 lines x 40 characters. The RIO 47xx2 increases the memory size to a total of 400 lines x 40 characters.

Editing Programs

Use Galil software to enter programs in the Editor window. After downloading a program, use the XQ command to execute the program. The RIO also has an internal editor that may be used to create and edit programs in the RIOs memory. The internal editor is a rudimentary editor and is only recommended when operating with Galil's DOS utilities or through a simple RS-232 communication interface such as Windows Hyperterminal. See the ED command in the Command Reference for more info.

Program Format

A RIO program consists of instructions combined to solve a programmable logic application. Action instructions, such as setting and clearing I/O bits, are combined with Program Flow instructions to form the complete program. Program Flow instructions evaluate real-time conditions, such as elapsed time or input interrupts, and alter program flow accordingly.

A delimiter must separate each RIO instruction. Valid delimiters are the semicolon (;) or carriage return. The semicolon is used to separate multiple instructions on a single program line where the maximum number of characters on a line is 40 (including semicolons and spaces). A line continuation character (`) (below the ~ on a standard keyboard) allows a command to be continued on the next

line in the case that 40characters is not enough for a single command (see example at the end of this section).

Using Labels in Programs

All RIO programs must begin with a label and end with an End (EN) statement. Labels start with the number (#) sign followed by a maximum of seven characters. The first character must be a letter; after that, numbers are permitted. Spaces are not allowed.

The maximum number of labels that can be defined on the RIO depends on the option ordered, see Table 1.2 for details.

Valid labels:

#BASICIO
#SQUARE
#X1
#input1

Invalid labels:

#1Square
#123
#PROGRAMMING (*longer than 7 characters)*

Special Labels

The RIO also has some special labels, which are used to define input interrupt subroutines and command error subroutines. The following is a list of the automatic subroutines supported by the RIO. Sample programs for these subroutines can be found in the section *Automatic Subroutines for Monitoring Conditions.*

#AUTO	Automatic Program Execution on power up
#ININTn	Label for Input Interrupt subroutine
#CMDERR	Label for incorrect command subroutine
#TCPERR	Ethernet communication error

#AUTO is a special label for automatic program execution. A program which has been saved into the controller non-volatile memory using the BP (Burn Program) command can be automatically executed upon power up or reset by beginning the program with the label #AUTO.

Commenting Programs

Using an Apostrophe to Comment

The RIO provides an apostrophe (‘) for commenting programs. This character allows the user to include up to 39 characters on a single line after the apostrophe and can be used to include comments from the programmer as in the following example:

```
#OUTPUT
' PROGRAM LABEL
SB1; CB2
'Set Bit 1 and Clear Bit 2
EN;  'END OF PROGRAM
```

Note: The NO command also works to comment programs. The inclusion of the apostrophe or NO commands will require process time by the RIO board.

Using REM Statements with the Galil Terminal Software

When using Galil software to communicate with the RIO, REM, as in remark, statements may also be included. 'REM' statements begin with the word 'REM' and may be followed by any comments that are on the same line. The Galil terminal software will remove these statements when the program is downloaded to the RIO board. For example:

```
#OUTPUT
REM PROGRAM LABEL
SB1;CB2;
REM Set Bit 1 and Clear bit 2
EN
REM END OF PROGRAM
```

Since the REM statements will be removed when the program is downloaded to RIO, be sure to keep a copy of the program with comments stored on the PC.

Program Lines Greater than 40 Characters

Line Continuation Character

A new character (`) {ascii character 96} has been included to allow a command in an application program to extend beyond the confines of the 40 character maximum line length.

```
#TEST
IF((var100=100)& (var101=50));MG"Condi`
tion satisfied";ELSE;MG"Stop";ENDIF
EN
```

This allows for

a) more efficient command compressing

b) the continuation of message commands (MG) on multiple lines.

c) Longer IF, JP, & JS conditional statements

(Note: the total length of a multi-line command can not exceed 80 characters)

Lock Program Access using Password

The RIO can lock out user access to the internal program using the PW and {cntrl}L{cntrl}K commands. The PW sets the Password for the unit and the {cntrl}L{cntrl}K will lock the application program from being viewed or edited . The commands ED, UL, LS and TR will give privilege error #106 when the RIO is in a locked state. The program will still run when locked. The locked or unlocked state can be burned with the BN command. Once the program is unlocked, it remains accessible until a lock command or a reset (with the locked condition burned in) occurs. An example of how to lock the program is shown here:

```
:PW test, test
:^L^K test,1          1 locks, 0 unlocks
:LS
?
TC1
106 Privilege violation
```

Executing Programs - Multitasking

The RIO can run up to 4 independent programs or threads simultaneously. They are numbered 0 thru 3, where 0 is the main thread.

The main thread differs from the others in the following ways:

1. Only the main thread, thread 0, may use the input command, IN.

2. When interrupts are implemented for command errors, the subroutines are executed in thread 0. However for the #ININTn subroutines, the RIO has the ability to execute multiple input interrupts (#ININTn) on designated threads, not limited to the main thread. For more information, refer to the II command in the Command Reference.

To begin execution of the various programs, use the following instruction:

XQ #A,n

Where A represents the label and n indicates the thread number. To halt the execution of any thread, use the instruction

HX n

where n is the thread number.

Note that both the XQ and HX commands can be performed from within an executing program.

For example:

Instruction	**Interpretation**
`#TASK1`	Task1 label
`AT0`	Initialize reference time
`CB1`	Clear Output 1
`#LOOP1`	Loop1 label
`AT 10`	Wait 10 msec from reference time
`SB1`	Set Output 1
`AT -40`	Wait 40 msec from reference time, then initialize reference
`CB1`	Clear Output 1
`JP #LOOP1`	Repeat Loop1
`#TASK2`	Task2 label
`XQ #TASK1,1`	Execute Task1
`#LOOP2`	Loop2 label
`WT20000`	Wait for 20 seconds
`HX1`	Stop thread 1
`MG"DONE"`	Print Message
`EN`	End of Program

The program above is executed with the instruction XQ #TASK2,0 which designates TASK2 as the main thread (i.e. Thread 0). #TASK1 is executed within TASK2.

Debugging Programs

The RIO provides commands and operands that are useful in debugging application programs. These commands include interrogation commands to monitor program execution, determine the state of

the RIO board and the contents of the program, array, and variable space. Operands also contain important status information, which can help to debug a program.

Trace Commands

The trace command causes the RIO to send each line in a program to the host computer immediately prior to execution. Tracing is enabled with the command, TR1. TR0 turns the trace function off. Note: When the trace function is enabled, the line numbers as well as the command line will be displayed as each command line is executed. The program lines come back as unsolicited messages.

Error Code Command

When a program error occurs, the RIO halts the program execution at the point of the error. To display the last line number of program execution, issue the command, MG _ED.

The user can obtain information about the type of error condition that occurred by using the command TC1. This command returns a number and text message, which describe the error condition. The command TC0 (or TC) will return the error code without the text message. For more information about the command TC, see the Command Reference.

RAM Memory Interrogation Commands

For debugging the status of the program memory, array memory, or variable memory, the RIO has several useful commands. The command DM ? will return the number of array elements currently available. The command DA? will return the number of arrays that can be currently defined. For example, if an RIO has a maximum of 400 and up to 6 arrays, and a single array of 100 elements is defined, the command DM ? will return the value 250, and the command DA ? will return 5.

To list the contents of the variable space, use the interrogation command LV (List Variables). To list the contents of array space, use the interrogation command LA (List Arrays). To list the contents of the program space, use the interrogation command LS (List Program). To list the application program labels only, use the interrogation command LL (List Labels).

Operands

In general, all operands provide information that may be useful in debugging an application program. Below is a list of operands that are particularly valuable for program debugging. To display the value of an operand, the message command may be used. For example, since the operand, _ED, contains the last line of program execution, the command MG _ED will display this line number.

_ED contains the last line of program execution (useful to determine where program stopped)
_DL contains the number of available labels
_UL contains the number of available variables
_DA contains the number of available arrays
_DM contains the number of available array elements

Debugging Example:

The following program has an error. It attempts to set bit 14 high, but “SD” is used as the command instead of “SB”. When the program is executed, the RIO stops at line 001. The user can then query the RIO board using the command, TC1. The RIO responds with the corresponding explanation:

Instruction	Interpretation
:LS	List Program
000 #A	Program Label
001 SD14	Set bit 14 high
002 SB15	Set bit 15 high
003 MG"DONE"	Print message
004 EN	End
:XQ #A	Execute #A
?001 SD14	Error on Line 1
:TC1	Tell Error Code
130 Unrecognized Command	This command doesn't
:MG_ED	Print line number where problem occurred
1.00	The error occurred on line 1 of the program

Program Flow Commands

The RIO provides instructions to control program flow. The RIO program sequencer normally executes program instructions sequentially. The program flow can be altered with the use of interrupts and conditional jump statements.

Interrupts

To function independently from the host computer, the RIO can be programmed to make decisions based on the occurrence of an input interrupt, causing the RIO board to wait for multiple inputs to change their logic levels before jumping into a corresponding subroutine. Normally, in the case of a Galil controller, when an interrupt occurs, the main thread will be halted. However, in the RIO, the user can indicate in which thread (the thread must be already running when the interrupt occurs) the interrupt subroutine is to be run. When the interrupt occurs, the specified thread's main program will be paused to allow the interrupt subroutine to be executed. Therefore, the user has the choice of interrupting a particular thread execution upon an input interrupt (see II command). The input interrupt routines are specified using #ININTn where n can be 0-3. In this way, the RIO can make decisions based on its own I/O status without intervention from a host computer. The Return from Interrupt (RI) command is used to return from this subroutine to the place in the program where the interrupt had occurred. If it is desired to return to somewhere else in the program after the execution of the #ININTn subroutine, the Zero Stack (ZS) command is used, followed by unconditional jump statements.
Note: When using multiple II commands in a program, each II command must point to a unique label and must activate on an unused thread. Two or more II commands cannot be set to execute on the same thread, nor can multiple II commands be pointed to the same #ININTn label. Please see the II command in the RIO-47xxx command reference for more details.

Examples:

Interrupt

Instruction	Interpretation
#A	Program Label
XQ#B,1	Execute #B in thread 1
II1,0,-1&3	#ININT1 in thread 0 when input 1 low and input 3 high
II2,1,-5&10	#ININT2 in thread 1 when input 5 low and input 10 high
AI 13&14	Trippoint on inputs 13 and 14
#LOOP;JP#LOOP	Pseudo program – Loop indefinitely
EN	End program
#B	Program Label
AI 7&-8	Trippoint on inputs 7 and 8
#LOOP2	
SB10	Set bit 10 high
WT500	Wait for half a second
CB10	Set bit 10 low
WT500	Wait for 500msec
JP#LOOP2	Create a 'light-blinker' effect
EN	End program
#ININT1	Input interrupt program label
MG"Loop stops"	Print message, saying loop program in main thread halted
RI0	Return to main program without restoring trippoint, but keeping the interrupt enabled
#ININT2	
MG"Blinker stops"	Print message, saying blinker effect in thread 1 halted, since #ININT2 runs in thread 1
WT10000	Wait 10 seconds for user to reset inputs 5 and 10
RI1,1	Return to thread 1's main program (blinker continues) while restoring trippoint on inputs 5 and 10; interrupt disabled

Note: This multitasking program can be executed with the instruction XQ #A,0 designating A as the main thread (i.e. Thread 0). #B is executed within A.

Event Trigger

This example waits for input 1 to go low and input 3 to go high, and then execute the TZ interrogation command. **Note:** The AI command actually halts execution of the program until the input occurs. If you do not want to halt the program sequences, use the Input Interrupt function (II) or a conditional jump on an input, such as:

JP #GO,(@IN[1] = 0) | (@IN[3] = 1).

Instruction	Interpretation
#INPUT	Program Label
AI-1&3	Wait for input 1 low and input 3 high
TZ	List the entire I/O status
EN	End program

Conditional Jumps

The RIO provides Conditional Jump (JP) and Conditional Jump to Subroutine (JS) instructions for branching to a new program location based on a specified condition. The conditional jump determines if a condition is satisfied and then branches to a new location or subroutine. Unlike event triggers such as the AI command, the conditional jump instruction does not halt the program sequence. Conditional jumps are useful for testing events in real-time. They allow the RIO to make decisions without a host computer.

Command Format - JP and JS

Format	Description
JS destination, logical condition	Jump to subroutine if logical condition is satisfied
JP destination, logical condition	Jump to location if logical condition is satisfied

The destination is a program line number or label where the program sequencer will jump if the specified condition is satisfied. Note that the line number of the first line of program memory is 0. The comma designates "IF". The logical condition tests two operands with logical operators.

Logical operators:

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

Conditional Statements

The conditional statement is satisfied if it evaluates to any value other than zero. The conditional statement can be any valid RIO numeric operand, including variables, array elements, numeric values, functions, keywords, and arithmetic expressions. If no conditional statement is given, the jump will always occur.

Examples:

Number	`V1=6`
Numeric Expression	`V1=V7*6`
	`@ABS[V1]>10`
Array Element	`V1<Count[2]`
Variable	`V1<V2`
Internal Variable	`_TI1=255`
	`_DM<100`
I/O	`V1>@IN[2]`
	`@IN[1]=0`

Multiple Conditional Statements

The RIO will accept multiple conditions 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 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 parentheses for proper evaluation by the RIO. In addition, the RIO executes operations from left to right.*

For example, using variables named V1, V2, V3 and V4:

JP #TEST, (V1<V2) & (V3<V4)

In this example, this statement will cause the program to jump to the label #TEST if V1 is less than V2 and V3 is less than V4. To illustrate this further, consider this same example with an additional condition:

JP #TEST, ((V1<V2) & (V3<V4)) | (V5<V6)

This statement will cause the program to jump to the label #TEST under two conditions: 1) If V1 is less than V2 AND V3 is less than V4. OR 2) If V5 is less than V6.

Using the JP Command:

If the condition for the JP command is satisfied, the RIO branches to the specified label or line number and continues executing commands from this point. If the condition is not satisfied, the RIO board continues to execute the next commands in sequence.

Instruction	Interpretation
`JP #Loop,COUNT<10`	Jump to #Loop if the variable, COUNT, is less than 10
`JS #MOVE2,@IN[1]=1`	Jump to subroutine #MOVE2 if input 1 is logic level high. After the subroutine MOVE2 is executed, the program sequencer returns to the main program location where the subroutine was called.
`JP #BLUE,@ABS[V2]>2`	Jump to #BLUE if the absolute value of variable, V2, is greater than 2
`JP #C,V1*V7<=V8*V2`	Jump to #C if the value of V1 times V7 is less than or equal to the value of V8*V2
`JP#A`	Jump to #A

Using If, Else, and Endif Commands

The RIO provides a structured approach to conditional statements using IF, ELSE and ENDIF commands.

Using the IF and ENDIF Commands

An IF conditional statement is formed by the combination of an IF and ENDIF command. The IF command has arguments of one or more conditional statements. 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 RIO will ignore commands until the associated ENDIF command is executed OR an ELSE command occurs in the program (see discussion of ELSE command below).

