



# **A/D Converter Card (RN1200)**

RabbitNet LAN Card

## **User's Manual**

019-0133 • 030731-A

# **RN1200 User's Manual**

Part Number 019-0133 • 030731-A • Printed in U.S.A.

©2003 Z-World Inc. • All rights reserved.

Z-World reserves the right to make changes and improvements to its products without providing notice.

## **Trademarks**

Rabbit and Rabbit 3000 are registered trademarks of Rabbit Semiconductor.

RabbitNet is a trademark of Z-World Inc.

Dynamic C is a registered trademark of Z-World Inc.

## **Z-World, Inc.**

2900 Spafford Street  
Davis, California 95616-6800  
USA

Telephone: (530) 757-3737  
Fax: (530) 753-5141

[www.zworld.com](http://www.zworld.com)

# TABLE OF CONTENTS

<b>Chapter 1. Overview</b>	<b>1</b>
1.1 A/D Converter Card Features .....	2
1.1.1 Software .....	2
1.1.2 Connectivity Tools .....	2
1.1.3 DIN Rail Mounting .....	3
1.2 Connecting Peripheral Cards .....	4
1.2.1 Power-Supply Connections .....	5
1.3 Pinout .....	6
1.3.1 Headers .....	6
1.3.2 Indicator LED .....	6
1.4 Analog Inputs .....	7
1.4.1 Analog Current Measurements .....	9
1.4.2 Calibrating the A/D Converter Chip .....	10
1.4.2.1 Modes .....	10
1.4.2.2 Calibration Constants .....	10
1.4.2.3 Calibration Recommendations .....	11
1.4.2.4 Factory Calibration .....	12
<b>Chapter 2. A/D Converter Card Software</b>	<b>13</b>
2.1 Dynamic C Libraries .....	13
2.1.1 Accessing and Downloading Dynamic C Libraries .....	14
2.2 Sample Programs .....	15
2.2.1 General RabbitNet Operation .....	15
2.2.2 A/D Converter Inputs .....	16
2.3 A/D Converter Card Function Calls .....	17
2.3.1 Analog Input Function Calls .....	17
2.3.2 Status Byte .....	29
<b>Appendix A. A/D Converter Card Specifications</b>	<b>31</b>
A.1 Electrical and Mechanical Specifications .....	31
A.1.1 Physical Mounting .....	33
A.2 Jumper Configurations .....	34
<b>Appendix B. RabbitNet</b>	<b>35</b>
B.1 General RabbitNet Description .....	35
B.2 Physical Implementation .....	37
B.2.1 Control and Routing .....	37
B.3 Function Calls .....	38
B.3.1 Status Byte .....	44
<b>Notice to Users</b>	<b>45</b>
<b>Index</b>	<b>47</b>
<b>Schematics</b>	<b>49</b>



# 1. OVERVIEW

Chapter 1 describes the features and the use of the A/D Converter Card, one of the peripheral I/O cards designed for use with the RabbitNet expansion ports on Z-World's Coyote (BL2500) and eDisplay (OP7200). The RabbitNet expansion ports enable a modular and expandable embedded control system whose configuration of I/O cards can be tailored to a large variety of demanding real-time control, display, and data-acquisition applications.

A typical RabbitNet™ system consists of a master single-board computer and one or more peripheral I/O cards. A high-performance Rabbit 3000® or Rabbit 2000® microprocessor on the master provides fast data processing, and the BL2500 master also provides the DCIN and +5 V power for the peripheral boards. Figure 1 shows a conceptual view of the A/D Converter Card connected to a master.

