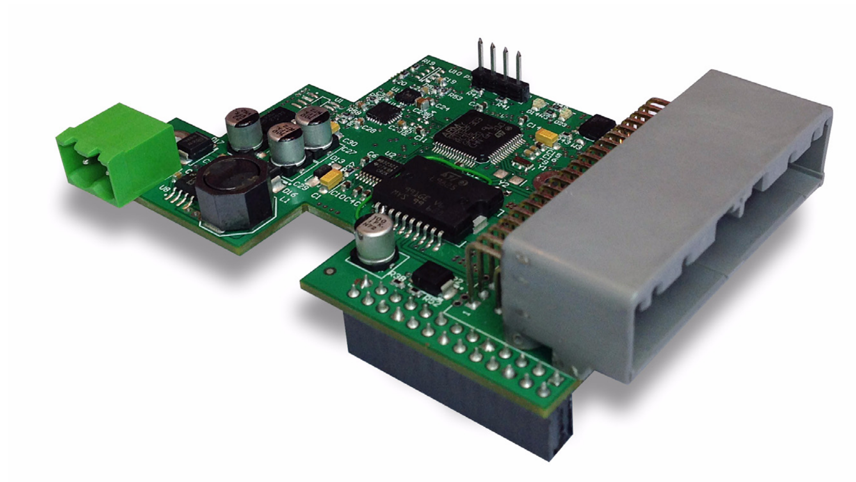
Preliminary - Subject to Change

Smart IO Expansion Card for Raspberry PI



Roboteq's RIO (Raspberry IO) is an intelligent I/O expansion card that is designed to turn Raspberry PI into a powerful and inexpensive embedded computer for robotics navigation, unmanned vehicles, machine control, industrial & home automation and any other applications that need interfacing to the real world.

RIO includes a 3A DC/DC converter that may be connected to a 8V to 30V DC supply, and generates the 5V needed by the PI and the RIO cards. The 5V is also brought to the IO connectors for powering external devices and sensors. The high input voltage range and high efficiency of the DC/DC converter allows the RIO/RPI stack to be used in a wide range of battery powered, and transformer powered applications

The card features 8 Digital Outputs capable of driving resistive loads such as lights, or inductive loads such as relay, solenoids or motors up to 1A each at 40V. 13 inputs (9 on AHRS version) are available and each can be individually configured as digital, 0-5V analog, or as pulse inputs. In the pulse mode, the inputs can capture pulse width, frequency, duty cycle, or quadrature encoder counts. Each input pin can also be configured as PWM output for driving RC servos.

The card also includes a 32-bit ARM microcontroller for processing and buffering the IO, and managing the communication with the processor on the RPI module. The processor can be configured to perform, on its own, a long list of conversion, capture, filtering, or conditioning on the IO so that the PI processor is relieved of these functions. A simple and powerful Basic-like programming language is built into the card and allows users to write programs that process the I/O in real time.

RIO provides a standard RS232 interface for communication to PCs and various peripherals. A TTL serial interface is provided for communication to microcomputers without RS232 drivers, such as Arduino. A CAN bus interface allows RIO to be a master or slave in a CAN bus network that can be up to 1000m long and contain up to 127 nodes. CAN bus also allows the simple interfacing to Roboteq motor controllers.

RIO comes with drivers and function library for configuring and accessing the I/O quickly and transparently: I/O appears as local memory to the Raspberry.

The card uses a 40-pin automotive-grade connector type MX34 (JAE) with locking mechanism to ensure easy insertion and removal while providing a most reliable connection.

RIO is available in a version with an AHRS (Attitude and Heading Reference System (RIO-AHRS card), including 3 axis accelerometer, 3 axis gyroscope and 3 axis magnetometer.

The RIO card firmware can easily be updated in the field to take advantage of new features as they become available.

Applications

- Robot navigation
- Unmanned air, land, sea and submarine vehicles
- Motion stabilization systems
- Automotive computers
- Industrial and Home automation
- Machine Control
- Movie/Stage props
- Automated Test Equipment

Features List

- 8V to 40V DCDC Converter. 5V 3A output for Raspberry and external accessories
- 13 Inputs (9 on AHRS version) configurable as Analog, Digital, or Pulse
- 0-5V Analog inputs range, 12-bit resolution
- Pulse width, Duty Cycle or Frequency pulse capture

- Pulse IO pins configurable as PWM for driving RC servos
- 8 Digital ouputs up to 1A each, 40V max
- Direct interfacing to Roboteq Magnetic Guide Sensor
- Standard RS232 port for interfacing to computers and other peripherals
- TTL Serial port for communication to microcontroller board without RS232 drivers (e.g. Arduino)
- CAN bus interface
- Status LED to indicate Raspberry to IO communication
- Optional AHRS with 3-axis Gyroscope, Accelerometer and Magnetometer
- Optional RS485 interface
- High reliability, high density 40-pin automotive-grade connector
- High performance on-board 32-bit ARM Cortex M3 processor with FPU
- Linux Libraries with functions for accessing the I/O
- Built-in Programming Language for local processing of I/O data in realtime. 50000 lines/s execution time
- Field upgradable firmware

Orderable Product References

TABLE 1.

Reference	**IO**	**Communication**	**AHRS**
RIO-IO	13 Ana/Digital/Pulse input	CAN, RS232, TTL serial	No
RIO-AHRS	11 Ana/Digital/Pulse input	CAN, RS232, TTL serial	Yes

Block diagram of RIO I/O Processor Card

The diagram below shows the RIO processor card.