Note: An ENDIF command must always be executed for every IF command that has been executed.

Using the ELSE Command

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

Nesting IF Conditional Statements

The RIO allows for IF conditional statements to be included within other IF conditional statements. This technique is known as 'nesting' and the RIO allows up to 255 IF conditional statements to be nested. This is a very powerful technique allowing the user to specify a variety of different cases for branching.

Command Format - IF, ELSE and ENDIF

Function	Condition
IF conditional statement(s)	Execute commands proceeding IF command (up to ELSE command) if conditional statement(s) is true, otherwise continue executing at ENDIF command or optional ELSE command.
ELSE	Optional command. Allows for commands to be executed when argument of IF command evaluates not true. Can only be used with IF command.
ENDIF	Command to end IF conditional statement. Program must have an ENDIF command for every IF command.

Example using IF, ELSE and ENDIF:

Instruction	Interpretation
`#TEST`	Begin Main Program "TEST"
`#LOOP`	Begin loop inside main program
`TEMP=@IN[1]|@IN[2]`	TEMP is equal to 1 if either Input 1 or Input 2 is high
`JS#COND, TEMP=1`	Jump to subroutine if TEMP equals 1
`JP#LOOP`	Loop back if TEMP doesn't equal 1
`EN`	End of main program
`#COND`	Begin subroutine "COND"
`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 INACTIVE"`	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
`ENDIF`	End of 1st conditional statement
`#WAIT`	Label to be used for a loop
`JP#WAIT,(@IN[1]=0) & (@IN[2]=0)`	Loop until both input 1 and input 2 are not active
`EN`	End of subroutine

Stack Manipulation

It is possible to manipulate the subroutine stack by using the ZS command. Every time a JS instruction, interrupt or automatic routine (such as #ININTn or #CMDERR) is executed, the subroutine stack is incremented by 1 (up to a maximum of 16). Normally the stack is restored with an EN instruction. Occasionally it is desirable not to return back to the program line where the subroutine or interrupt was called. The ZS1 command clears 1 level of the stack. This allows the program sequencer to continue to the next line. The ZS0 command resets the stack to its initial value. For example, if an interrupt occurs and the #ININT1 routine is executed, it may be desirable to restart the program sequence instead of returning to the location where the interrupt occurred. To do this, give a ZS (ZS0) command at the end of the #ININT1 routine.

Auto-Start Routine

The RIO has a special label for automatic program execution. A program that has been saved into the RIO non-volatile memory can be automatically executed upon power up or reset, simply by beginning the program with the label #AUTO.

Note: The program must be saved into non-volatile memory using the command, BP.

Automatic Subroutines for Monitoring Conditions

Often it is desirable to monitor certain conditions continuously without tying up the host or RIO program sequences. The RIO can monitor several important conditions in the background. These conditions include checking for the occurrence of a defined input, position error, a command error, or an Ethernet communication error. Automatic monitoring is enabled by inserting a special, predefined label in the applications program. The pre-defined labels are:

SUBROUTINE	DESCRIPTION
#AUTO	Automatic Program Execution on power up
#AUTOERR	Automatic Program Execution on power up if error condition occurs
#ININTn	Input specified by II goes low (n from 0 to 3)
#CMDERR	Bad command given
#TCPERR	Ethernet communication error
#COMINT	Communication Interrupt Routine

For example, the #ININT label could be used to designate an input interrupt subroutine. When the specified input occurs, the program will be executed automatically.

NOTE: An application program must be running for automatic monitoring to function.

Example - Input Interrupt

Instruction	Interpretation
`#A`	Label
`II0,0,1`	Input Interrupt on 1
`#LOOP;JP#LOOP;EN`	Loop
`#ININT0`	Input Interrupt
`MG "INPUT 1 IS HIGH"`	Send Message to screen
`RI0`	Return from interrupt routine to Main Program and do not re-enable trippoints

Example - Command Error

Instruction	Interpretation
`#BEGIN`	Begin main program
`IN "ENTER THE OUTPUT (0-15)", OUT`	Prompt for output number
`SB OUT`	Set the specified bit
`JP #BEGIN`	Repeat
`EN`	End main program
`#CMDERR`	Command error utility
`JP#DONE,_ED<>3`	Check if error on line 3
`JP#DONE,_TC<>6`	Check if out of range
`MG "VALUE OUT OF RANGE"`	Send message
`MG "TRY AGAIN"`	Send message
`ZS1`	Adjust stack
`JP #BEGIN`	Return to main program
`#DONE`	End program if other error
`ZS0`	Zero stack
`EN`	End program

The above program prompts the operator to enter the output port to set. If the operator enters a number out of range (greater than 15), the #CMDERR routine will be executed prompting the operator to enter a new number.

In multitasking applications, there is an alternate method for handling command errors from different threads. Using the XQ command along with the special operands described below allows the controller to either skip or retry invalid commands.

OPERAND	FUNCTION
`_ED1`	Returns the number of the thread that generated an error
`_ED2`	Retry failed command (operand contains the location of the failed command)
`_ED3`	Skip failed command (operand contains the location of the command after the failed command)

The operands are used with the XQ command in the following format:

XQ _ED2 (or _ED3),_ED1,1

Where the ",1" at the end of the command line indicates a restart; therefore, the existing program stack will not be removed when the above format executes.

The following example shows an error correction routine that uses the operands.

Example - Command Error w/Multitasking

Instruction	Interpretation
#A	Begin thread 0 (continuous loop)
JP#A	
EN	End of thread 0
#B	Begin thread 1
N=17	Create new variable
SB N	Set the 17th bit, an invalid value
TY	Issue invalid command
EN	End of thread 1
#CMDERR	Begin command error subroutine
IF _TC=6	If error is out of range (SB 8)
N=1	Set N to a valid number
XQ _ED2,_ED1,1	Retry SB N command
ENDIF	
IF _TC=1	If error is invalid command (TY)
XQ _ED3,_ED1,1	Skip invalid command
ENDIF	
EN	End of command error routine

Example – Ethernet Communication Error

This simple program executes in the RIO and indicates (via the serial port) when a communication handle fails. By monitoring the serial port, the user can re-establish communication if needed.

Instruction	Interpretation
#LOOP	Simple program loop
JP#LOOP	
EN	
#TCPERR	Ethernet communication error auto routine
MG {P1}_IA4	Send message to serial port indicating which handle did not receive proper acknowledgment.
RE	Return to main program

Note: The #TCPERR routine only detects the loss of TCP/IP Ethernet handles, not UDP.

Mathematical and Functional Expressions

Mathematical Operators

For manipulation of data, the RIO provides the use of the following mathematical operators:

Operator	Function
+	Addition
−	Subtraction
*	Multiplication
/	Division
&	Logical And (Bit-wise)
|	Logical Or (On some computers, a solid vertical line appears as a broken line)
( )	Parenthesis
%	Modulus

Mathematical operations are executed from left to right. Calculations within parentheses have precedence.

Examples:

SPEED=7.5*V1/2	The variable, SPEED, is equal to 7.5 multiplied by V1 and divided by 2
COUNT=COUNT+2	The variable, COUNT, is equal to the current value plus 2.
RESULT=Val1 -(@COS[45]*40)	Puts the value of Val1 - 28.28 in RESULT. 40 * cosine of 45° is 28.28
K=@IN[1]&@IN[2]	K is equal to 1 only if Input 1 and Input 2 are high

Note: Mathematical operations can be done in hexadecimal as well as decimal. Just precede hexadecimal numbers with a $ sign so that the RIO recognizes them as such.

Mathematical Operation Precision and Range

The controller stores non-integers in a fixed point representation (not floating point). Numbers are stored as 4 bytes of integer and 2 bytes of fraction within the range of +/- 2,147,483,647.9999. The smallest number representable (and thus the precision) is 1/65536 or approximately 0.000015.

Example:

Using basic mathematics it is known that 1.4*(80,000) = 112,000. However, using a basic terminal, a DMC controller would calculate the following:

```
:var= 1.4*80000;'          Storing the result of 1.4*80000 in var
:MG var;'                  Prints variable "var" to screen
111999.5117
:
```

The reason for this error relies in the precision of the controller. 1.4 must be stored to the nearest multiple of 1/65536, which is 91750/65536 = 1.3999. Thus, (91750/65536)*80000 = 111999.5117 and reveals the source of the error.

By ignoring decimals and multiplying by integers first (since they carry no error), and then adding the decimal back in by dividing by a factor of 10 will allow the user to avoid any errors caused by the limitations of precision of the controller. Continuing from the example above:

```
:var= 14*80000;'           Ignore decimals
```

```
:MG var;'                    Print result
1120000.0000
:var= var/10;'               Divide by 10 to add in decimal
:MG var;'                    Print correct result
112000.0000
:
```

Bit-Wise Operators

The mathematical operators & and | are bit-wise operators. The operator, &, is a Logical And. The operator, |, is a Logical Or. These operators allow for bit-wise operations on any valid RIO numeric operand, including variables, array elements, numeric values, functions, keywords, and arithmetic expressions. The bit-wise operators may also be used with strings. This is useful for separating characters from an input string. When using the input command for string input, the input variable will hold up to 6 characters. These characters are combined into a single value, which is represented as 32 bits of integer and 16 bits of fraction. Each ASCII character is represented as one byte (8 bits), therefore the input variable can hold up to six characters. The first character of the string will be placed in the top byte of the variable and the last character will be placed in the lowest significant byte of the fraction. The characters can be individually separated, by using bit-wise operations as illustrated in the following example:

Instruction	**Interpretation**
`#TEST`	Begin main program
`IN "ENTER",LEN{S6}`	Input character string of up to 6 characters into variable 'LEN'
`FLEN=@FRAC[LEN]`	Define variable 'FLEN' as fractional part of variable 'LEN'
`FLEN=$10000*FLEN`	Shift FLEN by 32 bits (IE - convert fraction, FLEN, to integer)
`LEN1=(FLEN&$00FF)`	Mask top byte of FLEN and set this value to variable 'LEN1'
`LEN2=(FLEN&$FF00)/$100`	Let variable, 'LEN2' = top byte of FLEN
`LEN3=LEN&$000000FF`	Let variable, 'LEN3' = bottom byte of LEN
`LEN4=(LEN&$0000FF00)/$100`	Let variable, 'LEN4' = second byte of LEN
`LEN5=(LEN&$00FF0000)/$10000`	Let variable, 'LEN5' = third byte of LEN
`LEN6=(LEN&$FF000000)/$1000000`	Let variable, 'LEN6' = fourth byte of LEN
`MG LEN6 {S4}`	Display 'LEN6' as string message of up to 4 chars
`MG LEN5 {S4}`	Display 'LEN5' as string message of up to 4 chars
`MG LEN4 {S4}`	Display 'LEN4' as string message of up to 4 chars
`MG LEN3 {S4}`	Display 'LEN3' as string message of up to 4 chars
`MG LEN2 {S4}`	Display 'LEN2' as string message of up to 4 chars
`MG LEN1 {S4}`	Display 'LEN1' as string message of up to 4 chars
`EN`	

This program will accept a string input of up to 6 characters, parse each character, and then display each character. Notice also that the values used for masking are represented in hexadecimal (as denoted by the preceding '$'). For more information, see the section on Sending Messages (page 74).

To illustrate further, if the user types in the string "TESTME" at the input prompt, the RIO will respond with the following:

```
T                    Response from command MG LEN6 {S4}
E                    Response from command MG LEN5 {S4}
S                    Response from command MG LEN4 {S4}
T                    Response from command MG LEN3 {S4}
M                    Response from command MG LEN2 {S4}
E                    Response from command MG LEN1 {S4}
```

Functions

Function	Description
@SIN[n]	Sine of n (n in degrees, with range of -32768 to 32767 and 16-bit fractional resolution)
@COS[n]	Cosine of n (n in degrees, with range of -32768 to 32767 and 16-bit fractional resolution)
@TAN[n]	Tangent of n (n in degrees, with range of -32768 to 32767 and 16-bit fractional resolution)
@ASIN[n]*	Arc Sine of n, between -90° and +90°. Angle resolution in 1/64000 degrees.
@ACOS[n]*	Arc Cosine of n, between 0 and 180°. Angle resolution in 1/64000 degrees.
@ATAN[n]*	Arc Tangent of n, between -90° and +90°. Angle resolution in 1/64000 degrees
@COM[n]	1's Complement of n
@ABS[n]	Absolute value of n
@FRAC[n]	Fraction portion of n
@INT[n]	Integer portion of n
@RND[n]	Round of n (Rounds up if the fractional part of n is .5 or greater)
@SQR[n]	Square root of n (Accuracy is +/-.004)
@IN[n]	Return digital input at general input n (where n starts at 0)
@OUT[n]	Return digital output at general output n (where n starts at 0)
@AN[n]	Return analog input at general input n (where n starts at 0)
@AO[n]	Return analog output at general output n (where n starts at 0)

***:** These functions are multi-valued. An application program may be used to find the correct band.

Functions may be combined with mathematical expressions. The order of execution of mathematical expressions is from left to right and can be over-ridden by using parentheses.

Examples:

```
V1=@ABS[V7]        The variable, V1, is equal to the absolute value of variable V7.
V2=5*@SIN[POS]     The variable, V2, is equal to five times the sine of the variable, POS.
V3=@IN[1]          The variable, V3, is equal to the digital value of input 1.
```

Variables

The number of variables available on the RIO depends on the option ordered, see Table 1.2.These variables can be numbers or strings. A program can be written in which certain parameters, such as I/O status or particular I/O bit, are defined as variables. The variables can later be assigned by the operator or determined by program calculations. Example:

```
SB Red             Uses variable "Red" in SB command
input1=_@IN[1]     Assigns value of digital input 1 status to variable "input1"
```

Programmable Variables

Each variable is defined by a name, which can be up to eight characters. The name must start with an alphabetic character, however, and numbers are permitted in the rest of the name. *Spaces are not permitted.* Variable names should not be the same as RIO instructions. For example, RS is not a good choice for a variable name.

Examples of valid and invalid variable names are:

Valid Variable Names

STATUS1

TEMP1

POINT

Invalid Variable Names

REALLONGNAME ; Cannot have more than 8 characters

123 ; Cannot begin variable name with a number

STAT Z ; Cannot have spaces in the name

Assigning Values to Variables:

Assigned values can be numbers, internal variables and keywords, functions, RIO board parameters and strings; the range for numeric variable values is 4 bytes of integer (2^{31}) followed by two bytes of fraction (+/-2,147,483,647.9999).

Numeric values can be assigned to programmable variables using the equal sign.

Any valid RIO functions can be used to assign a value to a variable. For example, s1=@ABS[V2] or s2=@IN[1]. Arithmetic operations are also permitted.

To assign a string value, the string must be in quotations. String variables can contain up to six characters that must be in quotation.