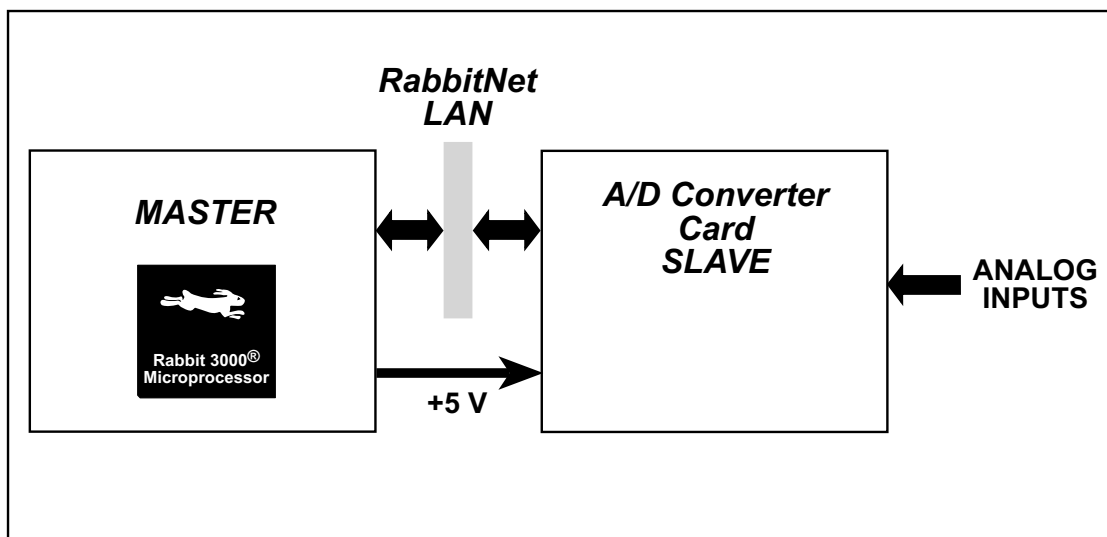


Figure 1. A/D Converter Card (Slave) Connected to Master

## 1.1 A/D Converter Card Features

- 8 single-ended 11-bit or 4 differential 12-bit analog inputs
- 4 channels can be set with jumpers to be 11-bit 4–20 mA analog inputs, all 8 channels can be special-ordered in quantity to be 11-bit 4–20 mA analog inputs
- 1 M $\Omega$  input impedance
- 2.5 ksamples/s sampling rate
- software-controlled voltage ranges: 0–1 V, 2 V, 5 V, 10 V, 20 V DC (single-ended) or  $\pm 1$  V,  $\pm 2$  V,  $\pm 5$  V,  $\pm 10$  V,  $\pm 20$  V DC (differential)
- can be mounted in standard 100 mm DIN rail trays sold by other suppliers
- interfaces with master through RabbitNet™ serial protocol at 1 Megabit per second using standard Ethernet cable up to 10 m (33 ft) long

### 1.1.1 Software

The A/D Converter Card is a slave; the master to which it is connected is programmed using version 8.01 or later of Z-World's Dynamic C. If you are using a BL2500 or an OP7200 as your master with an earlier version of Dynamic C, Z-World recommends that you upgrade your Dynamic C installation. Contact your authorized Z-World distributor or your Z-World Sales Representative at +1(530)757-3737 for more information on Dynamic C upgrades.

### 1.1.2 Connectivity Tools

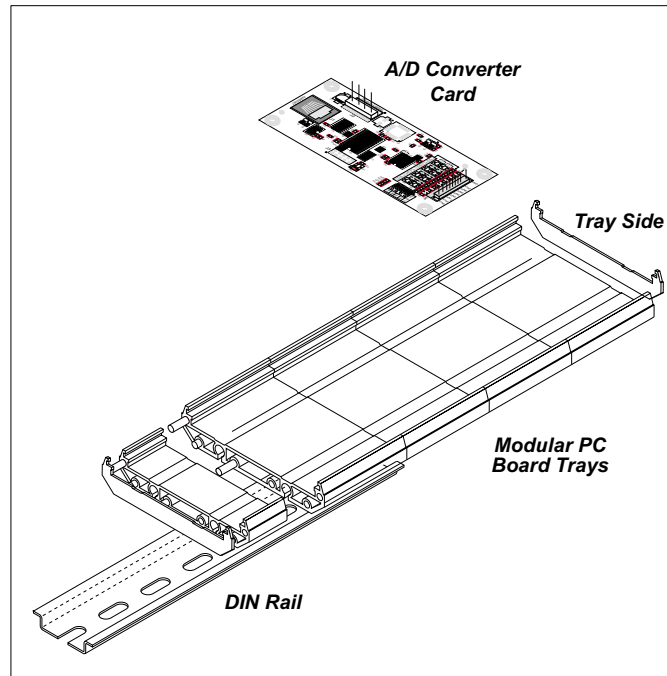
Z-World also has available additional tools and parts to allow you to make your own wiring assemblies to interface with the friction-lock connectors on the A/D Converter Card.

- Connectivity Kit (Z-World Part No. 101-0581)—Six 1  $\times$  10 friction-lock connectors (0.1" pitch) with sixty 0.1" crimp terminals; and two 1  $\times$  4 friction-lock connectors (0.156" pitch) and two 1  $\times$  2 friction-lock connectors (0.156" pitch) with fifteen 0.156" crimp terminals. Each kit contains sufficient parts to interface with at least two A/D Converter Cards.
- Crimp tool (Z-World Part No. 998-0013) to secure wire in crimp terminals.

Contact your authorized Z-World distributor or your Z-World Sales Representative at +1(530)757-3737 for more information.

### 1.1.3 DIN Rail Mounting

The A/D Converter Card may be mounted in 100 mm DIN rail trays as shown in Figure 2.



**Figure 2. Mounting A/D Converter Card in DIN Rail Trays**

DIN rail trays are typically mounted on DIN rails with “feet.” Table 1 lists Phoenix Contact part numbers for the DIN rail trays, rails, and feet. The tray side elements are used to keep the A/D Converter Card in place once it is inserted in a DIN rail tray, and the feet are used to mount the plastic tray on a DIN rail.

**Table 1. Phoenix Contact DIN Rail Mounting Components**

DIN Rail Mounting Component	Phoenix Contact Part Description	Phoenix Contact Part Number
Trays	UM 100-PROFIL cm*	19 59 87 4
Tray Side Elements	UM 108-SE	29 59 47 6
Foot Elements	UM 108-FE	29 59 46 3