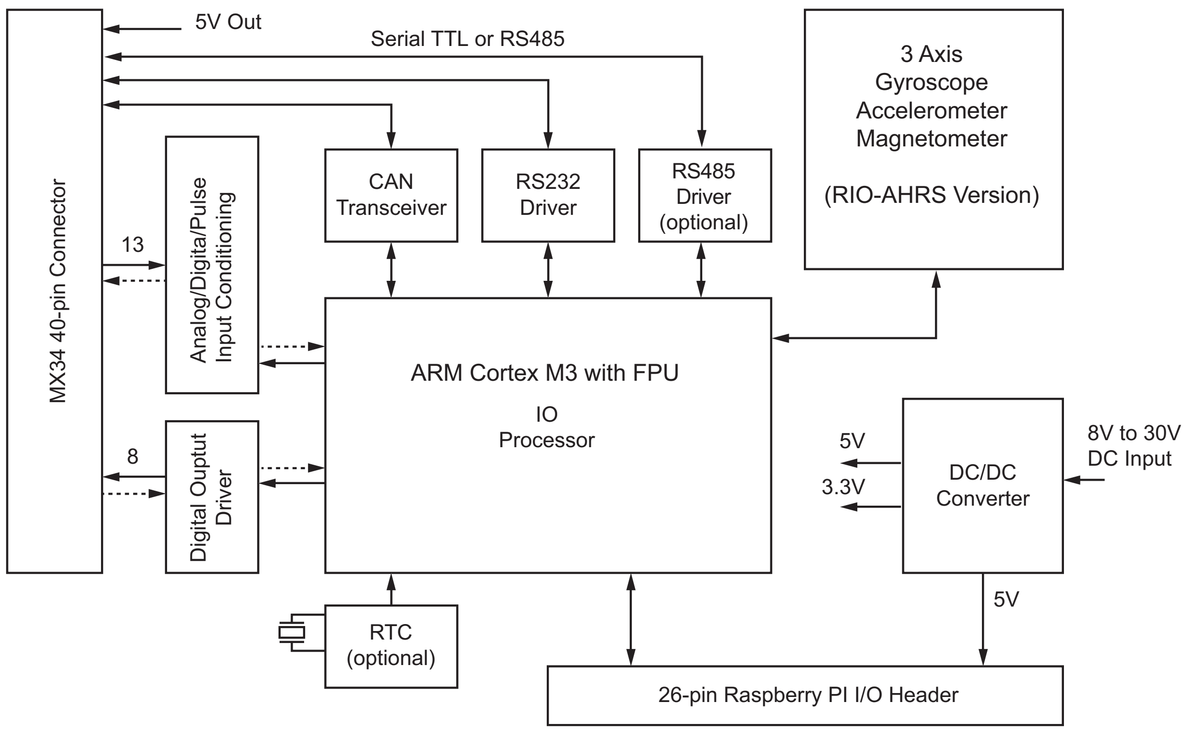


FIGURE 1. RIO I/O Processor Card

Board Connectors and LEDs

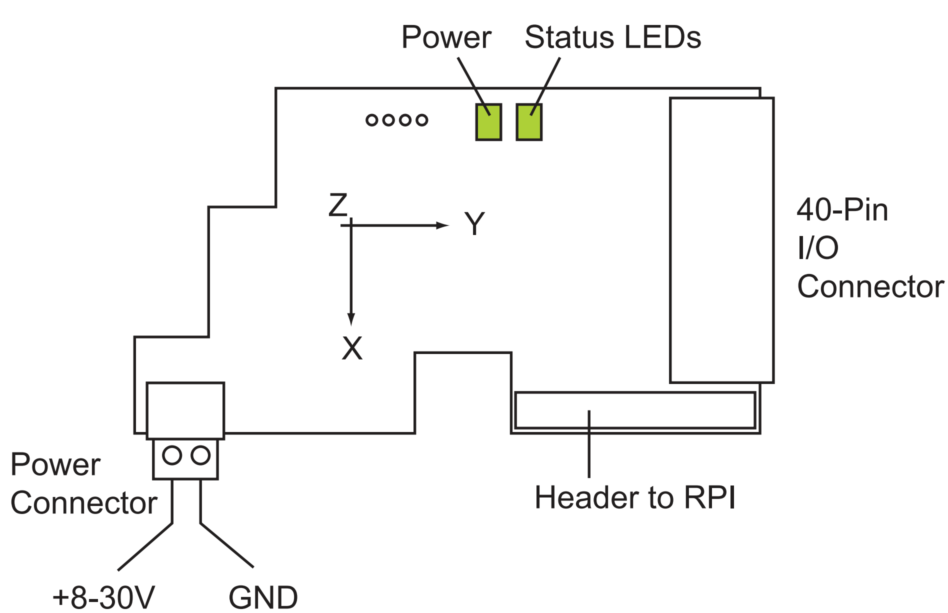


FIGURE 2. RIO board connectors and status LEDs

I/O Connector

The card uses a 40 automotive-grade connector type MX34 made by JAE Electronics. Contacts are rated up to 3A and can be fitted on AWG22-24 wires. This connector was selected for its superior reliability and high density, allowing the largest amount of connections in the small space available on the board. Pins are easy to attach at the end of wires and to insert into the connector shell. The cable/connector assembly can then conveniently be inserted/removed in and out of the RIO cards. A lock tab make the connector undetachable accidentally from the card.

User I/O Connector Pinout

Pin assignment is found in the table below.

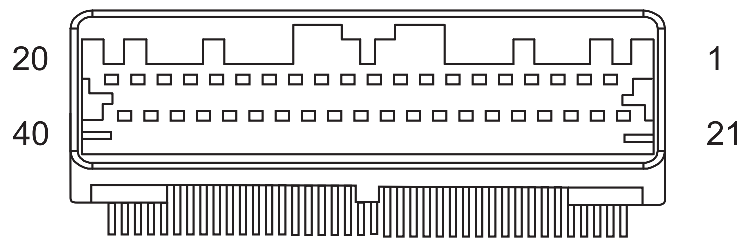


FIGURE 3. MX34 connector pin locations

TABLE 2.

Connector Pin	Signal	Type	Description
1	GND	Power	Power Ground
2	CANL	I/O	CAN Low
3	TTL-TxD	Output	Serial TTL Output Data (Optional RS485 Out)
4	RS232-TxD	Output	RS232 Output Data
5	GND	Power	Power Ground
6	VBackup	Power In	Backup supply for RTC
7	DIN13	Output	0-5V Dig/Ana/Pulse input 13
8	AOUT2	Output	12-bit 0-3.3V Analog Output 2
9	AOUT1	Output	12-bit 0-3.3V Analog Output 1
10	GND	Power	Power Ground
11	DOUT8	Output	1A Digital Output 8
12	DOUT7	Output	1A Digital Output 7
13	DOUT6	Output	1A Digital Output 6
14	DOUT5	Output	1A Digital Output 5
15	GND	Power	Power Ground
16	DOUT4	Output	1A Digital Output 4
17	DOUT3	Output	1A Digital Output 3
18	DOUT2	Output	1A Digital Output 2
19	DOUT1	Output	1A Digital Output 1
20	GND	Power	Power Ground
21	5VDC		5V Output
22	CANH	I/O	CAN High

TABLE 2.