Examples:

`INTWO=_TI2`	Assigns returned value from TI2 command to variable INTWO.
`INPUT=@IN[1]`	Assigns logical value of input 1 to variable INPUT
`V2=V1+V3*V4`	Assigns the value of V1 plus V3 times V4 to the variable V2.
`Var="CAT"`	Assign the string CAT to variable Var

Displaying the value of variables at the terminal

Variables may be sent to the screen using the format, variable=. For example, V1= , returns the value of the variable V1. V1=? or MG V1 are also valid ways of displaying a variable.

Operands

Operands allow status parameters of the RIO to be incorporated into programmable variables and expressions. Most RIO commands have an equivalent operand - which are designated by adding an underscore (_) prior to the command (see command reference).

Examples of Internal Variables:

`IN1=@IN[1]`	Assigns value of input 1 to the variable IN1.
`JP #LOOP,@AN[0]<2`	Jump to #LOOP if analog input 0 is less than 2
`JP #ERROR,_TC=1`	Jump to #ERROR if the error code equals 1.

Operands can be used in an expression and assigned to a programmable variable, but they cannot be assigned a value. For example: _TI0=1 is invalid.

Special Operands (Keywords)

The RIO provides a few additional operands that give access to internal variables that are not accessible by standard RIO commands.

Operand	Function
_BN	*Returns serial # of the board.
_DA	*Returns the number of arrays available
_DL	*Returns the number of available labels for programming
_DM	*Returns the available array memory
_UL	*Returns the number of available variables
TIME	Free-Running Real Time Clock (Resets with power-on). Note: TIME does not use an underscore character (_) as other keywords.

***:** All these keywords have corresponding commands except for TIME.

Examples of Keywords:

```
V1=_DA          Assign V1 the number of available array names
V3=TIME         Assign V3 the current value of the time clock
```

Arrays

For storing and collecting numerical data, the RIO-47xx0 provides array space for 400 elements. This number is increased to 1000 array elements on the RIO-47xx2. The arrays are one-dimensional, and up to 6 different arrays may be defined. Each array element has a numeric range of 4 bytes of integer (2^{31}) followed by two bytes of fraction (+/-2,147,483,647.9999). Arrays can be used to capture real-time data, such as the bit status of a particular I/O bank.

Defining Arrays

An array is defined with the command DM. The user must specify a name and the number of entries to be held in the array. An array name can contain up to eight characters, starting with an uppercase alphabetic character. The number of entries in the defined array is enclosed in [ ].

Example:

```
DM IOSTAT[100]      Defines an array names IOSTAT with 100 entries
DA *[]              Frees array space using Deallocate command
```

Assignment of Array Entries

Like variables, each array element can be assigned a value. Assigned values can be numbers or returned values from instructions, functions and keywords.

Array elements are addressed starting at count 0. For example, the first element in the OUTPUT array (defined with the DM command, DM OUTPUT[7]) would be specified as OUTPUT[0].

Values are assigned to array entries using the equal sign. Assignments are made one element at a time by specifying the element number with the associated array name.

NOTE: Arrays must be defined using the command, DM, before assigning entry values.

Examples:

```
DM OUTPUT[10]           Dimension Output Array
OUTPUT[1]=3             Assigns the second element of the array, OUTPUT, the value of 3.
OUTPUT[1]=              Returns array element value
OUTPUT[9]=_TI0          Assigns the 10th element of the array, OUTPUT, the value for bank
                        0 digital inputs
data [2]=@COS[POS]*2    Assigns the third element of the array "data" the cosine of the
                        variable POS multiplied by 2.
TIMER[1]=TIME           Assigns the second element of the array timer the returned value
                        of the TIME keyword.
```

Using a Variable to Address Array Elements

An array element number can also be a variable. This allows array entries to be assigned sequentially using a counter.

For example:

Instruction	Interpretation
`#A`	Begin Program
`COUNT=0;DM POS[10]`	Initialize counter and define array
`#LOOP`	Begin loop
`WT 10`	Wait 10 msec
`INPUT[COUNT]=_TI0`	Record bank 0's input bit value into array element
`INPUT[COUNT]=`	Report input bit value
`COUNT=COUNT+1`	Increment counter
`JP #LOOP,COUNT<10`	Loop until 10 elements have been stored
`EN`	End Program

The above example records 10 input bit values for bank 0 at a rate of one value per 10 msec. The values are stored in an array named INPUT. The variable, COUNT, is used to increment the array element counter. The above example can also be executed with the automatic data capture feature described below.

Uploading and Downloading Arrays to On Board Memory

Arrays may be uploaded and downloaded using the QU and QD commands.

QU array[],start,end,delim

QD array[],start,end

where array is an array name such as A[].

Start is the first element of array (default=0)

End is the last element of array (default=last element)

Delim specifies whether the array data is separated by a comma (delim=1) or a carriage return (delim=0).

The file is terminated using <control>Z, <control>Q, <control>D or \.

Automatic Data Capture into Arrays

The RIO provides a special feature for automatic capture of data such as inputs or outputs. Up to four types of data can be captured and stored in four arrays. The capture rate or time interval may be specified. Recording can be done as a one-time event or as a circular continuous recording.

Command Summary - Automatic Data Capture

Command	Description
RA n[],m[],o[],p[]	Selects up to four arrays for data capture. The arrays must be defined with the DM command.
RD type1,type2,type3,type4	Selects the type of data to be recorded, where type1, type2, type3, and type 4 represent the various types of data (see table below). The order of data type is important and corresponds with the order of n,m,o,p arrays in the RA command.
RC n,m	The RC command begins data collection. Sets data capture time interval where n is an integer between 1 and 8 and designates 2^n msec between data. m is optional and specifies the number of elements to be captured. If m is not defined, the number of elements defaults to the smallest array defined by DM. When m is a negative number, the recording is done continuously in a circular manner. _RD is the recording pointer and indicates the address of the next array element. n=0 stops recording.
RC?	Returns a 0 or 1 where, 0 denotes not recording, 1 specifies recording in progress

Data Types for Recording:

Data type	Description
_TIn	Inputs at bank n (0 or 1)
_OPn	Output bank n status (0 or 1)
_AFn	Analog input status (0-7)
_AOn	Analog output status (0-7)

Operand Summary - Automatic Data Capture

_RC	Returns a 0 or 1 where, 0 denotes not recording, 1 specifies recording in progress
_RD	Returns address of next array element.

Deallocating Array Space

Array space may be deallocated using the DA command followed by the array name. DA*[0] deallocates all the arrays.

Output of Data (Numeric and String)

Numerical and string data can be output from the RIO board using several methods. The message command, MG, can output string and numerical data. Also, the RIO can be commanded to return the values of variables and arrays, as well as other information using the interrogation commands, such as V1=? and TZ.

Sending Messages

Messages may be sent using the message command, MG. This command sends specified text and numerical or string data from variables or arrays to the screen.

Text strings are specified in quotes and variable or array data is designated by the name of the variable or array. For example:

MG "The Final Value is", RESULT

In addition to variables, functions and commands, responses can be used in the message command. For example:

MG "The input is", @IN[1]

Formatting Messages

String variables can be formatted using the specifier, {Sn} where n is the number of characters, 1 thru 6. For example:

MG STR {S3}

This statement returns 3 characters of the string variable named STR.

Numeric data may be formatted using the {Fn.m} expression following the completed MG statement. {$n.m} formats data in HEX instead of decimal. The actual numerical value will be formatted with n characters to the left of the decimal and m characters to the right of the decimal. Leading zeros will be used to display specified format.

For example:

MG "The Final Value is", RESULT {F5.2}

If the value of the variable RESULT is equal to 4.1, this statement returns the following:

The Final Value is 00004.10

If the value of the variable RESULT is equal to 999999.999, the above message statement returns the following:

The Final Value is 99999.99

The message command normally sends a carriage return and line feed following the statement. The carriage return and the line feed may be suppressed by sending {N} at the end of the statement. This is useful when a text string needs to surround a numeric value.

Example:

#A

FNAME="John"

LNAME="Smith"

MG "The name is ", FNAME{S3} {N}

MG " ",LNAME{S6}

EN

When #A is executed, the above example will appear on the screen as:

The name is John Smith

Using the MG Command to Configure Terminals

The MG command can be used to configure a terminal. Any ASCII character can be sent by using the format {^n} where n is any integer between 1 and 255.

Example:

MG {^07} {^255}

sends the ASCII characters represented by 7 and 255 to the bus.

Summary of Message Functions:

Function	Description
" "	Surrounds text string
{Fn.m}	Formats numeric values in decimal n digits to the right of the decimal point and m digits to the left
{$n.m}	Formats numeric values in hexadecimal
{^n}	Sends ASCII character specified by integer n
{N}	Suppresses carriage return/line feed
{Sn}	Sends the first n characters of a string variable, where n is 1 thru 6.
{Zn.m}	Formats values like {Fn.m} except leading zeroes are removed
{En}	Outputs message to Ethernet handle n where n is A,B or C
{P1}	Outputs message to Serial port
{M}	Sends Email message (see MA, MD, MS commands)

Displaying Variables and Arrays

Variables and arrays may be sent to the screen using the format, variable= **or** array[x]=. For example, V1= , returns the value of V1.

Removing Leading Zeros from Response

The leading zeros on data returned as a response to interrogation commands or variables and arrays can be removed by the use of the command, LZ. The default value for LZ is 1, meaning that the leading zeroes do not get printed out unless LZ0 command is entered.

Example - Using the LZ command

```
LZ0                 Disables the LZ function
MG@IN[0]            Print input status of bank 1
0000000001.0000     Response from Interrogation Command
                    (With Leading Zeros)

LZ1                 Enables the LZ function
MG@IN[0]            Print input status of bank 1
1.0000              Response from Interrogation Command
                    (Without Leading Zeros)
```

Formatting Variables and Array Elements

The Variable Format (VF) command is used to format variables and array elements. The VF command is specified by:

VF m.n

where m is the number of digits to the left of the decimal point (0 thru 10), and n is the number of digits to the right of the decimal point (0 thru 4).

A negative sign for m specifies hexadecimal format. The default format for VF is VF 10.4

Hex values are returned preceded by a $ and in 2's complement.

:V1=10	Assign V1
:V1=	Return V1
0000000010.0000	Default format
:VF2.2	Change format
:V1=	Return V1
10.00	New format
:VF-2.2	Specify hex format
:V1=	Return V1
$0A.00	Hex value
:VF1	Change format
:V1=	Return V1
9	Overflow

Local Formatting of Variables

VF command is a global format command that affects the format of all relevant returned values and variables. Variables may also be formatted locally. To format locally, use the command, {Fn.m} or {$n.m} following the variable name and the '=' symbol. F specifies decimal and $ specifies hexadecimal. n is the number of digits to the left of the decimal, and m is the number of digits to the right of the decimal. For example:

Examples:

:V1=10	Assign V1
:V1=	Return V1
0000000010.0000	Default Format
:V1={F4.2}	Specify local format
0010.00	New format
:V1={$4.2}	Specify hex format
$000A.00	Hex value
:V1="ALPHA"	Assign string "ALPHA" to V1
:V1={S4}	Specify string format first 4 characters
ALPH	

The local format is also used with the MG command (see page 76).

Programmable I/O

As described earlier, the RIO has 16 digital inputs, 16 digital outputs, 8 analog inputs and 8 analog outputs. The paragraphs below describe the commands that are used for I/O manipulation and interrogation.

Digital Outputs

The most common method of changing the state of digital outputs is by using the set bit 'SB' and clear bit 'CB' commands. The following table shows an example of the SB and CB commands.

Instruction	Interpretation
SB2	Sets bit 2
CB1	Clears bit 1

The Output Bit (OB) instruction is useful for setting or clearing outputs depending on the value of a variable, array, input or expression. Any non-zero value results in a set bit.

Instruction	Interpretation
OB1,POS	Set Output 1 if the variable POS is non-zero. Clear Output 1 if POS equals 0.
OB2,@IN [1]	Set Output 2 if Input 1 is high. If Input 1 is low, clear Output 2.
OB3,@IN[1]&@IN[2]	Set Output 3 only if Input 1 and Input 2 are high.
OB2,COUNT [1]	Set Output 2 if element 1 in array COUNT is non-zero.

The output port can be set by specifying the OP (Output Port) command. This instruction allows a single command to define the state of the entire output bank, where 2^0 is bit 0, 2^1 is bit 1 and so on. A 1 designates that the output is on.

For example:

Instruction	Interpretation
OP6	Sets bits 1 and 2 of bank 0 high. All other bits on bank 0 are 0. ($2^1 + 2^2 = 6$)
OP0,0	Clears all bits of bank 0 and 1
OP0,7	Sets output bits 0, 1 and 2 to one ($2^0 + 2^1 + 2^2$) on bank 1. Clears all bits on bank 0.

The state of the digital outputs can be accessed with the @OUT[n] where n is the output number (Ex: MG@OUT[1] displays the state of output number 1).

Digital Inputs

The digital inputs are accessed by using the @IN[n] function or the TI n command. The @IN[n] function returns the logic level of a specified input, n, where 'n' is the input bit number. The IQ command determines the active level of each input. The TI n command gives the input status of an entire bank, where 'n' is the bank number, 0 or 1. The AI command is a trippoint that pauses program execution until the specified combination of inputs is high or low.

Example – Using Inputs to control program flow

Instruction	Instruction
JP #A,@IN[1]=0	Jump to A if input 1 is low
MG@IN[2]	Display the state of input 2
AI 7&-6	Wait until input 7 is high and input 6 is low

Analog Inputs

Analog inputs are accessed with the @AN[n] function where n is the number assigned to the analog input channel. The returned value will be a voltage reading with 12 bit resolution (16-bit optional for RIO's with the ±10V configurable option). The voltage input range is configurable on ±10V configurable options using the AQ command.

Note: When analog input values are accessed from the Data Record or from the Record Array function, the returned value will be an integer number that represents the analog voltage. For a 0-5V analog input options, the equation used to determine the decimal equivalent of the analog voltage is as follows:

N= (((V-Vlo)*4095)/(Vhi-Vlo))*8

Where N is the integer equivalent of the analog voltage, V is the expected analog voltage, Vlo is the lowest voltage in the total range (0V for the standard analog input module) and Vhi is the highest voltage in the total range (5V for the standard module). The data range for N is 0-32760.

These integer values will also be returned when accessing the analog inputs by the API calls in C/C++ or Visual Basic.

The AQ command also configures the analog inputs to be either 8 single ended (default) or 4 differential inputs.