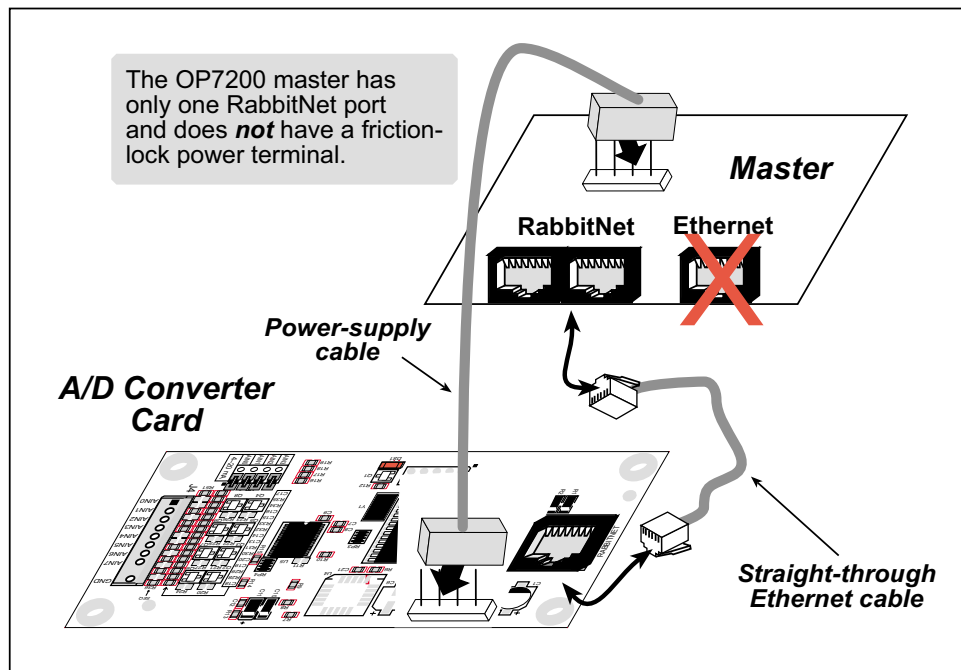
\* Length of DIN rail tray in cm

**NOTE:** Other major suppliers besides Phoenix Contact also offer DIN rail mounting hardware. Note that the width of the plastic tray should be 100 mm (3.95") since that is the width of the A/D Converter Cards. 108 mm plastic trays may be used with spacers.

## 1.2 Connecting Peripheral Cards

Use a straight-through Ethernet cable to connect the A/D Converter Card's RJ-45 RabbitNet jack to a RabbitNet port on the master. You may use either port if you are connecting to a master such as the BL2500 that has more than one RabbitNet port.

**NOTE:** The RJ-45 *RabbitNet* jacks are serial I/O ports for use with a master and a network of peripheral boards. The *RabbitNet* jacks do *not* support Ethernet connections.



**Figure 3. Connect A/D Converter Card to Master**

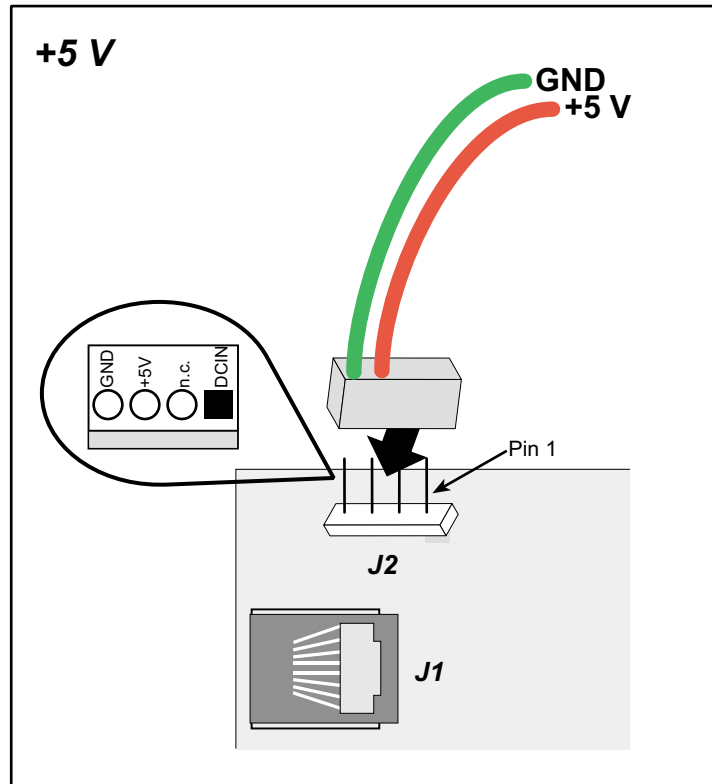
You will also have to provide +5 V DC power to your A/D Converter Card. The power supply is connected via the friction-lock terminal at header J2. If you are using a BL2500 as your master, you may draw this power from the BL2500 as shown in Figure 3. You may assemble a suitable cable using the friction-lock connectors from the Connectivity Kit described in Section 1.1.2. Although there is a standard RabbitNet DCIN power-supply input on the A/D Converter Card, the A/D Converter Card does not need DCIN power.

**NOTE:** Even if you are not drawing power from a BL2500 master, you will need to connect the A/D Converter Card ground to the ground on your master. The GND pin on header J2 should be used.

At the present time, you are limited by the number of RabbitNet ports on the master as to how many peripheral boards may be connected to that master. A router is being developed to allow additional peripheral boards to be used with one master. The router will also have connections available to provide the DCIN and +5 V power sources as well as a ground.

## 1.2.1 Power-Supply Connections

Figure 4 illustrates the assembled friction-lock connector wiring diagram for the power supplies used to supply power to the A/D Converter Card.



**Figure 4. Power-Supply Connections**

## 1.3 Pinout

The A/D Converter Card pinouts are shown in Figure 5.