Connector Pin	Signal	Type	Description
23	TTL-RxD	Input	Serial TTL Input Data (Optional RS485 In)
24	RS232-RxD	Input	RS232 Input Data
25	5VDC	Power Out	5V Output
26	ADIN12	Input	0-5V Dig/Ana/Pulse input 12
27	ADIN11	Input	0-5V Dig/Ana/Pulse input 11
28	ADIN10 (1)	Input	0-5V Dig/Ana/Pulse input 10 (1)
29	ADIN9 (1)	Input	0-5V Dig/Ana/Pulse input 9 (1)
30	5VDC	Power Out	5V Output
31	ADIN8	Input	0-5V Dig/Ana/Pulse input 8
32	ADIN7	Input	0-5V Dig/Ana/Pulse input 7
33	ADIN6	Input	0-5V Dig/Ana/Pulse input 6 - Encoder 3B
34	ADIN5	Input	0-5V Dig/Ana/Pulse input 5 - Encoder 3A
35	5VDC	Power Out	5V Output
36	ADIN4	Input	0-5V Dig/Ana/Pulse input 4 - Encoder 2B
37	ADIN3	Input	0-5V Dig/Ana/Pulse input 3 - Encoder 2A
38	ADIN2	Input	0-5V Dig/Ana/Pulse input 2 - Encoder 1B
39	ADIN1	Input	0-5V Dig/Ana/Pulse input 1 - Encoder 1A
40	5VDC	Power Out	5V Output
Note 1: Not available on AHRS version			

The connector is available from most major electronics parts distributors (Digikey, Newark, RS Components).

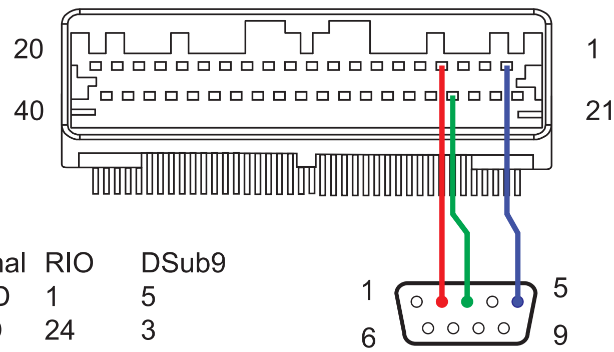
TABLE 3.

Description	Product Reference
Plastic housing	MX34040SF1
Contact pins	M34S75C4F2 or M34S75C4F4

Note: The pin number marking on the RIO PCB is incorrect. Use the pin numbering as shown in Table 2 on page 4. Pins numbers are also engraved on the plastic shell of the connector.

Serial Port Connections

A serial connection to a PC is necessary in order to configure the RIO card using a Graphical User Interface. A connection between the 40-pin connector and a DSub 9 pin connector must be done using 3 wires as shown in the figure below.



Signal	RIO	DSub9
GND	1	5
RxD	24	3
TxD	4	2

FIGURE 4. RS232 to MX34 connector wiring

Power Supply Connector

Power to the RIO Card is provided via a removable 2-pin connector with a standard profile and 5.08mm pin spacing. The figure below shows the power supply polarity. Applying power to this input will generate the 5V for the Raspberry and the RIO. Do not power the Raspberry through its USB connector when the RIO card is present.

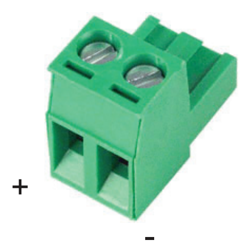


FIGURE 5. RIO power supply polarity

The connector is available from these and other suppliers.

TABLE 4.

Manufacturer	Product Reference
On Shore Technologies	EDZ950/2
Molex	0395300002
Phoenix Contact	1757019

Power and Status LEDs

RIO plugs on top of the Raspberry using the 26-pin header/connectors present on both cards. RIO will turn on, and the Power LED will lit, as soon as it is connected to a power supply. The card will also power the Raspberry via the 26-pin header.

A Status LED will flash to indicate that the onboard processor is running. The flash rate is 1Hz until the RIO card detects data exchange activity with the Raspberry, at which point the flashing rate will change to 4Hz. Data is constantly being exchanged using a Linux daemon running in the background (See "I/O Exchange Daemon" on page 12) and so the LED will flash as soon as the Raspberry has booted and for as long as the background communication task is active. The X,Y, Z on the Figure 2 on page 3 are the orientation for the Gyroscope, Accelerometer, and Magnetometer that are present on the AHRS version of the board.

I/O Connection

RIO has up to 13 inputs (11 on AHRS version) which can be used to capture digital, analog or pulse signals. Each input has a 53kOhm pull down resistance, and a pull up output buffer for driving RC servos.

Digital Input Connection

The diagrams below show how to connect switches to the input. A DC voltage can also be applied and a logic 1 will be detected above 2.5V.

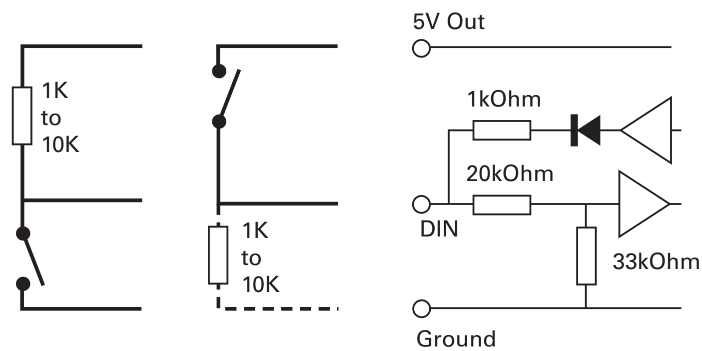


FIGURE 6. Pull down and Pull up switch wiring

Analog/Pulse Input Connection

DC, Pulse or AC voltage sources can be applied directly to the input. Analog captures must be configured in Absolute mode.

Potentiometers and other ratiometric sensors can use the 5V output available on the I/O connector. For best precision, capture must then be configured in Relative mode, to compensate fluctuations in the 5V output.

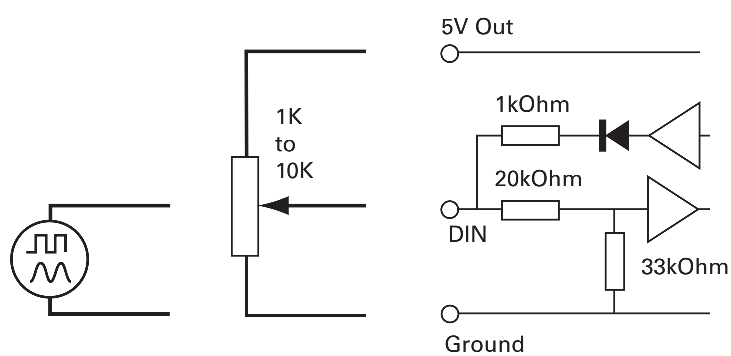


FIGURE 7. Analog/Pulse and Potentiometer wiring

RC Output Connection

All input pins can also be used to drive RC servos. The servos should be powered from an external voltage source and only be connected to the RIO's 5V output if they consume little power. When powered from the RIO, insert a 500uF/10V capacitor or higher. RIOs output drive capability is limited and not all servos may be compatible with the output signal. A pull down resistor at the pin may be required in some condition to make it work.