The AA command is a trippoint that halts program execution until the specified voltage on an analog input is reached. The third field of the AA command controls whether the trippoint will be satisfied when going higher or lower than the voltage. With a command such as AA 1,4.5,0 - if the specified voltage is exceeded prior to arrival at the AA command, the program will continue to execute without a pause. Analog inputs are useful for reading special sensors such as temperature, tension or pressure. The range of AA is dependant on the AQ setting. Here are some examples of using the Analog inputs:

Instruction	**Instruction**
`JP #C,@AN[1]>2`	Jump to A if analog input number 1 is greater than 2 volts
`MG@AN[2]`	Display the analog voltage reading on input 2
`AA 1,4.5,0`	Wait until the voltage on input 1 goes above 4.5V
`AA 1,3.2,1`	Wait until the voltage on input 1 goes below 3.2V

Analog Outputs

Analog output voltage is set with the AO command. The AO command has the format AO m,n where m is the output pin and n is the voltage assigned to it. The analog output voltage is accessed with the @AO[n] function where n is the analog output channel. Analog output modules come with a resolution of 12 bits (16-bit optional). The Analog Output voltage range is configurable using the DQ command when using RIOs with the ±10V configurable option. Use the ID command to see the model number of the RIO.

Note: When analog output values are accessed from the Data Record or from the Record Array function, the returned value will be an integer number that represents the analog voltage. For an RIO with 0-5V analog output option, the equation used to determine the decimal equivalent of the analog voltage is as follows:

N= ((V-Vlo)*4095)/(Vhi-Vlo)

Where N is the integer equivalent of the analog voltage, V is the expected analog voltage, Vlo is the lowest voltage in the total range (0V) and Vhi is the highest voltage in the total range (5V).

These integer values will also be returned when accessing the analog inputs by the API calls in C/C++ or Visual Basic.

The AO command can also be used to set the analog voltage on ModBus devices over Ethernet

Instruction	**Instruction**
`AO 7,1.5`	Set the output voltage on output 7 to 1.5V
`MG@AO[2]`	Display the analog voltage reading on output 2

Analog Process Control Loop

A Process Control Loop allows closed loop control of a process or device. RIO models with Standard Memory have two independent PID filters to provide process control of two devices simultaneously. The *Expanded Memory* models have a total of 6 PID loops available. Analog Process Control Loops are only available on the RIO-472xx when the AO Option (SCB-48608) is ordered. The set of commands shown in the table below are used to set the structure of the Process Control Loop.

Command	Description
AF	Analog Input for feedback
AZ	Analog Output for control
KP	Proportional Gain
KD	Derivative Gain
KI	Integral Gain
IL	Integrator Limit
DB	Deadband
CL	Control Loop Update Rate
PS	Commanded Setpoint
TE	Tell Error
AQ	Analog Input Range
DQ	Analog Output Range

*Note – All PID parameters are burnable except PS, DB, AQ, and DQ. If you issue a BN with the PID's enabled the default values for PS,DB,AQ, and DQ will be in effect upon power up.

To understand how a Process Control Loop works on the RIO, consider an example where it is desirable to control the temperature of an oven. The key items needed to do this are a heater, a temperature sensor, the oven itself, and a RIO unit to control the process. As shown in the diagram below, the heating element is coupled to the "System" which in this case is the oven. The temperature sensor provides feedback to the RIO in the form of an analog input. The RIO unit then compares the desired set-point (entered by the PS command) with the temperature sensor. The difference between the two is called the error "E". The error goes through a PID digital filter and then through a Digital to Analog Converter (DAC) which outputs a control voltage to the heater to close the loop.

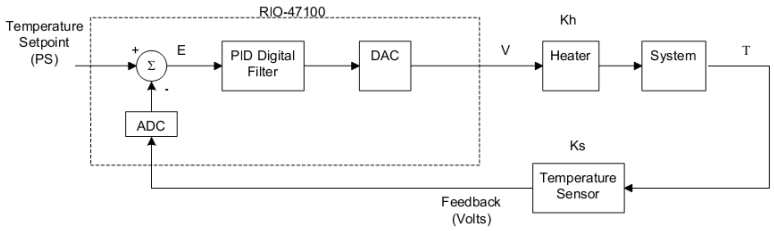


Figure 5.1: Process Control Loop

The example program below uses analog input 0 as the feedback from the temperature sensor and analog output 0 as the control voltage to the heater. An update rate of 25msec was set using the CL command, but a slower update rate could have been chosen due to the slow nature of temperature response. The PID values entered were experimentally found to provide optimum results based on the system. The desired set-point was chosen as 1V. A dead-band of 0.1V was added in order to prevent the system from responding to minor disturbances of the sensor.

```
#PCL
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
```

Note: When the Process Control Loop is enabled, the Analog output voltage is normalized to half of the total voltage input. For instance, with a 0-5V analog input range such as the RIO-47100 – the voltage is normalized to 2.5V. This allows the output to go below 2.5 to compensate for a negative error and above 2.5V to compensate for positive error.

The AQ and DQ must be set on the RIO-47120 to configure the Analog input and output ranges before the process control loops are run and prior to setting AZ & AF. The range of the PS command is dependent on the AQ command.

Current vs Flow Control Mode

The PID loop on the RIO-47xxx by default works as a "current" mode loop. This means that when position error is 0 the analog output will also be set to zero.

Firmware revisions Rev D and newer allow the user to set a negative value for the DB command that will set the Process control loop into a flow control or velocity mode. When DB is set to a negative value, the analog output will be held at its current value and the PID's will be held constant when the feedback is within the range set by the DB command. This mode is preferable for many fluid and temperature control applications.

Real Time Clock

The *Expanded Memory* models of the RIO (see Table 1.1 if your model qualifies) are equipped with a real time clock feature. The real time clock provides true time in seconds, minutes and hours. The RT command provides a method to set the time and operands to return the current time. The default real time clock does not persist through a power cycle and must be set whenever power is restored.

The *Expanded Memory* models can also be ordered with a clock upgrade (-RTC) including a higher precision clock than the default, and a battery backup for the time hardware. All hardware is within the standard sheet metal footprint. The –RTC clock will continue to run when power is removed from the RIO. The –RTC option also provides a calendar function including year, month of year, day of month, and day of week. This feature can be set and queried through the RY command.

Both versions of the real time clock can be set to a TIME protocol (RFC 868) server. Using IH, the RIO can connect to a TIME server over TCP on port 37 and receive the 32bit response. The firmware will then set the time and calendar (if applicable) to the TIME server value. The command RO is used to set the GMT time zone offset for localization of the current time. The TIME protocol synchronization is designed to connect to a server on the local network. Contact Galil if a local server is not available (e.g. an Internet Gateway is required to contact NIST).

See the -RTC section in the Appendix for further details and specifications for the real time clock.

Appendix

Electrical Specifications

Input/Output

Digital I/O	See Chapter 4 I/O.
DAC Output Current	4mA max output per channel
+5V Reference	10mA max output
47120: ±12V out	10mA max output
47x42: ±12V out	10mA max output

Power Requirements for EXT/AUX Power Option

Model	Input Voltage Range	Minimum Power*	Maximum Power*
RIO-4710x	18-36 VDC	1.4 Watts	4 Watts
RIO-4712x	18-36 VDC	2.6 Watts	4 Watts
RIO-47142	9-48 VDC	2.6 Watts	4 Watts
RIO-472xx	18-36 VDC	2.1 Watts	4 Watts
RIO-47300	9-48 VDC	2.6 Watts	4 Watts

*Power ratings with no external connections to the RIO.

Before connecting power to the RIO, read the section: Step 2. Connect Power to the RIO.

Power Supply Options

Galil offers several power supply options as accessories to the RIO. For more details regarding Power Supplies see A3 - Power Supplies and a list of RIO accessories can be under Accessories.

Certifications

The RIO-471xx is certified for the following when the product or package is marked.

ETL

CE

http://www.galilmc.com/products/ce_documents/rio47000_ce_dc.pdf

ROHS

ROHS Compliant

Ordering Options

The RIO-47xxx can be ordered in many different configurations and with different options. This section provides information regarding the different options available on the RIO-47xxx. For more information on pricing and how to order an RIO with these options, see our RIO-47xxx part number generator on our website.

http://www.galilmc.com/products/rio-47xxx-part-number.php

-DIN

If ordered with the –DIN option the RIO has a DIN rail mount attached to the case. This option is valid for all RIO-471xx controllers. It is not valid for the RIO-472xx family as the RIO-472xx comes in a DIN rail mount by default.

Part number ordering example: RIO-47100-DIN

-NO DIN

This option is only valid with the RIO-472xx. This option removes the din rail clips. The unit will still be in a plastic tray.

Part number ordering example: RIO-47200-NO DIN

-422

This option allows the RIO to communicate via RS-422 instead of RS-232.

Pin	Description	Pin	Description
1	RTS-	6	RTS+
2	TXD-	7	TXD+
3	RXD-	8	RXD+
4	CTS-	9	CTS+
5	GND		

Part number ordering example: RIO-47100-422

-RTC

RIO models with *Expanded Memory* (See if your model does in Table 1.1) come standard with some real time clock features. The –RTC option provides an extended feature set as shown below in Table 6.1.

Real time clock	Expanded Memory	with -RTC option
RT providing hours, minutes, seconds	Yes	Yes
RY providing year, month, date, and day of the week	No	Yes
Settable via TIME protocol server (IH and RO commands)	Yes	Yes
Clock persists through RIO power loss	No	Yes
C No-power clock battery life	N/A	More than 1 week[1]

Table 6.1: Real time clock features and expanded -RTC features set.

[1]Time till failure pending at the time of publication

Part number ordering example: RIO-47122-RTC

-12V

This option allows for the RIO to be powered from a 10.5 to 15 VDC source (standard is 18 to 36 VDC). This option is only available for the RIO-471xx products. Contact Galil if this option is needed on a RIO-472xx.

The RIO will no longer have the option to be powered over PoE with this modification.

Part number ordering example: RIO-47120-12V

-2LSRC

If a RIO-471xx is ordered with the -2LSRC option then outputs 8-15 are configured to source current. They will be capable of 5-24VDC with 25mA of current in a sourcing configuration. See 25mA Low Power Sourcing Outputs (LSRC) in Chapter 4 for more information.

Part number ordering example: RIO-47100-(1HSRC,2LSRC)

-1LSNK/-1LSRC & -2LSNK/-2LSRC

These four options are only available on the RIO-472xx. By default the RIO-472xx has all 16 high power outputs. These options allow either of the two banks of 8 outputs to be configured for low power sinking or low power sourcing. For example, if output 0-7 need to be configured for low power sourcing and outputs 8-15 need to be configured for high power sourcing the option would be (1LSRC, 2HRSC). The circuits for low power sourcing and sinking will be the same as the circuits for the low power outputs previously defined in Chapter 4.

Part number ordering example: RIO-47200-(1LSNK,2LSRC), where
1LSNK: Outputs 0-7 low power sinking
2LSRC: Outputs 8-15 low power sourcing

-QUAD, -SSI, and -BiSS

Most RIO models utilize Digital Inputs 12,13,14 and 15 and Digital Outputs 12,13,14 and 15 as encoder inputs. These digital inputs and outputs will not be available as standard digital I/O when the -QUAD, -SSI, or BiSS option is ordered.

RIO-472x2 and RIO-47300 models have additional board that does not require sacrificing I/O for the benefit of encoder inputs, but modifies the physical dimensions of the unit, see Special Note for RIO-47202 and Special Note for RIO-47300 below for more details.

The QE command is used to read the encoder register, the WE command sets the current position of the encoder (-QUAD only) and the SE command configures the encoder when the –SSI option is ordered. The register that is read using the QE command is updated by the RIO every 25ms. See the QE, WE and SE commands in the RIO command reference for more information.

Part number ordering example: RIO-47122-QUAD

Electrical Specifications

Input buffers:	AM26LV32
Output buffers (SSI Clock):	AMP26LV31
-QUAD maximum frequency:	8 MHz[1]
Single Ended Encoders:	Connect to A+/B+, leave A-/B- floating.

[1] Hardware update rate. Register read by the QE command is updated at a rate of 40Hz.

Special Note for RIO-47202

The RIO-47202 allows for QUAD, SSI, or BiSS encoder monitoring through an optional plug-in screw terminal board similar to the SCB-48608 shown in Figure A.1 below. The Encoder monitoring option has a 25msec update rate and is not available with the Analog output option. An external power source is required to power the encoders.

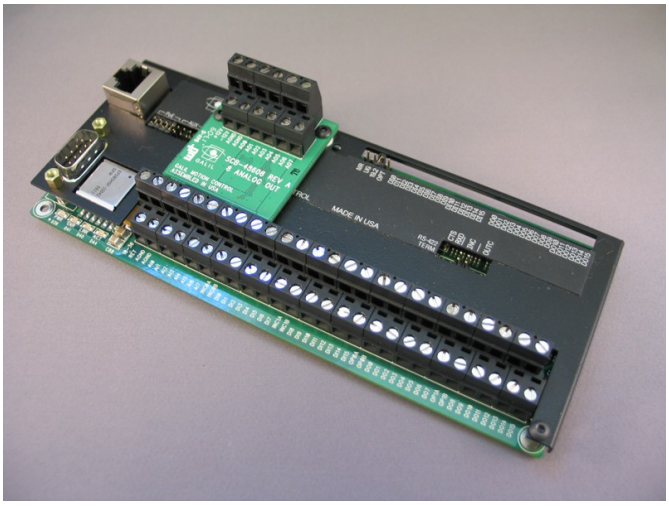


Figure A.1: RIO-472xx modification to allow for BiSS

Special Note for RIO-47300

The RIO-47300 allows for QUAD, SSI, or BiSS through an expanded board as shown in Figure A.2 below.