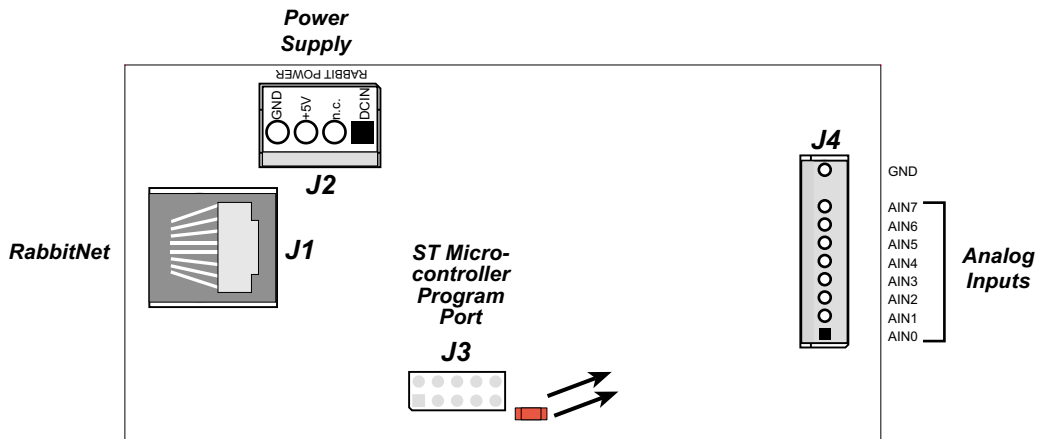


Figure 5. A/D Converter Card Pinouts

### 1.3.1 Headers

A/D Converter Cards are equipped with one polarized 1 × 9 friction-lock terminals at J4, a 1 × 4 friction-lock terminal at J2 (DCIN and +5 V power supplies), and an RJ-45 RabbitNet jack.

No header is installed at J3, which is used to program the A/D Converter Card at the factory.

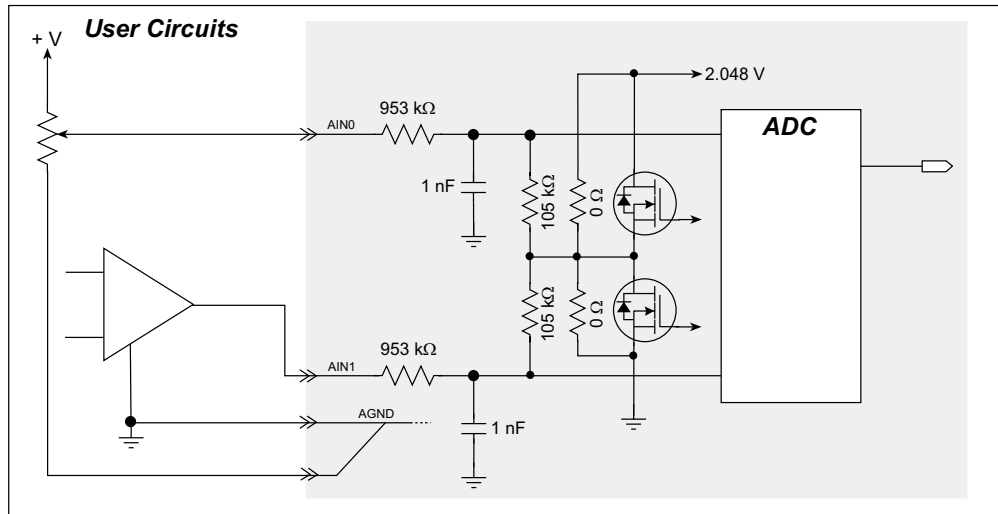
### 1.3.2 Indicator LED

An indicator LED located near the header J3 location turns on when the A/D Converter Card is powered up, then goes off when the A/D Converter Card has completed its initialization process and is running. The LED will be on while the A/D Converter Card is receiving a transmission from the master.

## 1.4 Analog Inputs

The single A/D converter used in the A/D Converter Card has a resolution of 11 bits (single-ended mode) or 12 bits (differential mode). There are eight channels of A/D conversion; the differential mode uses two channels for each differential-mode input.

Figure 6 shows a pair of A/D converter input circuits. Each A/D converter input essentially consists of resistors and a capacitor. The resistors form a 10:1 attenuator, and the capacitor protects the A/D converter input against electrostatic discharges.



**Figure 6. A/D Converter Inputs**

The A/D converter chip can make either single-ended or differential measurements depending on the value of the `opmode` parameter in the software function call, which turns the appropriate MOSFETs on or off and configures the A/D converter chip. Adjacent A/D converter inputs are paired when you make differential measurements. For single-ended conversions, the A/D converter chip works only with positive voltages for the ranges listed in Table 2.

**Table 2. Positive A/D Converter Input Voltage Ranges**

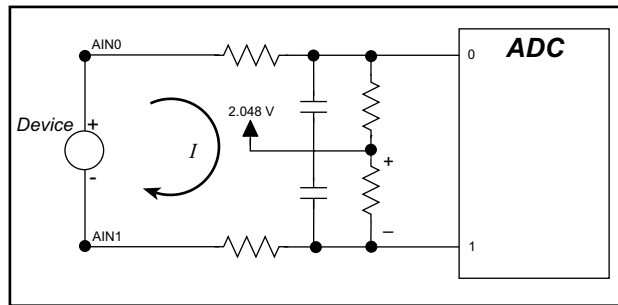
Min. Voltage (V)	Max. Voltage (V)	Amplifier Gain	mV per Tick
0.0	+20.0	1	10
0.0	+10.0	2	5
0.0	+5.0	4	2.5
0.0	+4.0	5	2.0
0.0	+2.5	8	1.25
0.0	+2.0	10	1.0
0.0	+1.25	16	0.625
0.0	+1.0	20	0.500

Many other possible ranges are possible by physically changing the resistor values that make up the attenuator circuit.

Differential measurements require two channels. As the name *differential* implies, the difference in voltage between the two adjacent channels is measured rather than the difference between the input and analog ground. Voltage measurements taken in the differential mode have a resolution of 12 bits, with the 12th bit indicating whether the difference is positive or negative.

When using the differential mode, the input can be either positive or negative, but **do not** exceed the maximum range by more than 20%.