Note: RC Output mode is not yet implemented in version 1.0 of the firmware.

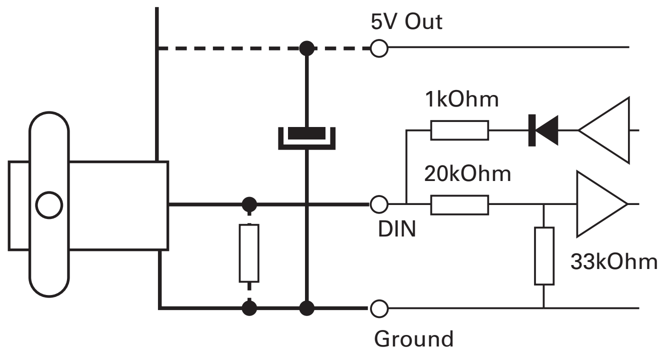


FIGURE 8. RC Servo wiring

Digital Output Connections

RIO's 8 Digital Ouputs are Open Drain MOSFET outputs capable of driving over 1A at up to 40V. Since the outputs are Open Drain, the output will be pulled to ground when activated. The load must therefore be connected to the output at one end and to a positive voltage source (e.g. a battery) at the other. Inductive loads must have a diode across the coil to prevent high voltage damage to the output driver.

Resistive and inductive loads can be connected simply as shown in the diagrams below.

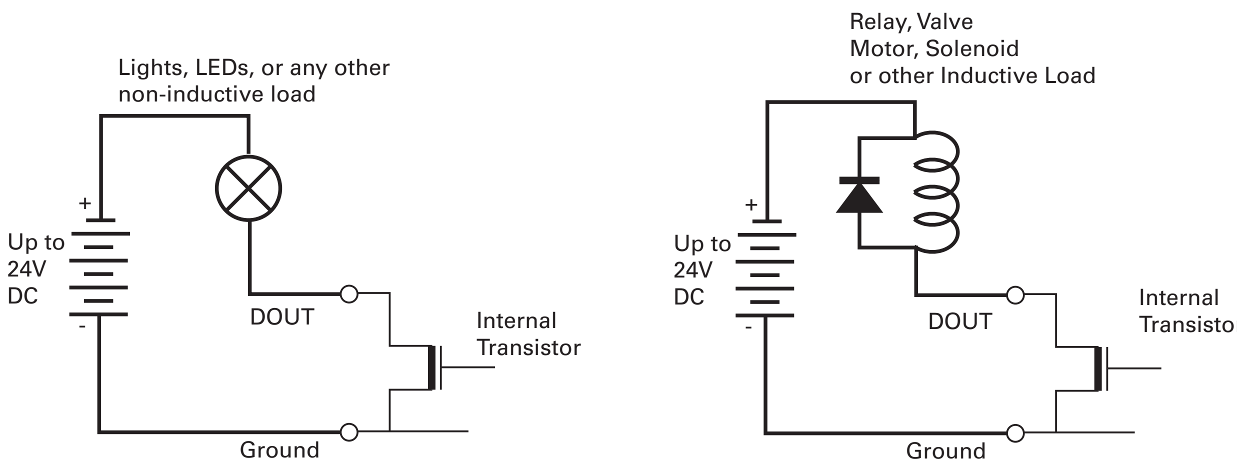


FIGURE 9. Connecting loads to DOUT pins

Electrical Specifications

Absolute Maximum Values

The values in the table below should never be exceeded, permanent damage to the card may result.

TABLE 5.

Parameter	Measure point	Min	Typ	Max	Units
Power Supply Voltage	Ground to Vin			35	Volts
Reverse Voltage on Battery Leads	Ground to Vin			25	Volts
Digital Output Voltage	Ground to Output pins			40	Volts
Analog and Digital Inputs Voltage	Ground to any signal pin on 40-pin connector			15	Volts
RS232 I/O pins Voltage	External voltage applied to Rx/Tx pins			15	Volts
Board Temperature	Board	-40		85	oC
Humidity	Board			100 (1)	%
Note 1: Non-condensing					

Power Stage Electrical Specifications

TABLE 6.

Parameter	Measure point	Min	Typ	Max	Units
Operating Voltage	Ground to Vin	8		30	Volts
Operating Current	Ground to Vin		50(1)(2)		mA
5V Output Voltage	Ground to 5V pins on 40-pin connector	4.6	4.75	4.9	Volts
5V Output Current	5V pins on 40-pin connector			1000 (3)	mA
Digital Output Voltage	Ground to Output pins			40	Volts
Digital Output Current	Output pins, sink current			1 (4)	Amps
Output On resistance	Output pin to ground		0.75	1.5	Ohm
Output Short circuit threshold	Output pin	1.05	1.4	1.75	Amps
Input Impedances	ADIN Input to Ground		53		kOhm
Digital Input 0 Level	Ground to Input pins	-1		1	Volts
Digital Input 1 Level	Ground to Input pins	3		15	Volts
Analog Input Range	Ground to Input pins	0		5.1	Volts
Analog Input Precision	Ground to Input pins		0.5		%
Analog Input Resolution	Ground to Input pins		1		mV
Analog Output Range	Ground to Input pins	0		3.3	Volts
Analog Input Resolution	Ground to Input pins			12	bits
Pulse durations	Pulse inputs	20000		10	us
Pulse repeat rate	Pulse inputs	50		250	Hz
Pulse Capture Resolution	Pulse inputs		1		us
Frequency Capture	Pulse inputs	100		10000	Hz
Encoder count	Internal	-2.147		2.147	10^9 Counts
Encoder frequency	Encoder input pins			1000	kHz

TABLE 6.

Parameter	Measure point	Min	Typ	Max	Units
Note 1: Excluding RPI and without load at 5V Outputs					
Note 2: At 12V supply. Current decreases as input voltage increases					
Note 3: Sum of all 5VOut outputs					
Note 4: Total average current on all outputs not to exceed 4.5A					

Operating & Timing Specifications

TABLE 7.

Parameter	Measure Point	Min	Typ	Max	Units
Command Latency	Command to output change	1	0.5	1	ms
IO Update rate	Internal		1000		Hz
RS232 baud rate	Rx & Tx pins		115200 (1)		Bits/s
Note 1: 115200, 8-bit, no parity, 1 stop bit, no flow control					

Scripting

TABLE 8.