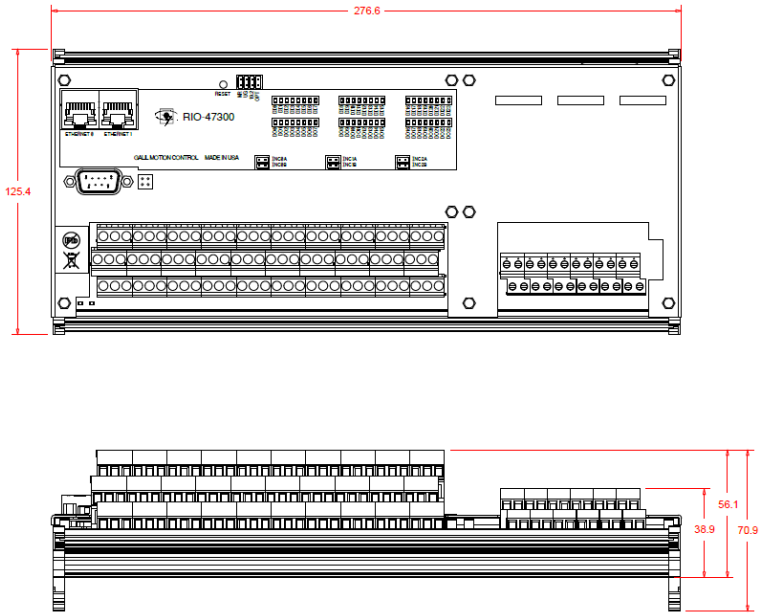


Figure A.2: DMC-47300 Quad, SSI, and BiSS Option dimension changes

-QUAD Pinout

RIO-47122 & RIO-47142

RIO-47122 Pin	RIO-47142 Pin	Label	Encoder Signal
24	24	DO14	Channel 0 A+
39	39	DO12	Channel 0 A-
38	38	DO15	Channel 0 B+
9	9	DO13	Channel 0 B-
31	31	DI14	Channel 1 A+
2	2	DI12	Channel 1 A-
1	1	DI15	Channel 1 B+
17	17	DI13	Channel 1 B-
41	4	N/C	Ground

Table A.2: The encoder pins located on the "DIGITAL" 44-pin HD D-sub connector

RIO-47202

Label	Encoder Signal
0A+	Channel 0 A+
0A-	Channel 0 A-
0B+	Channel 0 B+
0B-	Channel 0 B-
1A+	Channel 1 A+
1A-	Channel 1 A-
1B+	Channel 1 B+
1B-	Channel 1 B-
GND	Ground

Table A.3: The screw-terminal labels for the encoder connections

RIO-47300

Label	Encoder Signal
0A+	Channel 0 A+
0A-	Channel 0 A-
0B+	Channel 0 B+
0B-	Channel 0 B-
1A+	Channel 1 A+
1A-	Channel 1 A-
1B+	Channel 1 B+
1B-	Channel 1 B-
+5	+5V reference
GND	Ground

Table A.4: The screw-terminal labels for the encoder connections

-SSI/BiSS Pinout

RIO-47122 & RIO-47142

RIO-47122 Pin	RIO-47142 Pin	Label	Encoder Signal
24	24	DO14	Channel 0 Clock+
39	39	DO12	Channel 0 Clock-
38	38	DO15	Channel 0 Data+
9	9	DO13	Channel 0 Data-
31	31	DI14	Channel 1 Clock+
2	2	DI12	Channel 1 Clock -
1	1	DI15	Channel 1 Data+
17	17	DI13	Channel 1 Data-
41	4	N/C	Ground

Table A.5: The encoder pins located on the "DIGITAL" 44-pin HD D-sub connector

RIO-47202

Label	Encoder Signal
0A+	Channel 0 Clock+
0A-	Channel 0 Clock-
0B+	Channel 0 Data+
0B-	Channel 0 Data-
1A+	Channel 1 Clock+
1A-	Channel 1 Clock -
1B+	Channel 1 Data+
1B-	Channel 1 Data-
GND	Ground

Table A.6: The screw-terminal labels for the encoder connections as seen on the SCB-48290

RIO-47300

Label	Encoder Signal
0A+	Channel 0 Clock+
0A-	Channel 0 Clock-
0B+	Channel 0 Data+
0B-	Channel 0 Data-
1A+	Channel 1 Clock+
1A-	Channel 1 Clock -
1B+	Channel 1 Data+
1B-	Channel 1 Data-
+5	+5V reference
GND	Ground

Table A.7: The screw-terminal labels for encoder connections

-PWM

The DY, PM and FQ commands[1] are used on the RIO-47xxx to output 3.3 V PWM signals on Digital Outputs 14 and 15. These outputs are normally optoisolated, which limits the PWM frequency to a maximum of 50 Hz. The -PWM option bypasses optoisolation and provides buffered outputs for Digital Outputs 14 and 15, increasing the maximum frequency to 20 kHz. The location of the reference ground for these signals will vary from model to model, see Table A.8 for the reference ground connections on models that include the -PWM option.

[1]These commands are only available on firmware Rev 1.1d and above.

RIO Part Number	Connector	Label
RIO-47122	Pin 41 on 44 Pin D-Sub Connector	N/C
RIO-47142	Pin 41 on 44 Pin D-Sub Connector	N/C
RIO-47202	Screw Terminal	AGND
RIO-47300	Screw Terminal	GND

Table A.8: Reference Ground connections for PWM signals

The PWM output frequency is set with the FQ command. The actual output frequency f_a measured at Digital Outputs 14 and 15 can be calculated as shown below.

- 10 – 311 Hz
  In this frequency range, f_a is equal to FQ, accurate to ±1 Hz.

$$f_a = \frac{80{,}000\ Hz}{@RND\left[\frac{80{,}000\ Hz}{FQ}\right]}$$

Figure A.3: Output frequency calculation for frequencies 311 - 20,000 Hz

- 312 – 20,000 Hz
  In this frequency range, f_a is calculated as shown below in Figure A.3, accurate to ±0.1 Hz.

The rounding function results in a frequency resolution that is dependent on the commanded frequency FQ. This relationship is shown in Figure A.4.

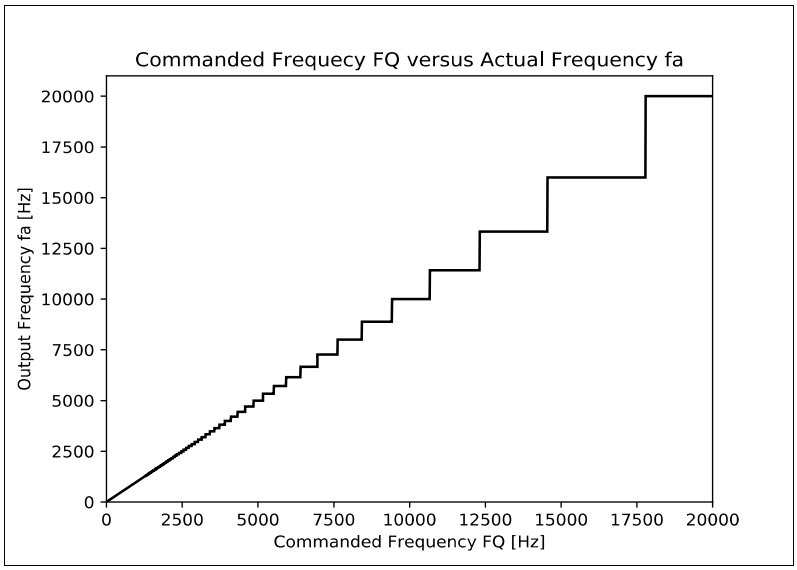


Figure A.4: Commanded Frequency FQ vs F_a

Electrical Specifications for DO15:14 with –PWM option

V_O Output Voltage Range:	0V to 3.3V
I_O Current output - Sink/Source:	20 mA (Max)
Part number ordering example:	RIO-47102-PWM

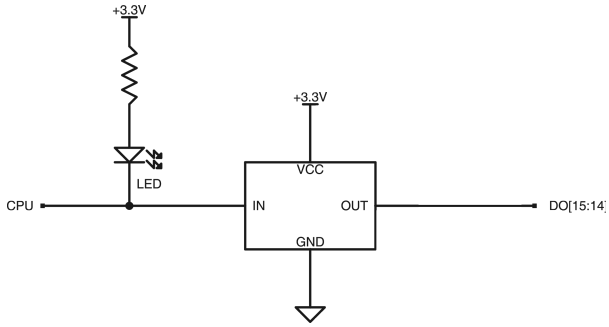


Figure A.5: -PWM option

-HS

This option changes digital input 3 (DI3) to a high speed digital input. It is available on expanded memory RIO models (RIO-47002 and RIO-47300) as a standard option. With this option, the input becomes a TTL level input that is differential with respect to digital input 2 (DI2 is not available as an input with the –HS option). The maximum frequency of pulses that can be captured is increased to 3Mhz (50% duty cycle). If higher values are required, please consult factory.

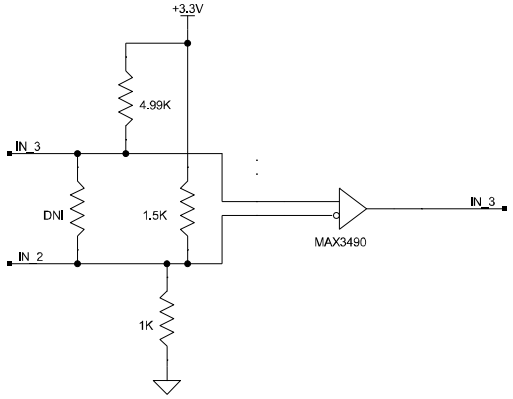


Figure A.6: HS Option Schematic

Part number ordering example: RIO-47102-HS

Note: the termination resistor labeled 'DNI' in Figure A.6 is not installed with the -HS option. Contact Galil if termination resistor is required.

-16Bit

The -16 option specifies 16 bit resolution on the analog inputs and outputs. This option is valid on the RIO-4712x, RIO-47142, RIO-472xx (see AI_10v16Bit), and RIO-47300 only.

Part number ordering example: RIO-47120-16bit

-(AI_10v12Bit)

This option changes the analog inputs on the RIO-472xx to accept +-10V analog signals with 12 bit resolution. The range of the analog inputs can be changed with the AQ command, similar to the RIO-4712x.

Part number ordering example: RIO-47200-(AI_10v12bit)

-(AI_10v16Bit)

This option changes the analog inputs on the RIO-472xx to accept +-10V analog signals with 16 bit resolution. The range of the analog inputs can be changed with the AQ command, similar to the RIO-4712x.

Part number ordering example: RIO-47200-(AI_10v16bit)

-(4-20mA)

This option installs resistors in parallel with each analog input. On RIO's with 0-5V analog input ranges the resistor is 237 ohms and on RIO's with +/-10V analog input ranges the resistor value is 475 ohms (1%).

An RIO with +/-10V analog inputs should be configured for 0-10V range (AQ n, 4). With this setting, the range for 4-20mA will be 1.9V-9.5V.

The equation for calculating the current for an RIO with +/-10V analog inputs is:

$I_{mA}=2.105\ V$

The equation for calculating the current for an RIO with 0-5V analog inputs is:

$I_{mA}=2.11\ V$

Where I_{mA} = current in mA

V = Voltage reading from RIO

Part number ordering example: RIO-47120-(4-20mA)

AO Option (SCB-48608)

The RIO-472xx by default does not have analog outputs however analog outputs can be added using the AO option. When analog outputs are added, a new screw terminal board is added called the SCB-48608 and is attached to the RIO-472xx at the factory (cannot be installed in the field). This board supplies 8 analog outputs to the RIO-472xx.

The option can be ordered with ±10V configurable analog outputs in either 12 or 16 bits – same as RIO-4712x, or with 0-5V analog outputs 12 bit resolution – same as RIO-4710x. See the DQ command for specifics on the ±10V configurable options.

The ±12V terminals will provide ±12V output only when the outputs are ordered as ±10V configurable outputs. Maximum current draw is 10mA each.

When then 0-5V analog outputs are ordered the ±12V terminals will be No Connects.

(8AO_5v12bit)

This option adds 12 bit 0-5V analog outputs via the SCB-48608 on the RIO-472xx. See 0-5V Analog Outputs in Chapter 4 for more information.

Part number ordering example: RIO-47200-(8AO_5v12bit)

Qty 8, 0-5V analog outputs with 12 bit resolution.

(8AO_10v12bit)

This option adds 12 bit ±10V configurable analog outputs via the SCB-48608 on the RIO-472xx. See +/-10V Configurable Analog Outputs in Chapter 4 for more information.

Part number ordering example: RIO-47200-(8AO_10v12bit)

Qty 8, ±10V configurable analog outputs with 12 bit resolution.

(8AO_10v16bit)

This option adds 16 bit ±10V configurable analog outputs via the SCB-48608 on the RIO-472xx. See +/-10V Configurable Analog Outputs in Chapter 4 for more information.

Part number ordering example: RIO-47200-(8AO_10v16bit)

Qty 8, ±10V configurable analog outputs with 16 bit resolution.

-24ExIn & -24ExOut

The I/O capability of the RIO-47300 can be expanded with an additional 24 digital optoisolated inputs or outputs. NOTE: The extended I/O is configured as either inputs *or* outputs and should be wired based on the option ordered. For alternative configurations contact Galil.

Dimensions

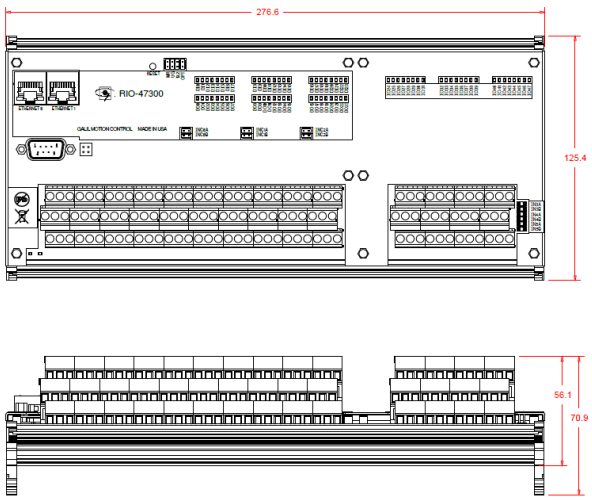
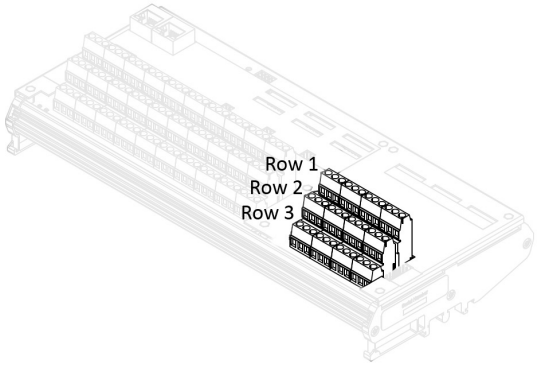


Figure A.7: RIO-47300-27ExIn/Out Dimensions

Pin-outs



Row 1		Row 2		Row 3	
Label	**Description**	**Label**	**Description**	**Label**	**Description**
INC3B [1]	Input Reference Ground (Bank 3)	INC3A	Input Common (bank 3)	OP3B	Output GND (Bank 3)
OP3A	Output PWR (Bank 3)	IO24	Input or Output 24	IO25	Input or Output 25
IO26	Input or Output 26	IO27	Input or Output 27	IO28	Input or Output 28
IO29	Input or Output 29	IO30	Input or Output 30	IO31	Input or Output 31
INC4B [1]	Input Reference Ground (Bank 4)	INC4A	Input Common (Bank 4)	OP4B	Output GND (Bank 4)
OP4A	Output PWR(Bank 4)	IO32	Input or Output 32	IO33	Input or Output 33
IO34	Input or Output 34	IO35	Input or Output 35	IO36	Input or Output 36
IO37	Input or Output 37	IO38	Input or Output 38	IO39	Input or Output 39
INC5B [1]	Input Reference Ground (Bank 5)	INC5A	Input Common (Bank 5)	OP5B	Output GND (Bank 5)
OP5A	Output PWR (Bank 5)	IO40	Input or Output 40	IO41	Input or Output 41
IO42	Input or Output 42	IO43	Input or Output 43	IO44	Input or Output 44
IO45	Input or Output 45	IO46	Input or Output 46	IO47	Input or Output 47

[1]Rarely used, but if wired improperly will cause damage to the controller. Only to be used when the INC jumpers are installed. See INC jumpers section for more detail.

Electrical Specifications

Inputs (-24ExIn Option)

The -24ExIn option provides an additional 24 optoisolated inputs with the same electrical specifications as listed under Digital Inputs, Electrical Specifications, pg 46. Figure A.8 - Figure A.10 provide the wiring diagrams for these inputs.