If a device such as a battery is connected across two channels for a differential measurement, and it is **not** referenced to analog ground, then the current from the device will flow through both sets of attenuator resistors as shown in Figure 7. This will generate a negative voltage at one of the inputs, ADC1, which will almost certainly lead to inaccurate A/D conversions. To allow for such differential measurements, the A/D Converter Card uses a 2.048 V reference voltage. This allows input voltages that are negative with respect to analog ground. Table 3 provides the differential voltage ranges for this setup.



**Figure 7. Current Flow from Ungrounded or Floating Source**

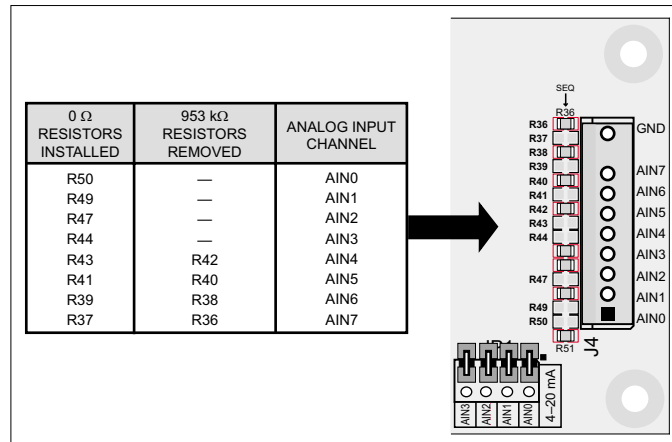
**Table 3. Differential Voltage Ranges**

Min. Differential Voltage (V)	Max. Differential Voltage (V)	Amplifier Gain	mV per Tick
0	±20.0	×1	10
0	±10.0	×2	5
0	±5.0	×4	2.5
0	±4.0	×5	2.0
0	±2.5	×8	1.25
0	±2.0	×10	1.00
0	±1.25	×16	0.625
0	±1.0	×20	0.500

The differential mode described above may also be used to measure negative voltages.

### 1.4.1 Analog Current Measurements

The A/D converter inputs can also be used with 4–20 mA current sources by measuring the resulting analog voltage drop across a 100  $\Omega$  1% precision resistor. These 100  $\Omega$  1% precision resistors are included on the A/D Converter Card for analog input channels AIN0–AIN3, and each of these channels may be configured individually using the jumpers on header JP1. If you need additional 4–20 mA inputs, you may use your own 100  $\Omega$  1% precision resistors for AIN4–AIN7 placed as shown in Figure 8.



**Figure 8. Locations of Resistors for 4–20 mA Mode**



**CAUTION:** The input impedance is low for the 4–20 mA current mode. Be careful not to exceed 2.5 V across the input when using the 4–20 mA current mode to keep the power dissipation by the 100  $\Omega$  precision resistors below their maximum rating.

For volume orders where 4–20 mA analog current conversions will be made, 0  $\Omega$  surface-mounted resistors can be installed at the factory instead of the jumpers on header JP1, and 0  $\Omega$  resistors can be added to allow AIN4–AIN7 to be used as 4–20 mA analog inputs. Contact your Z-World Sales representative at +1(707)757-3737 or your authorized distributor for more information on these options and the minimum order quantity.

The single-ended scale of 0–2.56 V with a gain of 8 is used to get an A/D current conversion of 12.5  $\mu$ A/tick.

## 1.4.2 Calibrating the A/D Converter Chip

Manufacturing tolerances for resistors, bias currents, offset voltages, gain, and the like introduce errors into the A/D conversions. Ideally there would be a one-to-one straight-line relationship between the input voltage and the output of the A/D converter, and a graph of such a line would have a slope of 1 and would pass through the (0,0) coordinate. However, the errors arising from manufacturing tolerances introduce a deviation between the applied input voltage and the voltage that is output by the A/D converter. The actual plot of voltage in vs. the voltage out from A/D converter is not actually a straight line. However, a straight line is a very good first-order approximation, and the calibration routines provided for the A/D Converter Card are based on a straight line with a slope of 1 and an offset from (0,0). The calibration routines use two known measurement points on the voltage-in vs. voltage-out line as the basis to calculate calibration constants that will be used to adjust for the slope of the line and the offset from (0,0). The calibration routines typically use input voltage points that are 10% less than the maximum and 10% more than the minimum readings possible for the A/D converter on any given range.

Quality calibration procedures are extremely important in obtaining good A/D converter results. No matter how high a resolution the A/D converter has, it cannot compensate for improper calibration. A/D converter results will never be more accurate than the meter used in the calibration process. Therefore, use the best digital volt and milli-amp meter available that meets or exceeds the accuracy of the A/D converter chip.

### 1.4.2.1 Modes

The A/D converter operates in three different modes:

- the single-ended mode,
- the differential mode, and
- the 4–20 mA current mode

The calibration and read routines provided correspond to these three modes.

### 1.4.2.2 Calibration Constants

The A/D converter has eight individual input channels, and each channel has eight programmable gains. Additionally, the A/D converter has the capability for adjacent inputs to be paired to make differential measurements with eight different gains, and provision is also made to convert 4–20 mA analog current measurements.

To get the best results from the A/D converter, it is necessary to calibrate each mode for each of its gains. The following table provides a grid for each possible set of calibration constants.

		Mode																
		Single-Ended								mA	Differential							
Gain		1	2	4	5	8	10	16	20	4	1	2	4	5	8	10	16	20
Input	0																	
	1																	
	2																	
	3																	
	4																	
	5																	
	6																	
	7																	

For the single-ended mode there are calibration constants for each channel and for each of its gains, for a total of 64 sets of calibration constants. The 4–20 mA mode covers 4–20 mA (actually 0–25 mA) currents. Separate calibration and read-back routines are provided for this. Since only one range of current measurement is provided, these routines use only one gain (4). One set of calibration constants is provided for each of the eight input channels. The differential-mode routines use a pair of input channels to make measurements. In this case, calibration constants are stored for each pair of channels and for each of the eight gains, for a total of 32 sets of calibration constants.