Parameter	Measure Point	Min	Typ	Max	Units
Scripting Flash Memory	Internal		16384		Bytes
Max Basic Language programs	Internal	2000		3000	Lines
Integer Variables	Internal		1024		Words (1)
Boolean Variables	Internal		1024		Symbols
Execution Speed	Internal	50 000	100 000		Lines/s
Note 1: 32-bit words					

Linux Software Installation

Several software elements are necessary in order to use the RIO card:

- The **rioboard.d** daemon that ensures background data exchange between the RIO and the Raspberry
- Libraries of C calls to writing programs that access the I/O on the RIO
- The minicom terminal emulator program for configuring the RIO and performing firmware updates
- The most recent firmware file. It is highly recommended that you install the latest firmware
- A test program to immediately see the RIO in action
- Source code to all Linux software

All the element are installed using the simple procedure described below. This procedure applies to Raspberry with the Rasbian OS distribution. You must have “root” privileged and be logged in as root to perform the installation.

Installing the RioBoard Software

To install the RioBoard software follow the steps listed below.

- Login to RaspberryPi with SSH or open the console if you are using the Raspberry GUI. Use an account with “root” privileges. If no root password is set, use the following command to create one:
  `sudo passwd root`
- Change the current directory to the home directory of the user. To do so issue command
  `cd ~`
- Download the install script with the following command:
  `wget http://roboteq.com/riofiles/rio_board_install.sh`
- If you do not have the “wget” program install it with the following command first:
  `apt-get install wget`
- After you have successfully downloaded the install script, change the file attributes to make it executable:
  `chmod 755 rio_board_install.sh`
- Run the installation script by typing
  `./rio_board_install.sh`

After the installation is complete, the board will automatically reboot. You can easily verify that the installation was successful and that the RIO and RPI are communicating by looking at the flashing rate of the Status LED on RIO. It will be flashing at a rapid 4Hz rate when operational.

Installed Files Locations

Most of the files are loaded into the “**/rioboard**” directory. Shared libraries needed by RioBoard daemon are copied to “**/usr/local/lib**”. Daemon binary is copied in “**/usr/sbin**”. Daemon startup script is copied in “**/etc/init.d**”.

Testing RIO

A precompiled test program is installed by the script in **/rioboard**. From within this directory, make the test program executable with `chmod 755 rioboard_test`.

Run the program by typing `./rioboard_test`

The test program will display the values captured on the RIO input pins and the data from the Gyroscope, Accelerometer and Magnetometer (AHRS version).

The test program uses several functions from RioBoard library and is a good basis for writing your own custom application.

Writing and Building Applications

To use the RioBoard library in your applications, include the **RioBoard.h** header file in your program and set the linker to link against **librioboard.a**.

The "**/riofiles/raspio-src**" directory contains source files of the test program, the library and the makefile to build it. The source files of the daemon can also be found in that directory.

The list of all available library functions and their short description can be found in "**/riofiles/raspio-src/RioBoard.h**". Function parameters explain the usage of every individual function.

The only prerequisite for the library usage is that before using any functions the user application should call `RioInit()` and check the returned result.

Note that the library is not thread-safe and can't be used by several processes at the same time.

I/O Exchange Daemon

In a typical control and automation application, Inputs are read often and periodically and Outputs also need to be updated frequently. To handle I/O efficiently, the Raspberry can read the state of the inputs and change outputs on the RIO using a mirror memory architecture shown in the figure below.

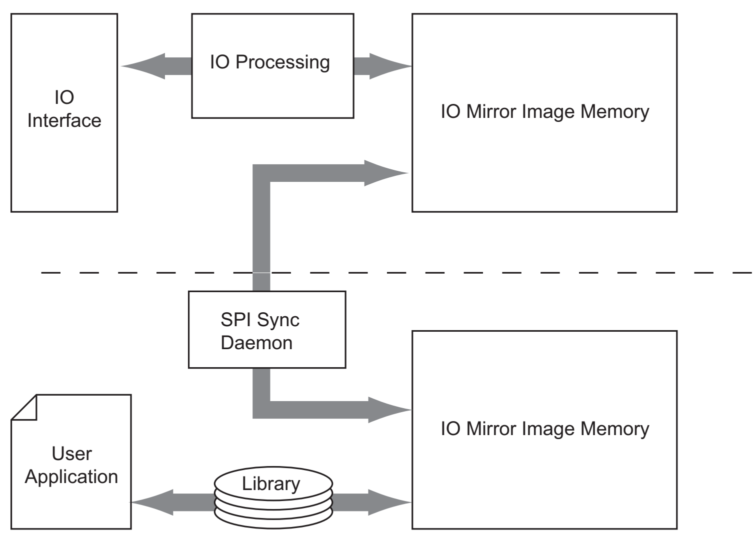


FIGURE 10. Mirror I/O Architecture

Because SPI bus is relatively slow, rather than actually writing and reading the IO directly on the RIO, Raspberry programs change the value in a section of the local memory. Likewise, the IO processor then updates the Output pins with the memory image data, and puts the state of the inputs in the memory.

A daemon running in the background then continuously synchronizes the content of the RPI memory to the mirror memory in the RIO so that both contain the same image. The two memory segments are synchronized every 1ms using Direct Memory Access, burdening very little both processors.

Using this technique, user programs are not slowed down for I/O access. However a delay of up to 2ms (1ms average) will be present from the time the program changes the state of an output, and the time the change is on the RIO output pin. Likewise, when reading the state of an input, the value that is returned by the function may be up to 2ms old (1ms average).

If some IO need lower latency, they can be processed directly on the RIO card using the programs written in MicroBasic.

The deamon verifies that the RIO card is present and functional. A function is provided for reading the result of that check. If an input is read while the RIO card is not detected by the daemon, the function will return the value 0. RIO also checks that the daemon periodically refreshes the mirror memory. The Status LED will flash at a fast 4Hz rate when this activity is detected. When the deamon is not running, the flashing rate drops to 1Hz.

Start/Stop the RioBoard Daemon

Normally, the daemon starts automatically at OS boot time. In case it is necessary to start/stop the daemon manually use the following commands:

```
service rioboard_d start

service rioboard_d stop
```

Serial Log Disabling

In the default Linux image, the Raspberry serial port is used by the OS allowing users to login over serial port. During system boot, the boot log is also sent to the serial port for monitoring.

Since the serial port is needed for many of the RIO functions (e.g. configuration and firmware updates), these serial port activities must be disabled. The install script takes care of this and you will not be able to use the serial port for login or boot logs observations.

Fetching the Latest RIO Firmware

The latest firmware release can be obtained from the Roboteq web site using the following shell command:

```
wget http://roboteq.com/riofiles/raspioF3-update.zip
```

Unzip the firmware with the following command:

```
unzip raspioF3-update.zip
```

A **.bin** file will be extracted in the current directory. This file can then be loaded in the RIO from the Raspberry or from a PC running the RIO PC Configuration Utility.