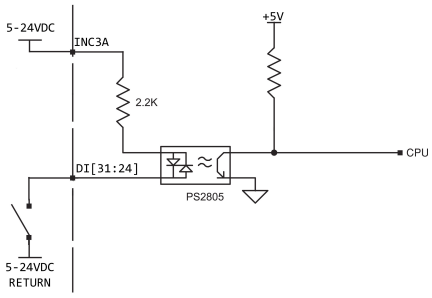


Figure A.8: Wiring diagram for DI[31:24], Bank 3

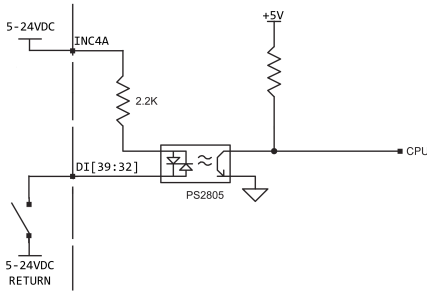


Figure A.9: Wiring diagram for DI[39:32], Bank 4

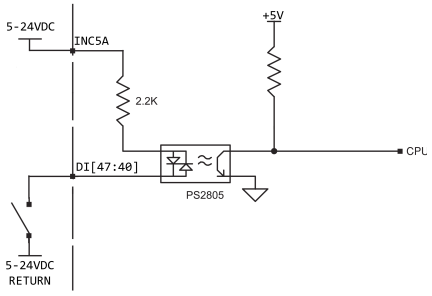


Figure A.10: Wiring diagram for DI[47:40], Bank 5

INC Jumpers

The INC jumpers on the -24ExIN option provides the same functionality as described in Digital Inputs, INC jumpers, pg 49. Figure 4.11 below provides a wiring jumper for using INC jumpers with input Banks 3-5.

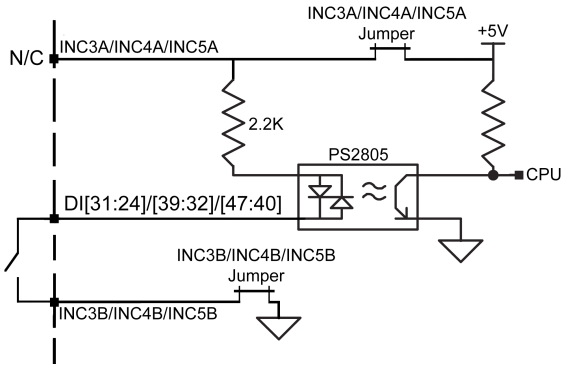


Figure A.11: INC jumpers wiring diagram for input Banks 3-5

Outputs (-24ExOut Option)

The -24ExOut option provides an additional 24, 500mA sourcing, optoisolated outputs with the same electrical specifications as listed under Digital Outputs, 500mA Sourcing Outputs (HSRC), pg 42. Figure A.12 - Figure A.14 provide the wiring diagrams for digital output Banks 3-5.

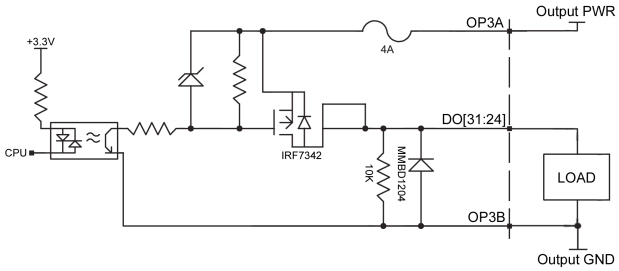


Figure A.12: Wiring diagram for DO[31:24], Bank 3

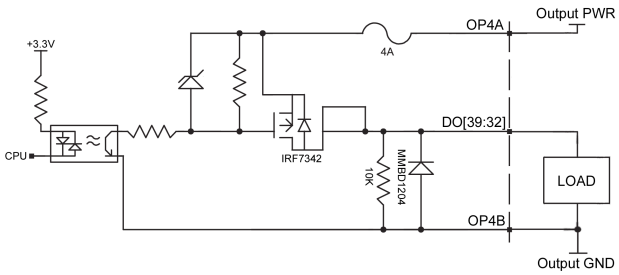


Figure A.13: Wiring diagram for DO[39:32], Bank 4

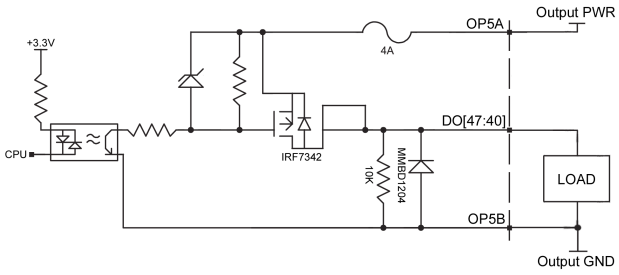


Figure A.14: Wiring diagram for DO[47:40], Bank 5

Connectors for RIO-47xxx

RIO-471xx - 44 pin D-Sub Connector

Pin	Label	Description	Pin	Label	Description	Pin	Label	Description
1	DI15	Digital Input 15	16		No connect / INC1B [1]	31	DI14	Digital Input 14
2	DI12	Digital Input 12	17	DI13	Digital Input 13	32	DI11	Digital Input 11
3	DI9	Digital Input 9	18	DI10	Digital Input 10	33	DI8	Digital Input 8
4	N/C	No Connect	19	INC1/ INC1A	Input Common (Bank 1)	34	N/C [1]	No Connect / INC0B
5	DI6	Digital Input 6	20	DI7	Digital Input 7	35	DI5	Digital Input 5
6	DI3 [4]	Digital Input 3	21	DI4	Digital Input 4	36	DI2 [4]	Digital Input 2
7	DI0	Digital Input 0	22	DI1	Digital Input 1	37	INC0/ INC0A	Input Common (Bank 0)
8	OP1B [2]	Output PWR/GND (Bank 1)	2	N/C	No Connect	38	DO15 [5]	Digital Output 15
9	DO13	Digital Output 13	24	DO14 [5]	Digital Output 14	39	DO12	Digital Output 12
10	DO10	Digital Output 10	25	DO11	Digital Output 11	40	DO9	Digital Output 9
11	OP1A [3]	Output GND/PWR (Bank 1)	26	DO8	Digital Output 8	41	N/C [6]	No Connect/OP1A
12	DO7	Digital Output 7	27	OP0B	Output GND (Bank 0)	42	DO6	Digital Output 6
13	DO4	Digital Output 4	28	DO5	Digital Output 5	43	DO3	Digital Output 3
14	DO1	Digital Output 1	29	DO2	Digital Output 2	44	DO0	Digital Output 0
15	OP0A	Output PWR (Bank 0)	30	OP0A	Output PWR (Bank 0)			

[1] Input Reference Ground. Rarely used, but if wired improperly will cause damage to the controller. Only to be used when the INC jumpers are installed. See INC jumpers section for more detail.

[2] When ordered with -2LSRC this pin will actually be Output Power Ground for Bank 1.

[3] When ordered with -2LSRC this pin will actually be +5-24V Output Power Supply for Bank 1.

[4] When ordered with -HS option DI3 is high-speed input+ and DI2 is high-speed input- (DI2 is lost)

[5] PWM outputs. See -PWM option in Appendix and Chapter 4 I/O.

[6] Output PWR (Bank 1) for RIO-47142. No connect for all other RIO models.

Note: For inputs Bank 0 is DI[7:0] and Bank 1 is DI[15:8]. For outputs Bank 0 is DO[7:0] and Bank 1 is DO[15:8].

RIO-471xx - 26 pin D-Sub Connector

Pin	Label	Description	Pin	Label	Description	Pin	Label	Description
1	N/C	No Connect	10	N/C	No Connect	19	N/C	No Connect
2	N/C +12V	47100: No Connect 47120/47142: +12V out	11	RSV	Reserved	20	N/C -12V	47100: No Connect 47120/47142: -12V out
3	AI7	Analog Input 7	12	GND	Ground	21	AI6	Analog Input 6
4	AI4	Analog Input 4	13	AI5	Analog Input 5	22	AI3	Analog Input 3
5	AI1	Analog Input 1	14	AI2	Analog Input 2	23	AI0	Analog Input 0
6	GND	Ground	15	GND	Ground	24	AO7	Analog Output 7
7	AO5	Analog Output 5	16	AO6	Analog Output 6	25	AO4	Analog Output 4
8	AO2	Analog Output 2	17	AO3	Analog Output 3	26	AO1	Analog Output 1
9	GND	Ground	18	AO0	Analog Output 0			

RIO-472xx - Screw Terminals

Label	Description	Label	Description
18-36	18-36VDC logic power input	DI10	Digital Input 10
RET	Return side of logic power input	DI11	Digital Input 11
AGND	Analog Ground	DI12	Digital Input 12
AGND	Analog Ground	DI13	Digital Input 13
AI0	Analog Input 0	DI14	Digital Input 14
AI1	Analog Input 1	DI15	Digital Input 15
AI2	Analog Input 2	OP0A[4]	Output PWR/GND (Bank 0)
AI3	Analog Input 3	OP0B[4]	Output GND/PWR (Bank 0)
AI4	Analog Input 4	DO0	Digital Output 0
AI5	Analog Input 5	DO1	Digital Output 1
AI6	Analog Input 6	DO2	Digital Output 2
AI7	Analog Input 7	DO3	Digital Output 3
INC0A	Input Common (Bank 0)	DO4	Digital Output 4
INC0B[3]	Input Reference Ground (Bank 0)	DO5	Digital Output 5
DI0	Digital Input 0	DO6	Digital Output 6
DI1	Digital Input 1	DO7	Digital Output 7
DI2[2]	Digital Input 2	OP1A[4]	Output PWR/GND (Bank 1)
DI3[2]	Digital Input 3	OP1B[4]	Output GND/PWR (Bank 1)
DI4	Digital Input 4	DO8	Digital Output 8
DI5	Digital Input 5	DO9	Digital Output 9
DI6	Digital Input 6	DO10	Digital Output 10
DI7	Digital Input 7	DO11	Digital Output 11
INC1A	Input Common (Bank 1)	DO12	Digital Output 12
INC1B[3]	Input Reference Ground (Bank 1)	DO13	Digital Output 13
DI8	Digital Input 8	DO14[1]	Digital Output 14
DI9	Digital Input 9	DO15[1]	Digital Output 15

[1] PWM outputs. See -PWM option in Appendix and Chapter 4 I/O.

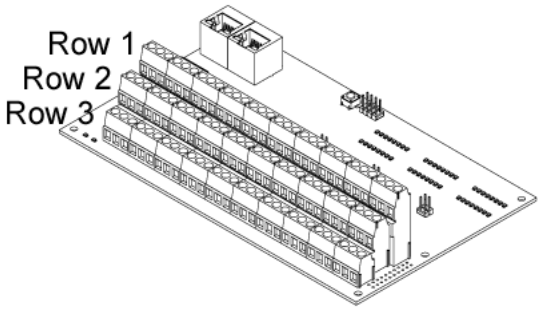
[2] When ordered with -HS option DI3 is high-speed input+ and DI2 is high-speed input- (DI2 is lost)

[3] Rarely used, but if wired improperly will cause damage to the controller. Only to be used when the INC jumpers are installed. See INC jumpers section for more detail.

[4] When ordered with the -1LSNK/-1LSRC & -2LSNK/-2LSRC options, the purpose of this pin changes from either Output PWR to GND or vice verse depending on the option. Be sure to check the -1LSNK/-1LSRC & -2LSNK/-2LSRC for reference and Digital Outputs section for correct wiring.

Note: For inputs Bank 0 is DI[7:0] and Bank 1 is DI[15:8]. For outputs Bank 0 is DO[7:0] and Bank 1 is DO[15:8].

RIO-47300 – Screw Terminals



Row 1		Row 2		Row 3	
Label	**Description**	**Label**	**Description**	**Label**	**Description**
9-48	9-48VDC logic power input	GND	Digital ground	GND	Digital ground
+12	+12V output reference	-12	-12V output reference	+5A	+5V output analog reference
AGND	Analog ground	AI0	Analog input 0	AI1	Analog input 1
AI2	Analog input 2	AI3	Analog input 3	AI4	Analog input 4
AI5	Analog input 5	AI6	Analog input 6	AI7	Analog input 7
AGND	Analog ground	AGND	Analog ground	AGND	Analog ground
AGND	Analog ground	AO0	Analog output 0	AO1	Analog output 1
AO2	Analog output 2	AO3	Analog output 3	AO4	Analog output 4
AO5	Analog output 5	AO6	Analog output 6	AO7	Analog output 7
INC0A	Input Common (Bank 0)	DI0	Digital input 0	DI1	Digital input 1
DI2[3]	Digital input 2	DI3[3]	Digital input 3	DI4	Digital input 4
DI5	Digital input 5	DI6	Digital input 6	DI7	Digital input 7
INC0B[1]	Input Reference Ground (Bank 0)	N/C	No connect	OP0B	Output GND (Bank 0)
OP0A	Output PWR (Bank 0)	DO0	Digital output 0	DO1	Digital output 1
DO2	Digital output 2	DO3	Digital output 3	DO4	Digital output 4
DO5	Digital output 5	DO6	Digital output 6	DO7	Digital output 7
INC1A	Input Common (Bank 1)	DI8	Digital input 8	DI9	Digital input 9
DI10	Digital input 10	DI11	Digital input 11	DI12	Digital input 12
DI13	Digital input 13	DI14	Digital input 14	DI15	Digital input 15
INC1B[1]	Input Reference Ground (Bank 1)	N/C	No connect	OP1B	Output GND (Bank 1)
OP1A	Output PWR (Bank 1)	DO8	Digital output 8	DO9	Digital output 9
DO10	Digital output 10	DO11	Digital output 11	DO12	Digital output 12
DO13	Digital output 13	DO14[2]	Digital output 14	DO15[2]	Digital output 15
INC2A	Input Common (Bank 2)	DI16	Digital input 16	DI17	Digital input 17
DI18	Digital input 18	DI19	Digital input 19	DI20	Digital input 20
DI21	Digital input 21	DI22	Digital input 22	DI23	Digital input 23
INC2B[1]	Input Reference Ground (Bank 2)	N/C	No connect	OP2B	Output GND (Bank 2)
OP2A	Output PWR (Bank 2)	DO16	Digital output 16	DO17	Digital output 17
DO18	Digital output 18	DO19	Digital output 19	DO20	Digital output 20
DO21	Digital output 21	DO22	Digital output 22	DO23	Digital output 23

[1]Rarely used, but if wired improperly will cause damage to the controller. Only to be used when the INC jumpers are installed. See INC jumpers section for more detail.

[2]PWM outputs. See -PWM option in Appendix and Chapter 4 I/O.