When a calibration is performed, it fills in one of the squares in the table with a set of calibration constants representing the corresponding mode, channel, and gain. These constants are stored in flash memory, and are thus maintained even when power is been removed from the A/D Converter Card. Note that calibration constants are stored for each of the modes. Since A/D converter read routines select the appropriate calibration constants based on the mode, it is possible for software calls to move from one mode to another without recalibration.

#### 1.4.2.3 Calibration Recommendations

It is imperative that you calibrate each of the A/D converter inputs in the same manner as they are to be used in the application. For example, if you will be performing floating differential measurements or differential measurements using a common analog ground, then calibrate the A/D converter in the corresponding manner. The calibration table only holds calibration constants based on mode, channel, and gain. *Other factors affecting the calibration must be taken into account by calibrating using the same method, mode, and gain setup as in the intended use.*

It is not necessary to fill out the entire calibration table. Only the entries associated with the modes, channels, and gains that you will be using are necessary. This fact can be used to simplify and speed up the calibration process.

Each calibration is normally done at 10% less than the maximum and 10% more than the minimum within a given voltage range defined by the mode, channel, and gain. However, if an application is known to use only portion of a particular range, it is possible to obtain improved accuracy by using calibration points that are 10% less than the expected maximum and 10% greater than the expected minimum.

#### 1.4.2.4 Factory Calibration

Because of the large number of possible calibrations, the factory performs only a rudimentary calibration on the A/D converter Card. The factory performs a single-ended calibration on each of the eight channels with a gain of 1 (0–20 V range). The remaining single-ended calibration constants for the other seven gains are approximated and are filled in based on the initial calibration. The milli-amp and differential portions of the table are filled in using typical expected values. All read routines will work properly with these factory-initialized calibration constants, but only the single-ended mode should be expected to return accurate results over a range of 0–20 V until you recalibrate the A/D Converter Card for your use.

Sample programs are provided to illustrate how to read and calibrate the various A/D inputs.

Mode	Read	Calibrate
Single-Ended, one channel	<code>AIN_RDSE_CH.C</code>	<code>AIN_CALSE_CH.C</code>
Single-Ended, all channels	–	<code>AIN_CALSE_ALL.C</code>
4–20 mA Current	<code>AIN_RDMA_CH.C</code>	<code>AIN_CALMA_CH.C</code>
Differential	<code>AIN_RDDIFF_CH.C</code>	<code>AIN_CALDIFF_CH.C</code>

These sample programs are found in the in the `SAMPLES\RABBITNET\RN1200` directory. See Section 2.2, “Sample Programs,” for more information on these sample programs and how to use them.

## 2. A/D CONVERTER CARD SOFTWARE

Dynamic C is an integrated development system for writing embedded software. It runs on an IBM-compatible PC and is designed for use with Z-World controllers and other controllers based on the Rabbit microprocessor.

Chapter 2 provides the libraries, function calls, and sample programs related to the A/D Converter Card.

### 2.1 Dynamic C Libraries

In addition to the library associated with the master single-board computer such as the BL2500 or OP7200, several other libraries are needed to provide function calls for the A/D Converter Card.

- **RN\_CFG\_BL25.LIB**—used to configure the BL2500 for use with RabbitNet peripheral boards. Function calls for this library are discussed in the *Coyote (BL2500) User's Manual*.
- **RN\_CFG\_OP72.LIB**—used to configure the OP7200 for use with RabbitNet peripheral boards. Function calls for this library are discussed in the *eDisplay (OP7200) User's Manual*.
- **RNET.LIB**—provides functions unique to the RabbitNet protocol. Function calls for this library are discussed in Appendix B., “RabbitNet.”
- **RNET\_DRIVER.LIB**—provides background functions unique to the RabbitNet data transmission protocol.
- **RNET\_AIN.LIB**—provides functions unique to the analog inputs on the A/D Converter Card and the A/D Converter Card. Function calls for this library are discussed in this chapter.

Other functions applicable to all devices based on Rabbit microprocessors are described in the *Dynamic C Function Reference User's Manual*. Functions relevant to the other peripheral boards are described in the manual specific to the peripheral board.

### 2.1.1 Accessing and Downloading Dynamic C Libraries

The libraries needed to run the A/D Converter Card are available on the CD included with the Development Kit for the master single-board computer, or they may be downloaded from <http://www.zworld.com/support/downloads/> on Z-World's Web site.

When downloading the libraries from the Web site, click on the product-specific links until you reach the links for the RabbitNet peripheral cards download. Once you have downloaded the file, double-click on the file name to begin the installation. InstallShield will install the files for you at a location you designate, and a pop-up **readme** file will explain the available options to add the files to your existing Dynamic C installation or to modify the relevant files in your existing Dynamic C installation.

You will be able to use the revamped Dynamic C installation with the A/D Converter Card and you will continue to be able to use this installation with all the other Z-World products you were able to use before.

## 2.2 Sample Programs

Sample programs are provided in the Dynamic C **SAMPLES** folder.

The various folders contain specific sample programs that illustrate the use of the corresponding Dynamic C libraries. For example, the sample program **PONG.C** demonstrates the output to the **STDIO** window.

The **RABBITNET** folder provides sample programs specific to the RabbitNet peripheral boards. Each sample program has comments that describe the purpose and function of the program. Follow the instructions at the beginning of the sample program.