Update Firmware via the Raspberry

New firmware can be installed into the RIO using the following steps:

- Launch the minicom terminal emulator with the proper preconfigured settings by typing `riocom`
- Once minicom is running, change the settings for adding line feeds at the end of received strings by typing `Ctrl-a a`
- Issue the **?FID** query to verify that the RIO is present and communicating. RIO will reply with the firmware ID, date and RIO board type (AHRS or basic).

  Example: **?FID**
  FID=Roboteq v1.2 RIOAHRS 09/18/2013
- Issue the **?FUPD 321654987** query to enter the firmware update mode. RIO will display IAP followed by the C character, repeating every second. The repeating C indicates that RIO is awaiting the update file.
- Type `Ctrl-a s` to enter the file send mode and select ymodem. Navigate to the firmware file (e.g. **RaspioF3-update-090913.bin**).

The firmware will then be transferred to the RIO and run immediately upon transfer completion.

C Library Functions

The following functions are provided for reading and writing I/O data.

TABLE 9.

Function	Description
signed short RioReadPulseAnaInputs(unsigned char ch)	Read the value of an analog or pulse input
unsigned short RioReadAllDigitalInputs()	Read all digital inputs into a single 32bit value
bool RioReadDigitalInput(unsigned char ch)	Read the state of a single digital input channel
RioWriteAnaOutput(unsigned char ch, signed short value)	Write analog output channel
void RioWriteAllDigitalOutputs(unsigned long value)	Write all digital outputs with a single u32 word
void RioSetDigitalOutput(unsigned char ch)	Set a single output line
void RioResetDigitalOutput(unsigned char ch)	Clear a single digital output line
void RioWritePWMOutput (unsigned char val, unsigned short ch)	Set a PWM output channel with a value
signed short RioReadGyro(unsigned char ch)	Read the Gyroscope values on the X, Y or Z axis (ch = 0, 1, or 2)
signed short RioReadAccel(unsigned char ch)	Read the Accelerometer values on the X, Y or Z axis (ch = 0, 1, or 2)
signed short RioReadMag(unsigned char ch)	Read the Magnetometer values on the X, Y or Z axis (ch = 0, 1, or 2)
float GetAngle(unsigned char ch)	Read the AHRS Euler Angles output values on the Roll, Pitch or Yaw axis (ch = 0, 1, or 2)
unsigned char RioReadStatus(void)	Read the RIO Status flags
bool RioIsPresent(void)	Returns 1 if RIO is detected and operational

Serial Communication between RIO and Raspberry

Another communication mechanism using the RPI serial port is provided for the exchange of non periodic data, such as configuration parameters and strings. Since the Raspberry's serial port is a supported device by the Operating System, it can be readily used by any program that accesses the serial port, such as the minicom terminal emulator. User programs can be written using the standard library for opening and read/writing in the serial port.

The diagram below shows the various parts of the RIO that are accessible via the Raspberry's serial interface.

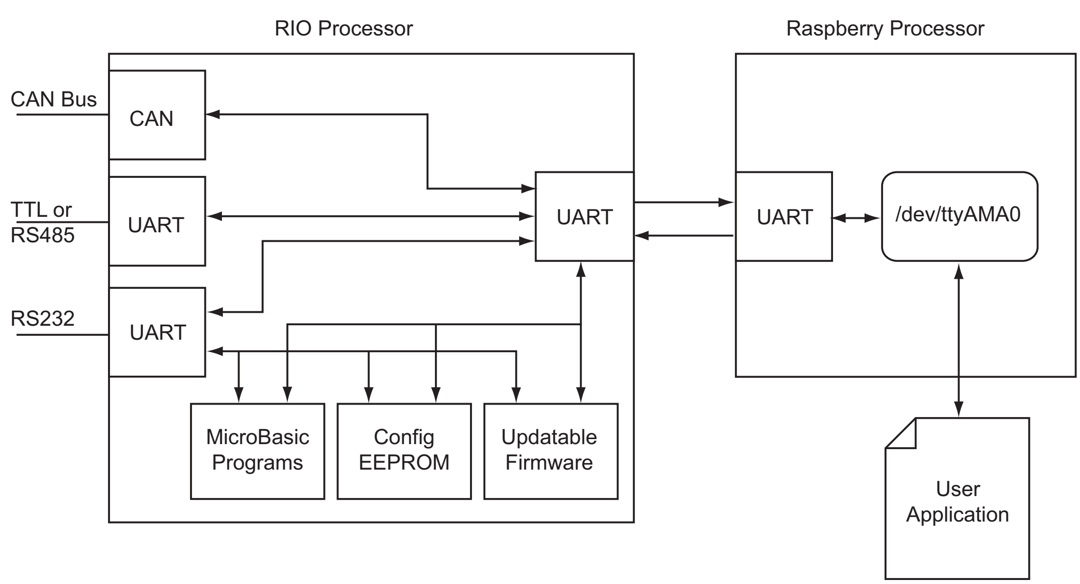


FIGURE 11. Serial Communication between RIO and Raspberry Pi

Reading/Setting Configuration and Accessing I/O

The serial communication can be used to read and change the many configuration and operating parameters of the RIO. It can also be used to read and write I/O data although the mirrored I/O memory provides a more efficient mechanism. A set of console commands are available for this purpose and described in the serial commands section of this document.

AHRS

The RIO-AHRS is equipped with a 3-axis Accelerometer, Gyroscope and Magnetometer. A special algorithm “fuses” the data of each of these sensors to give the board’s Attitude and Heading. The board’s Pitch, Roll and Yaw angles is given using Euler angles or Quaternions. Figure below shows the angles relative to the board hardware.

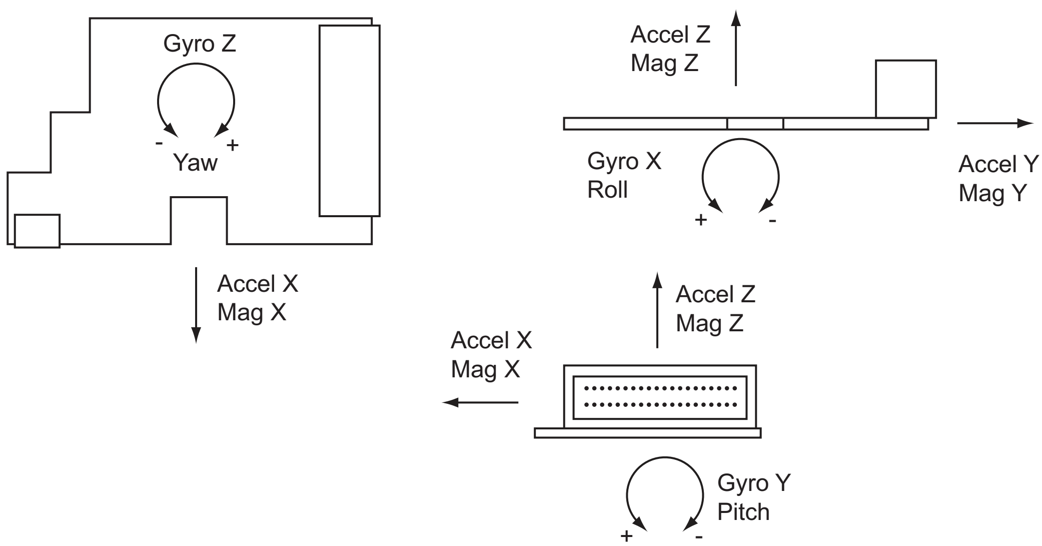


FIGURE 12. Sensors and AHRS references

The data of each individual sensor, and the AHRS output can be read by the Raspberry PI using library of functions for this purpose. The data can also be processed by the MCU on board RIO using the MicroBasic language.