[3]When ordered with -HS option DI3 is high-speed input+ and DI2 is high-speed input- (DI2 is lost)

Note: For inputs Bank 0 is DI[7:0], Bank 1 is DI[15:8], and Bank 2 is DI[23:16]. For outputs Bank 0 is DO[7:0], Bank 1 is DO[15:8], and Bank 2 is DO[23:16].

RS-232 Port: DB-9 Pin Male

The location of the RS-232 on the board varies slightly with product. Use the table below as reference:

Product	Location
RIO-471xx	J2
RIO-472xx	J2
RIO-47300	J3

The RS232 uses a standard connector and cable, 9-Pin:

Pin	Signal
1	No Connect
2	TXD
3	RXD
4	No Connect
5	Ground
6	No Connect
7	CTS
8	RTS
9	No Connect

Note: A straight-through serial cable should be used to connect the RIO to a standard PC serial port.

Ethernet Port: 10/100 Base-T (RJ-45)

The location of the Ethernet ports on the board varies slightly with product. Some products will also have a dual-Ethernet port. Use the table below as reference:

Product	Port 1	Port 2
RIO-4712x RIO-4710x	J1	–
RIO-47142	J1	J7
RIO-472xx	J1	–
RIO-47300	J1	J2

The pin-outs for each Ethernet port is the same between products and single versus Dual-Ethernet ports. Their pin-outs are listed below:

Pin	Signal
1	TXP
2	TXN
3	RXP
4	Reserved
5	Reserved
6	RXN
7	Reserved
8	Reserved

Power: J5, 2-pin Molex

Please see the Step 2. Connect Power to the RIO for instructions on connecting power to the RIO and Power Requirements for EXT/AUX Power Option. This connector is not used when powering the RIO via POE.

Pin	Signal
1	GND (Ground)
2	DC Voltage Supply +

On Board Connector	Common Mating Connectors[1]	Crimp Part Number	Type
MOLEX# 39-31-0020	MOLEX# 39-01-2025	MOLEX# 44476-3112	2 Position

[1]The mating connectors listed are not the only mating connectors available from Molex. See http://www.molex.com/ for the full list of available mating connectors.

Jumper Descriptions

RIO-4710x/4712x

Jumper	Label	Function (If jumpered)
J5	MRST	Master Reset enable. Returns RIO to factory default settings and erases non-volatile memory. Requires power-on or RESET to be activated.
	UPGD	Used to upgrade the controller if the unit becomes unresponsive.
	19.2	Set baud Rate to 19.2k (default without jumper is 115k)
	OPT	10BaseT Ethernet Communication

Jumper	Label	Function (If jumpered)
JP6	EXT (4 jumpers)	Power for board comes from external power source, see Step 2. Connect Power to the RIO and Power Requirements for EXT/AUX Power Option.
JP7	PoE (4 jumpers)	Power for board comes from Power over Ethernet (No power cable is necessary – Ethernet cable with PoE Switch is required)

Jumper	Label	Function (If jumpered)
JP102	INC	Connects INC0 & INC1 to +5V and INC0B & INC1B to GND
JP102	OUTC	Connects OP1A to GND and OP1B to +5V

RIO-47142

Jumper	Label	Function (If jumpered)
J6	MRST	Master Reset enable. Returns RIO to factory default settings and erases non-volatile memory. Requires power-on or RESET to be activated.
	UPGD	Used to upgrade controller if the unit becomes unresponsive.
	19.2	Set baud Rate to 19.2k (default without jumper is 115k)
	OPT	10BaseT Ethernet Communication

RIO-472xx

Jumper	Label	Function (If jumpered)
JP5	MRST	Master Reset enable. Returns RIO to factory default settings and erases non-volatile memory. Requires power-on or RESET to be activated.
	UPGD	Used to upgrade controller if the unit becomes unresponsive.
	19.2	Set baud Rate to 19.2k (default without jumper is 115k)
	OPT	10BaseT Ethernet Communication

Jumper	Label	Function (If jumpered)
JP6	AUX (4 jumpers)	Power for board comes from external power source, see Step 2. Connect Power to the RIO and Power Requirements for EXT/AUX Power Option.
	PoE (4 jumpers)	Power for board comes from Power over Ethernet (No power cable is necessary – Ethernet cable with PoE Switch is required).

Jumper	Label	Function (If jumpered)
JP3	INC	Connects INC0 & INC1 to +5V and INC0B & INC1B to GND

RIO-47300

Jumper	Label	Function (If jumpered)
JP5	MRST	Master Reset enable. Returns RIO to factory default settings and erases non-volatile memory. Requires power-on or RESET to be activated.
	UPGD	Used to upgrade controller if the unit becomes unresponsive.
	19.2	Set baud Rate to 19.2k (default without jumper is 115k)
	OPT	10BaseT Ethernet Communication

Jumper	Label	Function (If jumpered)

JP13	INC0A	Connects INC0A +5V and INC0B to GND
	INC0B	
JP14	INC1A	Connects INC1A +5V and INC1B to GND
	INC1B	
JP15	INC2A	Connects INC2A +5V and INC2B to GND
	INC2B	

RIO Dimensions

RIO-4710x & RIO-4712x

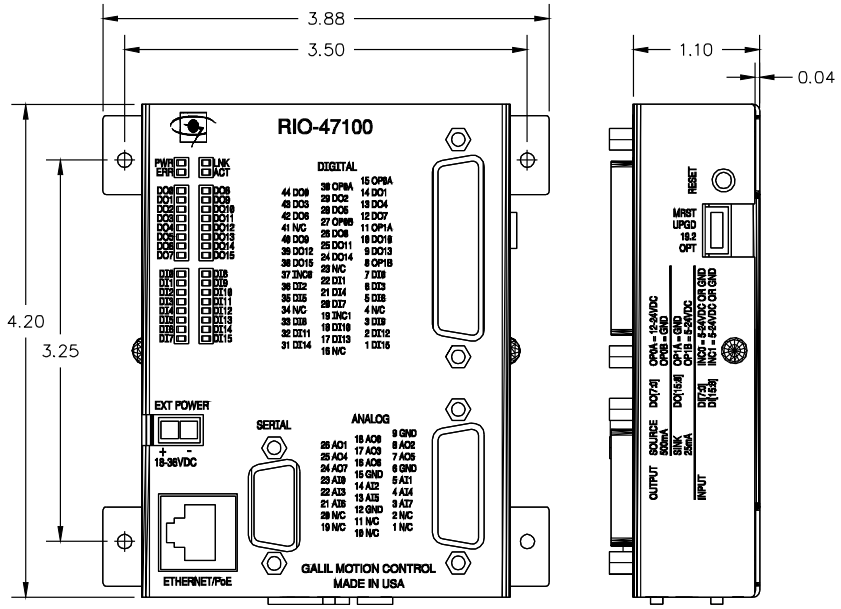


Figure A.15: Dimensions for RIO-471xx (in inches)

RIO-47142

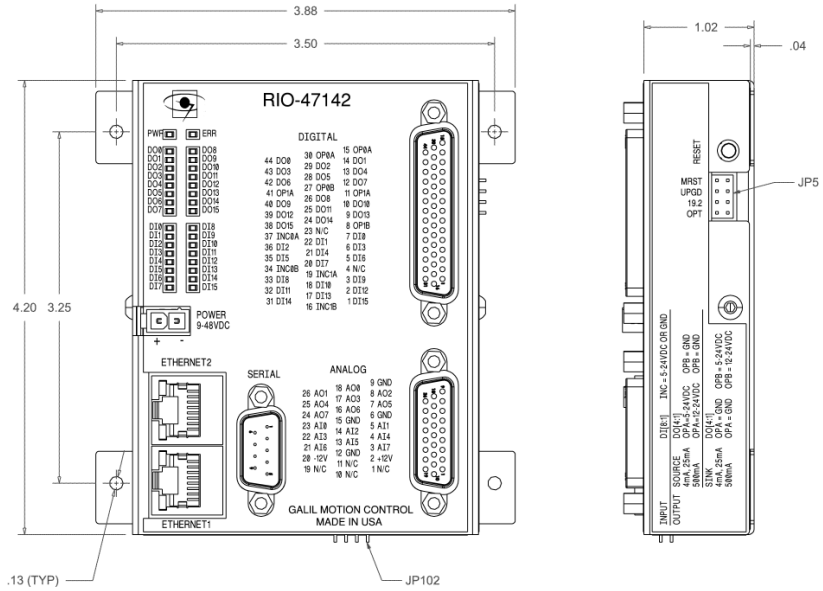


Figure A.16: Dimensions of the RIO-47142 (in inches)

RIO-472xx

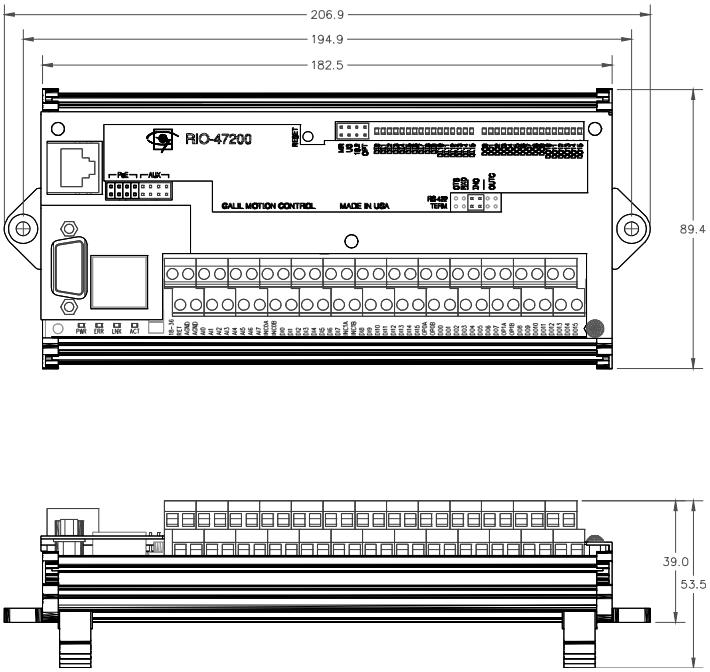


Figure A.17: Dimensions for RIO-472xx (in mm)

RIO-47300

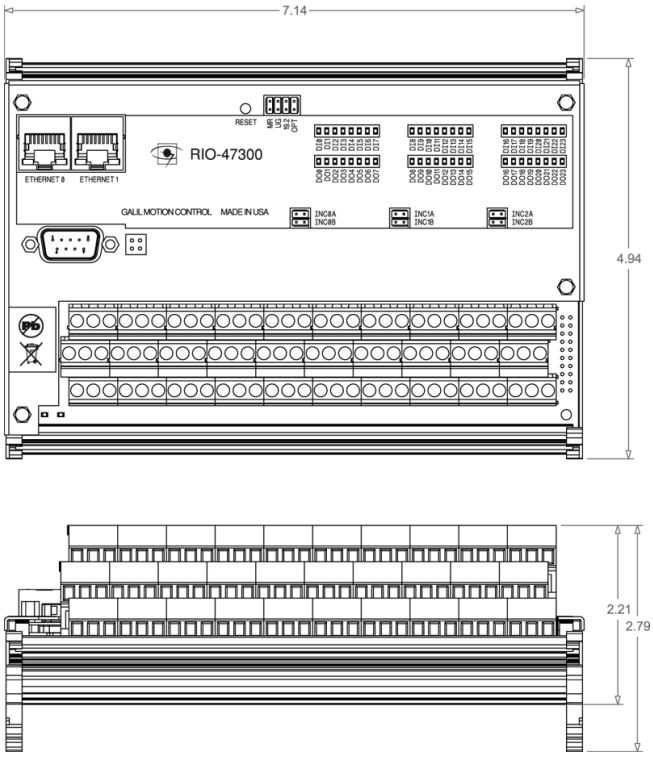


Figure A.18: Dimensions of RIO-47300 (in inches)

Accessories

Product	Description
PS-2.50-24	Low power switching power supply that comes with a 2-pin Molex connector to allow for mating directly to the RIO. See specifications here: A3 - Power Supplies, pg 116.
ICS-48026-M	26-pin D high-density male to screw terminals. Use 1 for each RIO-471x0 to break out analog signals
ICS-48044-M	44-pin D high-density male to screw terminals. Use 1 for each RIO-471x0 to break out analog signals
SCB-48206	26-pin D high-density Signal Conditioning Board interfaces to up to six RTDs (Resistive Temperature Device). See A1 – SCB-48206 for details.
SCB-48306-KTYPE	26-pin D high-density Signal Conditioning Board provides interface for up to six K-type thermocouples with screw-terminal type connectors. SeeA2 – SCB-48306/48316 for details.
SCB-48316-KTYPE	26-pin D high-density Signal Conditioning Board provides interface for up to six K-type thermocouples with thermocouple mating-type connectors. SeeA2 – SCB-48306/48316 for details.
CABLE-44M-1M	44-pin D high-density male cable to discrete wires. Use 1 for each RIO-471x0 to break out analog signals -1M = 1 meter length. Order -2M for 2 meter length
CABLE-26M-1M	26-pin D high-density male cable to discrete wires. Use 1 for each RIO-471x0 to break out analog signals

List of Other Publications

"Step by Step Design of Motion Control Systems"

by Dr. Jacob Tal

"Motion Control Applications"

by Dr. Jacob Tal

"Motion Control by Microprocessors"

by Dr. Jacob Tal

Contacting Us

Galil Motion Control
270 Technology Way
Rocklin, CA 95765

Phone: 916-626-0101
Fax: 916-626-0102
E-Mail Address: support@galilmc.com
URL: www.galil.com

Training Seminars

Galil, a leader in motion control with over 500,000 controllers working worldwide, has a proud reputation for anticipating and setting the trends in motion control. Galil understands your need to keep abreast with these trends in order to remain resourceful and competitive. Through a series of seminars and workshops held over the past 15 years, Galil has actively shared their market insights in a no-nonsense way for a world of engineers on the move. In fact, over 10,000 engineers have attended Galil seminars. The tradition continues with three different seminar, each designed for your particular skill set--from beginner to the most advanced.

MOTION CONTROL MADE EASY

WHO SHOULD ATTEND

Those who need a basic introduction or refresher on how to successfully implement servo motion control systems.

TIME: 4 hours (8:30 am-12:30 pm)

ADVANCED MOTION CONTROL

WHO SHOULD ATTEND

Those who consider themselves a "servo specialist" and require an in-depth knowledge of motion control systems to ensure outstanding controller performance. Also, prior completion of "Motion Control Made Easy" or equivalent is required. Analysis and design tools as well as several design examples will be provided.

TIME: 8 hours (8:00 am-5:00 pm)

PRODUCT WORKSHOP

WHO SHOULD ATTEND

Current users of Galil motion controllers. Conducted at Galil's headquarters in Rocklin, CA, students will gain detailed understanding about connecting systems elements, system tuning and motion programming. This is a "hands-on" seminar and students can test their application on actual hardware and review it with Galil specialists.

TIME: Two days (8:30 am-5:00 pm)

WARRANTY

All products manufactured by Galil Motion Control are warranted against defects in materials and workmanship. The warranty period for all products is 18 months except for motors and power supplies which have a 1 year warranty.