To run a sample program, open it with the **File** menu (if it is not still open), compile it using the **Compile** menu, and then run it by selecting **Run** in the **Run** menu. The RabbitNet peripheral board must be connected to a master such as the BL2500 with its Demonstration Board connected as explained in the *Coyote (BL2500) User's Manual* or other user's manual. The BL2500 or other master must be in Program Mode, and must be connected via the programming cable to a PC.

More complete information on Dynamic C is provided in the *Dynamic C User's Manual*.

### 2.2.1 General RabbitNet Operation

The **SAMPLES\RABBITNET\** subdirectory contains the following sample programs. When running these sample programs, the A/D Converter Card may be connected to either RabbitNet port on a master such as the BL2500 that has two RabbitNet ports. The sample program will use **rn\_device()** to first look for peripheral boards connected to the master. The last peripheral board found will be used to run the sample program. The sample program will also display the serial number(s) of the peripheral boards connected to which RabbitNet port on the master using the **STDIO** window, or that no board is connected to a particular port.

- **ECHOCHAR.C**—Demonstrates a simple character echo to any RabbitNet peripheral board. A character is sent to the RabbitNet peripheral board connected at a physical node address of 0x00 or 0000 octal. If a peripheral board is connected, the character will be returned back along with the status of the peripheral board. Otherwise, the status byte will indicate there is no connection.
- **ECHOTERM.C**—Demonstrates a simple character echo to any RabbitNet peripheral board through a serial terminal on the master. A character is sent to the RabbitNet peripheral board connected at a physical-node address of 0x00 or 0000 octal. If a board is connected, the character will be returned back along with the status of the peripheral board. Otherwise, the status byte will indicate there is no connection.
- **HWATCHDOG.C**—Demonstrates setting the hardware watchdog on a RabbitNet peripheral board. This sample program will first look for a peripheral board that matches the search criteria. The hardware watchdog will be set and a hardware reset should occur in approximately 1.5 seconds. The hardware watchdog will be disabled after the reset is done.
- **SWATCHDOG.C**—Demonstrates setting and hitting the software watchdog on a RabbitNet peripheral board using costatements. This program will first look for a peripheral board matching the search criteria. The software watchdog will be set for 2.5 seconds. The watchdog will be hit at every increasing timeout until the timeout is past 2.5 seconds. A software reset will occur and the software watchdog will be disabled.

## 2.2.2 A/D Converter Inputs

The `SAMPLES\RABBITNET\RN1200` subdirectory contains the following sample programs.

**NOTE:** The Demonstration Board does not have to be connected to run these sample programs.

- **AIN\_CALDIFF\_CH.C**—Demonstrates how to recalibrate a differential A/D converter channel using two known voltages to generate constants for that channel that are rewritten into the A/D Converter Card flash. The voltages being monitored will be displayed continuously.
- **AIN\_CALMA\_CH.C**—Demonstrates how to recalibrate a 4–20 mA A/D converter channel using two known currents to generate constants for that channel that are rewritten into the A/D Converter Card flash. The currents being monitored will be displayed continuously.
- **AIN\_CALSE\_ALL.C**—Demonstrates how to recalibrate all the single-ended A/D converter channels for one gain using two known voltages to generate constants for each channel that are rewritten into the A/D Converter Card flash. A hardware reset will be issued to complete writes to flash once the constants are written, and the hardware watchdog will be set.
- **AIN\_CALSE\_CH.C**—Demonstrates how to recalibrate one single-ended A/D converter channel using two known voltages to generate constants for that channel that are rewritten into the A/D Converter Card flash. A hardware reset will be issued to complete writes to flash once the constants are written, and the hardware watchdog will be set.
- **AIN\_RDDIFF\_CH.C**—Demonstrates reading a differential A/D converter channel using two known voltages and constants for that channel. The voltage being monitored will be displayed continuously in the **STDIO** window.
- **AIN\_RDMA\_CH.C**—Demonstrates reading a 4–20 mA A/D converter channel. The current being monitored will be displayed continuously in the **STDIO** window.
- **AIN\_RDSE\_CH.C**—Reads and displays the voltage and equivalent values of one single-ended analog input channel. Coefficients are read from the A/D Converter Card. The computed raw data and equivalent voltages will be displayed.

## 2.3 A/D Converter Card Function Calls

### 2.3.1 Analog Input Function Calls

```
int rn_anaInConfig(int handle, int channel,  
int opmode, int gaincode, int reserved);
```

Configures each analog input channel to the desired operation at the desired gain. Once all channels have been set to single-ended voltages, differential voltages, or current, use `rn_anaIn()`, `rn_anaInVolts`, `rn_anaInmAmps`, or `rn_anaInDiff` to read an A/D converter channel.

#### PARAMETERS

**handle** is an address index to device information. Use `rn_device()` or `rn_find()` to establish the handle.

**channel** is the analog input channel number (0 to 7) corresponding to AIN0–AIN7

channel	SINGLE	DIFF	mAMP
0	+AIN0	+AIN0 -AIN1	+AIN0
1	+AIN1	—	+AIN1
2	+AIN2	+AIN2 -AIN3	+AIN2
3	+AIN3	—	+AIN3
4	+AIN4	+AIN4 -AIN5	+AIN4*
5	+AIN5	—	+AIN5*
6	+AIN6	+AIN6 -AIN7	+AIN6*
7	+AIN7	—	+AIN7*

\* These channels need to be configured for current measurements as explained in Section 1.4.1.

**opmode** is the mode of operation for the specified channel. Use one of the following macros to set the mode for the channel being configured.