Sensor Monitoring with PC Utility

The value of each sensor can be plotted in real time using the supplied PC utility. The utility’s chart recorder has six tracks, each of which can be assigned to a sensor output. Sensor data can be saved in a log for later analysis.

The AHRS Pitch, Roll and Yaw data is used to animate a 3D figure that, with proper calibration, will move on the screen the same way as the board is moved. The AHRS output angles can also be plotted and logged in the 2D chart.

Sensor Calibration

The sensor must be calibrated before using it for the first time. For the magnetometer in particular, the calibration must be done in the actual place where the board will be used, in order to take into account the possible distortion to the magnetic field induced that the surrounding hardware. Calibration is done using the PC utility in the **Setup** tab, in two steps.

The Gyroscope zero offset must first be calibrated. This is done by leaving the RIO on a surface (in any position) and make sure it does not move. Click then the **Gyroscope Calibration** button. The offset will be measured and stored into the RIO's EEPROM. You can verify that the calibration was successful by monitoring the Gyroscope data in the chart recorder. A small amount of noise (approx. +/- 20) will show in the chart instead of 100 or more for an uncalibrated gyroscope.

The AHRS calibration process requires that the sensor be rotated in every direction, in every angle, for approximately one minute. The rotation must be made while keeping the board in the same location in space, i.e. not moved laterally, up or down). The collected data is then analyzed to automatically determine a couple of dozen parameters needed by the AHRS fusion algorithm to work correctly. A button is provided on the setup screen of the PC utility. It must be pressed only after gyroscope calibration. Once pressed, the utility will prompt you on what to do next. At the end of the process, the new calibration values are stored in RIO's RAM and are effective immediately. You must click on **Save to Device** for the calibration to be stored in the RIO's EEPROM and then automatically reloaded every time the board is powered up.

Using the PC Utility

A powerful utility is available for download from Roboteq's web site for setting up, monitoring and performing maintenance functions. While RIO is delivered ready to use right off the box, it contains many parameters that can easily be changed using user-friendly menus. The utility can be used for testing and troubleshooting, for performing field updates of the RIO firmware and for editing and running scripts.

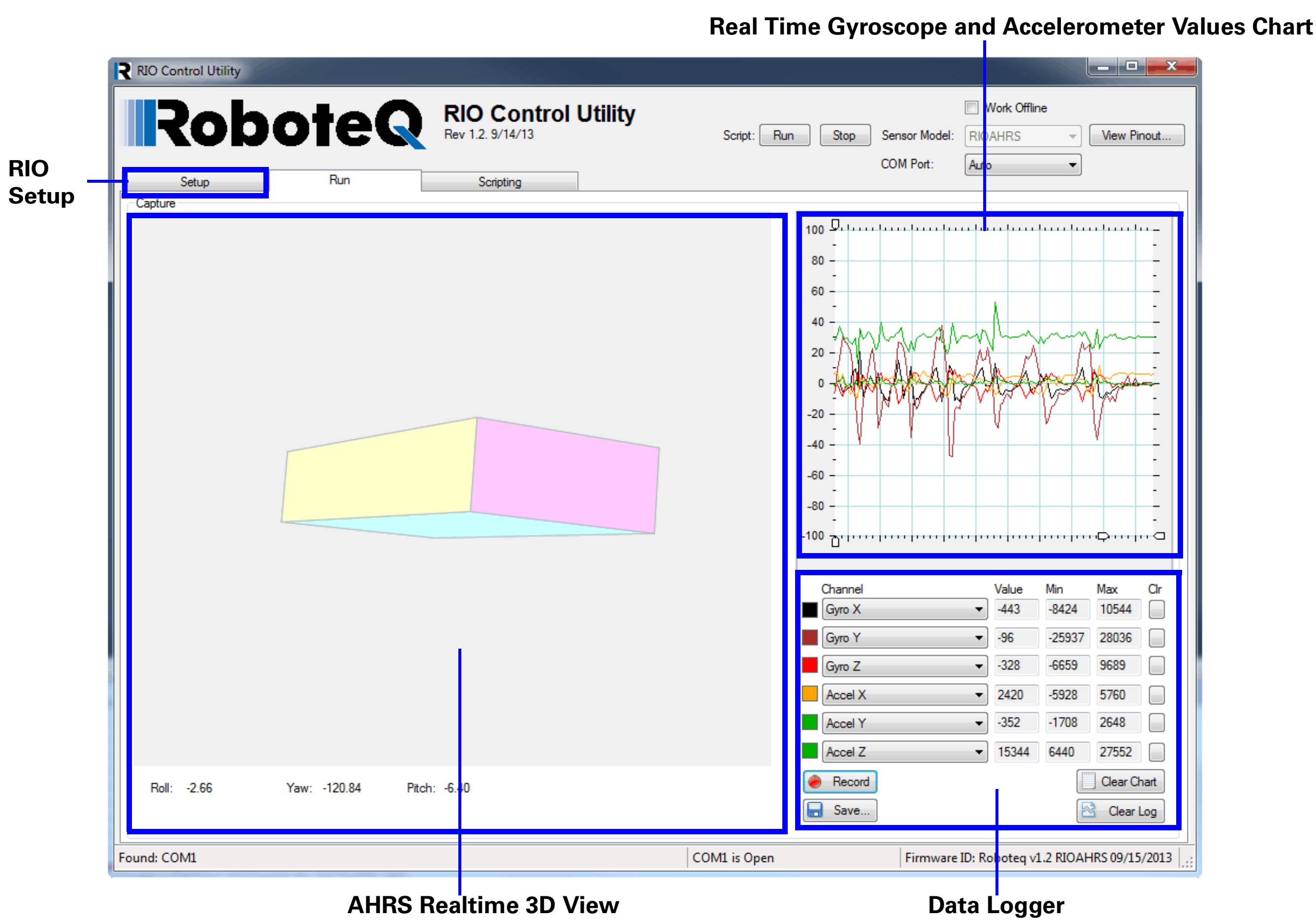


FIGURE 13. RIO PC Configuration Utility

MicroBasic Scripting

The RIO features the ability for the user to write programs that are permanently saved into, and run from the card's Flash Memory. This capability is the equivalent of combining the functionality of a PLC or Single Board Computer directly into the expansion card. The language is a very simple, yet powerful language that resembles Basic. Scripts can be simple or elaborate, and can be used for various purposes. Language description and reference can be found on the Roboteq web site.