In the event of any defects in materials or workmanship, Galil Motion Control will, at its sole option, repair or replace the defective product covered by this warranty without charge. To obtain warranty service, the defective product must be returned within 30 days of the expiration of the applicable warranty period to Galil Motion Control, properly packaged and with transportation and insurance prepaid. We will reship at our expense only to destinations in the United States.

Any defect in materials or workmanship determined by Galil Motion Control to be attributable to customer alteration, modification, negligence or misuse is not covered by this warranty.

EXCEPT AS SET FORTH ABOVE, GALIL MOTION CONTROL WILL MAKE NO WARRANTIES EITHER EXPRESSED OR IMPLIED, WITH RESPECT TO SUCH PRODUCTS, AND SHALL NOT BE LIABLE OR RESPONSIBLE FOR ANY INCIDENTAL OR CONSEQUENTIAL DAMAGES.

A1 – SCB-48206

Description

The SCB-48206 Signal Conditioning Board interfaces to up to six 3-wire RTD's (Resistive Temperature Device). The SCB-48206 is designed to work with the RIO-4712x or RIO-47142.

The SCB-48206 plugs directly into the Analog 26-pin high-density D-sub connector and will use Analog Inputs 0-5 on the RIO for the 6 RTD inputs. (RTD[0:5] = AI[0:5]). [1] It is oriented vertically from the RIO connector as shown in Figure A1.1. Other mounting options are available upon request.

Figure A1.1: RIO-47122 with SCB-48206

1 Analog inputs 0-5 will not be available for general use analog inputs when the SCB-48026 is connected to the RIO.

Specifications

Number of Inputs	6 RTD inputs
RTD input – Analog Input Map	RTD[0:5] = AI[0:5]
Output Range	0-5V
Excitation Current	1 mA
Input Range	18 – 230 Ω[1]
Temperature Range (100 Ω RTD)	-200 to 350 deg C[1]

1 If greater than 230Ω (350 deg C) is required, contact Galil.

Wiring

The SBC-48206 has qty 6, 3-wire RTD inputs. The RTD is wired directly to the screw terminals as indicated in Figure A1.2 below.

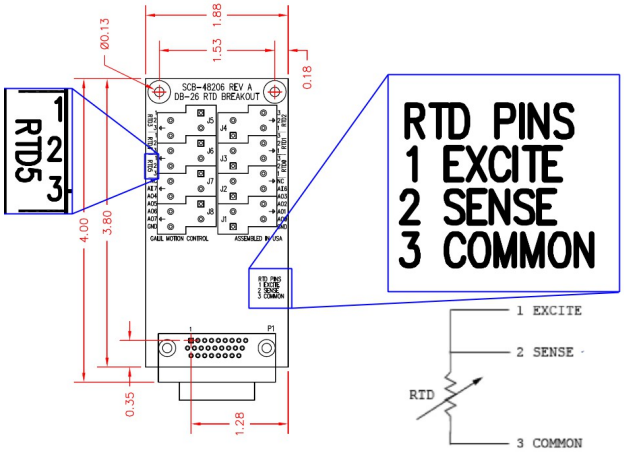


Figure A1.2: RTD wiring to SBC-48206

Dimensions

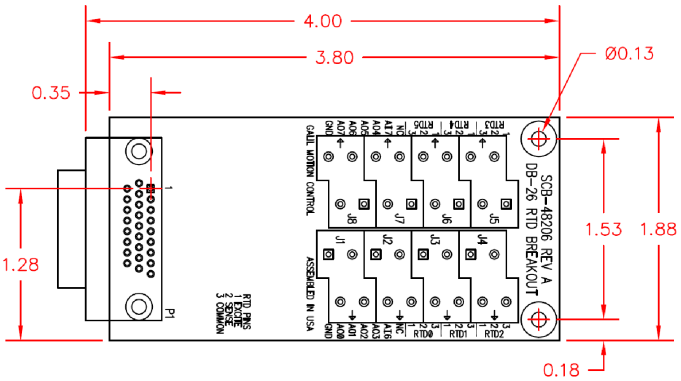


Figure A1.3: Dimensions for SCB-48206

Operation

The SBC-48206 will send a 0-5V analog voltage to the RIO that is related to the resistance of the RTD. When using the SBC-48026, the analog inputs should be set to 0-5V inputs for the 6 RTD inputs. This is done with the AQ command with a setting of 3 (AQ n,3 – where n = 0-5).

The calculation for the resistance of the RTD from the analog voltage is given from the following equation.

R = (1000*V)/21

Where R = Resistance of RTD

V = Analog Read from RIO

There are 2 methods for calculating the temperature once the resistance of the RTD has been calculated.

Note: The following calculations assume an RTD with R_0 = 100 Ω and α = 0.00385 (Platinum RTD).

Method 1

This method strictly uses the RTD coefficient and assumes a proportional relationship between impedance and temperature. The equation for this is given in the following equation.

$Tc = (R - R_0) / (\alpha * 100)$

Where Tc = Temperature in deg C

R_0 = 100 Ω

α = 0.00385

Below is an example program for using Method 1 that could run on the RIO-4712x or RIO-47142.

```
#MAIN
REM set Analog inputs 0-5 to 0-5V inputs
AQ 0,3
AQ 1,3
AQ 2,3
```

```
AQ 3,3
AQ 4,3
AQ 5,3
AT0;'set initial time reference
#Calc
REM calculate resistance of RTD
r0 = (1000*@AN[0])/21
r1 = (1000*@AN[1])/21
r2 = (1000*@AN[2])/21
r3 = (1000*@AN[3])/21
r4 = (1000*@AN[4])/21
r5 = (1000*@AN[5])/21
REM calculate deg C
Tc0 = (r0-100)/0.385
Tc1 = (r1-100)/0.385
Tc2 = (r2-100)/0.385
Tc3 = (r3-100)/0.385
Tc4 = (r4-100)/ 0.385
Tc5 = (r5-100)/0.385
REM calculate deg F (not required)
Tf0 = ((9*Tc0)/5)+32
Tf1 = ((9*Tc1)/5)+32
Tf2 = ((9*Tc2)/5)+32
Tf3 = ((9*Tc3)/5)+32
Tf4 = ((9*Tc4)/5)+32
Tf5 = ((9*Tc5)/5)+32
AT-100;'wait 100 ms from last time reference
JP#Calc
```

This method provides a relatively accurate temperature reading with a simple and straight-forward calculation. A limitation with this method is that it uses an idealized relationship between the impedance of an RTD and the temperature of the RTD. In reality, the relationship between impedance and temperature is not linear, so if higher precision is required from the temperature reading, the following Method should be used.

Method 2

This method uses the following equations to calculate the temperature of the RTD. These equations more accurately describe the relationship between temperature and impedance of the RTD than Method 1.

For Tc > 0 deg C (R(t)>100) $R(t) = R_0 (1 + A* Tc + B * Tc^2)$

For Tc < 0 deg C (R(t)<100) $R(t) = R_0 (1 + A*Tc + B * Tc^2 + C (Tc-100)*Tc^3)$

Where R(t) = Resistance of RTD

$R_0 = 100\ \Omega$

$A = 3.9083 * 10^{-3} * \text{deg C}^{-1}$

$B = -5.775 * 10^{-7} * \text{deg C}^{-2}$

$C = -4.183 * 10^{-12} * \text{deg C}^{-4}$

Below is an example program for using Method 2 that could run on the RIO-4712x or RIO-47142.

Note: The coefficients have been modified to avoid round off errors in the calculations in the temperature readings.

```
#MAIN
REM set Analog inputs 0-5 to 0-5V inputs
AQ 0,3
AQ 1,3
```

```
AQ 2,3
AQ 3,3
AQ 4,3
AQ 5,3
AT0;'set initial time reference
#Calc
REM calculate resistance of RTD
r0 = (1000*@AN[0])/21
r1 = (1000*@AN[1])/21
r2 = (1000*@AN[2])/21
r3 = (1000*@AN[3])/21
r4 = (1000*@AN[4])/21
r5 = (1000*@AN[5])/21
REM calculate deg C
r=r0;JS#Celcius;Tc0 = Tc
r=r1;JS#Celcius;Tc1 = Tc
r=r2;JS#Celcius;Tc2 = Tc
r=r3;JS#Celcius;Tc3 = Tc
r=r4;JS#Celcius;Tc4 = Tc
r=r5;JS#Celcius;Tc5 = Tc
AT-100;'wait 100 ms from last time ref
JP#Calc

#Celcius
sqrt=@SQR[992137.445376*(761.2471-r)]
Tc = (-25613.43488+sqrt)/(-7.569408)
REM adjust for Tc < 0 deg C
IF (Tc < 0)
Ta=-(((Tc-100)*Tc*Tc)/239062873.536)*Tc
Ta = Ta * 0.2311
Tc = Tc - Ta
ENDIF
EN
```

A2 – SCB-48306/48316

Description

The SCB-48306 and the SCB-48316 Signal Conditioning Board interface to up to 6 thermocouples. The SCB-483x6 boards are designed to work with the RIO-4712x or RIO-47142. The SCB-48316 provides thermocouple terminal connectors for the 6 thermocouple inputs, the SCB-48306 provides screw terminals inputs for the 6 thermocouple inputs. Both SCB boards provide screw terminal connections for Analog inputs 6 and 7 (AI6:7), all 8 analog outputs (AO0:7) and two GND terminals.

The SCB-48306 can plug directly into the Analog 26-pin high-density D-sub connector and will use Analog inputs 0-5 on the RIO for the 6 thermocouple inputs. (TC[0:5] = AI[0:5]).[1] It is oriented vertically from the RIO connector as shown in Figure A2.1. Other mounting options are available upon request.

By default the SCB-483x6 will be setup for type K thermocouple inputs. Types E, J and T are also available. The thermocouples interfacing to the SCB-483x6 must have an Ungrounded or Exposed Junction (aka Floating Junction); contact Galil if Grounded Junction thermocouples are required.

Figure A2.1: SCB-48306 on RIO-47120

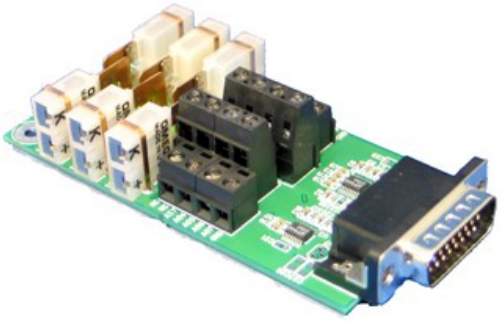


Figure A2.2: SCB-48316

1 Analog inputs 0-5 will not be available for general use analog inputs when the SCB-483x6 is connected to the RIO.

Specifications

Number of Inputs	6 Thermocouple Inputs	
Thermocouple input – Analog Input Map	TC[0:5] = AI[0:5]	
Range[1]	Type K (default)	0 – 345 deg C
	Type E	0 – 230 deg C
	Type J	0 – 270 deg C
	Type T	0 – 345 deg C
Voltage Constant [2]	Type K (default)	10.15 mV/deg C
	Type E	15.225 mV/deg C
	Type J	12.925 mV/deg C
	Type T	10.15 mV/deg C

1 Contact Galil if required temperatures are outside of listed ranges.

2 Voltage Constant will change if Range is modified

Wiring

The SCB-483x6 has qty 6 thermocouple inputs. The thermocouples interfacing to the SCB-483x6 must have an Ungrounded or Exposed Junction; contact Galil if Grounded Junction (Figure A2.4) thermocouples are required. The wiring of the thermocouple to the SCB-483x6 is shown in Figure A2.3 below.

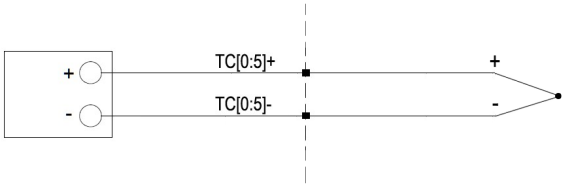


Figure A2.3: Thermocouple Wiring to SCB-483x6

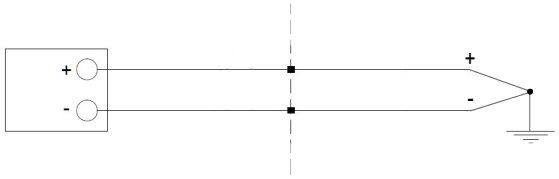


Figure A2.4: Grounded Thermocouple Input - Not supported with SCB-483x6

Operation

The SCB-483x6 will send an analog voltage to the RIO-4712x or RIO-47142 that is proportional to the temperature of the junction by the Voltage constant defined in the Specifications section. When using the SCB-483x6, the analog inputs should be set to 0-5V inputs for the thermocouple inputs. This is done with the AQ command with a setting of 3 (AQ n,3 – where n=0-5 for TC[0:5]).

The temperature can be determined by using the Voltage constants given in the Specifications section. The equation for calculating Temperature in deg C is:

Temperature (deg C) = (@AN[0:5] * 1000)/Voltage Constant

Where @AN[0:5] Analog input readings for TC[0:5]

Voltage Constant Voltage constant for SCB-483x6 and thermocouple type is defined in the Specifications section

The below code uses analog inputs 0-5 and stores the temperature into array Tc[0:5] – written for type K thermocouples.

```
#MAIN
REM Analog inputs 0-5 to 0-5V inputs
AQ 0,3
AQ 1,3
AQ 2,3
AQ 3,3
AQ 4,3
AQ 5,3
DM Tc[6]
voltK=10.15;'mV/deg C - type K
AT0;'set initial time reference
#Calc
n=0
#CalcH
Tc[n]=(@AN[n]*1000)/voltK
n=n+1
JP#CalcH,n<6
AT-100;'wait 100ms from last time ref
JP#Calc
```

A3 - Power Supplies

Galil offers a power supply that can be used to power the RIO product line, the PS-2.50-24. This low power switching mode supplies come with a 2-pin Molex Mini-Fit, Jr ™ connector to allow for mating directly to the RIO.

PS-2.50-24 Electrical Specifications

Power:	60 W Max
Voltage Output:	24 VDC
Max Current:	2.5 Amps
Input:	100-240 VAC, 50/60Hz
UL:	E183223 32 WK
CE Certified	

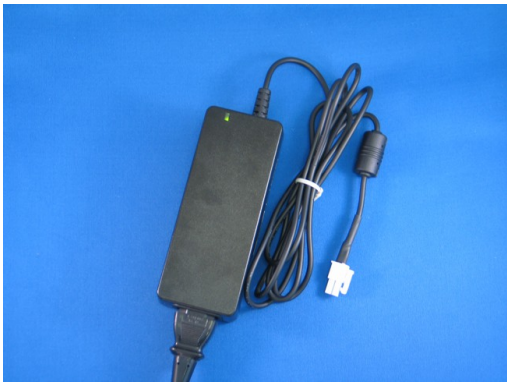


Figure 1: PS-0.25-24 Power Supply