Command Reference Summary

RIO accepts a number of commands via its serial port for reading operational data, sending commands, setting configuration, and performing maintenance. Commands can be sent from any port: Raspberry COM port, RS232, TTL Serial and are processed as they arrive.

Real Time Queries

These are commands for reading RIO data. They begin with the question mark character. Table 10 shows the list of supported queries.

Each time a query is executed, it is stored in a history buffer and may therefore be automatically repeated at a periodic rate using the **#** character with the following syntax:

repeat last query in queue
 # nn repeat last queries ever nn ms. Example: **# 100** to execute one query from the history queue every 100ms
 # C clear queue

TABLE 10.

Query	Arguments	Description
AI	[InputNbr]	Read Analog Inputs
AIC	[InputNbr]	Read Analog Inputs Converted
B	[VarNbr]	Read User Boolean Variable
C	[Channel]	Read Absolute Encoder Count
CAN	[Channel]	Read Raw CAN Message
CF	[Channel]	Read Raw CAN Received Frames Count
D	[InputNbr]	Read All Digital Inputs
DI	[InputNbr]	Read Individual Digital Inputs
DO	none	Read Current Digital Outputs
EO	[Channel]	Read AHRS Degrees Output
FID	none	Read Firmware ID String
MA	[Channel]	Read AHRS Q Out
MBB	[VarNumber]	Read Microbasic Integer Variable
MBV	[VarNumber]	Read Microbasic Boolean Variable
MG	[Channel]	Read MEMS Gyroscope
MGD	none	Read Magsensor Track Detect
MGM	[Channel]	Read Magsensor Markers
MGS	none	Read Magsensor Status
MGT	[Channel]	Read Magsensor Track Position
ML	[Channel]	Read all MEMS
MM	none	Read Magnetometer

TABLE 10.

Query	Arguments	Description
PI	[InputNbr]	Read Pulse Inputs
PIC	[InputNbr]	Read Pulse Inputs Converted
QO	[Channel]	Read AHRS Quaternion Output
RF	none	Read Raspberry Flags
RP	none	Read Raspberry Present
RTS	none	Read Raspberry Time Stamp
SVI	[VarNumber]	Read RIO to RPI User Variable
SVO	[VarNumber]	Read RPI to RIO User Variable
T	none	Read IC Temperatures
TM	[Channel]	Read Time
TRN	none	Read Power Unit Tree filename
V	none	Read 5V output
VAR	[VarNumber]	Read User Integer Variable

Real Time Commands

These are commands used to instruct RIO to do something. They begin with the exclamation mark character. Table 11 shows the list of supported commands

TABLE 11.

Command	Arguments	Description
AO	Value	
B	Value	Set User Boolean Variable
C	Channel Counter	Set Encoder Counters
CG	Channel Command	Set Motor Command via CAN
CS	Var number Value	Send Raw CAN frame
D0	BitNumber	Reset Individual Digital Out bits
D1	BitNumber	Set Individual Digital Out bits
DS	Value	Set all Digital Out bits
EES	None	Save Config to Flash
R	Option	Run MicroBasic Script
RC	Channel RCPulse	Set RC Pulse Out
SVI	Value	Set RIO to RPI User Variable
SVO	Value	Set RPI to RIO User Variable
VAR	Value	Set User Variable

Configuration Commands

These commands are used to read or modify the configuration parameters. They begin with the **~** character for reading and the **^** character for writing. Table 12 shows the list of supported configuration commands. However, it is easier and preferable to use the PC utility menus for inspecting and changing configurations. If changing manually, remember to save the new configuration to flash with the %EESAV. Otherwise, RIO will revert to the previously active configuration next time it is powered on.

TABLE 12.

Command	Argument	Default	Description
ACTR	InputNbr	2500	Analog Center
ADB	InputNbr	5%	Deadband
ALIN	InputNbr	Linear	Analog Linearity
AMAX	InputNbr	5000	Analog Max
AMIN	InputNbr	0	Analog Min
AMOD	InputNbr	Absolute	Analog Input Mode
AOS	none	ADInput	Source of Analog Output value
APOL	InputNbr	Direct	Analog Input Polarity
BRUN	none	Disabled	MicroBasic Auto Start
CBR	none	250Kbit/s	CAN Bit Rate
CEN	none	Disabled	CAN Mode Select
CHB	none	100ms	CAN HeartBeat
CLSN	none	0	CAN Listening COB
CNOD	none	1	CAN Node Address
CSRT	none	0	CAN SendRate
DINL	InputNbr	Direct	Digital Input Active Level
DOL	InputNbr	Direct	Digital Output Level
DOS	none	RPI	Digital Output Source
ECHOF	none	Echo On	Disable/Enabe RS232 & USB Echo
EE	Address	0	User EE Storage
EQS	none	Euler	Euler or Quaternion
INS	Channel	ADinput	Digital Input Source
PCTR	InputNbr	1500 (us)	Pulse Center
PDB	InputNbr	5%	5% Pulse Deadband
PLIN	InputNbr	Linear	Pulse Linearity
PMAX	InputNbr	2000 (us)	Pulse Max
PMIN	InputNbr	1000 (us)	Pulse Min
PMOD	InputNbr	RC Pulse	Pulse Input Mode
POS	none	RPI	Pulse Output Source
PPOL	InputNbr	Direct	Pulse Input Polarity
RSBR	none	115200	Set RS232 bit rate
SCRO	none	Last Used	Select Print output port for scripting

Maintenance Commands

These commands are used to perform maintenance functions. They begin with the **%** character. Table 13 shows the list of supported configuration commands.

TABLE 13.

Command	Arguments	Description
CLRST	Key 321654987	Load Factory default configs
CLSAV	Key 321654987	Calibration values to EEPROM
EELD		Load Parameters from EEPROM
EERST	Key 321654987	Restore Factory Defaults
EESAV		Save Parameters to EEPROM
FUPD	Key 321654987	Enter Firmware Update Mode
GREF		Sets Gyro 0 Ref
RESET		Reset Controller
RTEST		Enter SPI test mode
SLD		Load Minibasic scripts
STIME	Number of seconds since 1/1/2000	Set Time
YREF		Sets Yaw 0 Ref