**RNSINGLE**—single-ended input line, background sampling enabled

**RNDIFF**—differential input line, background sampling enabled

**RNmAMP**—4–20 mA input line, background sampling enabled

**gaincode** is the gain code of 0 to 7 (use a gain code of 4 for 4–20 mA operation)

Gain Code	Multiplier	Voltage Range	
		Single-Ended	Differential
0	×1	0–20 V	± 20 V
1	×2	0–10 V	± 10 V
2	×4	0–5 V	± 5 V
3	×5	0–4 V	± 4 V
4	×8	0–2.5 V	± 2.5 V
5	×10	0–2 V	± 2 V
6	×16	0–1.25 V	± 1.25 V
7	×20	0–1 V	± 1 V

**reserved** is reserved for future use. Set to 0.

#### RETURN VALUE

The status byte from the previous command. -1 means that device information indicates the A/D Converter Card is not connected to the master.

#### SEE ALSO

`rn_anaIn`, `rn_anaInVolts`, `rn_anaInDiff`, `rn_anaInmAmps`

```
int rn_anaIn(int handle, int channel, int *retdata,  
int sample, int reserved);
```

Reads the raw data value of an analog input channel. Set the **sample** parameter greater than one to average the readings.

This function provides the fastest A/D conversion rate.

#### PARAMETERS

**handle** is an address index to device information. Use **rn\_device()** or **rn\_find()** to establish the handle.

**channel** is the channel number (0 to 7) corresponding to AIN0–AIN7

**retdata** is a pointer to a raw data value of 0–2047 for 11-bit A/D conversions with a signed 12th bit

**sample** is x number of samples

1 = read current value of the given A/D converter channel

>1 = read given channel x times, and average the readings taken

**reserved** is reserved for future use. Set to 0.

#### RETURN VALUE

The status byte from the previous command and a return pointer to the raw input data. -1 means that device information indicates the A/D Converter Card is not connected to the master.

#### SEE ALSO

**rn\_anaInConfig, rn\_anaInVolts, rn\_anaInmAmps, rn\_anaInDiff**

```
int rn_anaInVolts(int handle, int channel,  
float *retdata, int sample, int reserved);
```

Reads the state of an analog input channel. Set the **sample** parameter greater than one to average the readings.

#### PARAMETERS

**handle** is an address index to device information. Use **rn\_device()** or **rn\_find()** to establish the handle.

**channel** is the channel number (0 to 7) corresponding to AIN0–AIN7

**retdata** is a pointer to the voltage value (if there is a data overflow or an out-of-range error, the value will be set to -4096 as defined by the macro **ADOVERFLOW**)

**sample** is x number of samples

1 = read current value of the given A/D converter channel

>1 = read given channel x times, and average the readings taken

**reserved** is reserved for future use. Set to 0.

#### RETURN VALUE

The status byte from the previous command and a return pointer to the voltage input data. -1 means that device information indicates the A/D Converter Card is not connected to the master.

#### SEE ALSO

**rn\_anaInConfig**, **rn\_anaIn**, **rn\_anaInmAmps**, **rn\_anaInDiff**

```
int rn_anaInDiff(int handle, int channel,
float *retdata, int sample, int reserved);
```

Reads the state of an analog input channel. Set the **sample** parameter greater than one to average the readings.

#### PARAMETERS

**handle** is an address index to device information. Use **rn\_device()** or **rn\_find()** to establish the handle.

**channel** is the channel number (0, 2, 4, 6)

channel	DIFF
0	+AIN0 -AIN1
2	+AIN2 -AIN3
4	+AIN4 -AIN5
6	+AIN6 -AIN7

**retdata** is a pointer to the voltage value (if there is a data overflow or an out-of-range error, the value will be set to -4096 as defined by the macro **ADOVERFLOW**)

**sample** is x number of samples

1 = read current value of the given A/D converter channel

>1 = read given channel x times, and average the readings taken

**reserved** is reserved for future use. Set to 0.

#### RETURN VALUE

The status byte from the previous command and a return pointer to the voltage input data. -1 means that device information indicates the A/D Converter Card is not connected to the master.

#### SEE ALSO

**rn\_anaInConfig**, **rn\_anaIn**, **rn\_anaInmAmps**, **rn\_anaInVolts**

```
int rn_anaInmAmps(int handle, int channel,  
float *retdata, int sample, int reserved);
```

Reads the state of a 4–20 mA analog input channel. Set the **sample** parameter greater than one to average the readings.

#### PARAMETERS

**handle** is an address index to device information. Use **rn\_device()** or **rn\_find()** to establish the handle.

**channel** is the channel number (0 to 7) corresponding to AIN0–AIN7

**retdata** is a pointer to the 4–20 mA value (if there is a data overflow or an out-of-range error, the value will be set to -4096 as defined by the macro **ADOVERFLOW**)

**sample** is x number of samples

1 = read current value of the given A/D converter channel

>1 = read given channel x times, and average the readings taken

**reserved** is reserved for future use. Set to 0.

#### RETURN VALUE

The status byte from the previous command and a return pointer to the 4–20 mA input data. -1 means that device information indicates the A/D Converter Card is not connected to the master.

#### SEE ALSO

**rn\_anaInConfig**, **rn\_anaIn**, **rn\_anaInVolts**, **rn\_anaInDiff**

```
int rn_anaInCalib(int channel, int opmode,
    int gaincode, int value1, float volts1,
    int value2, float volts2, rn_AinData *adata);
```

Calibrates the response of an analog input channel as a linear function using the two conversion points provided. Four values are calculated, and the results are sent later to the analog input device using the function `anaInWrCalib()`.

Each channel will have the following information:

- a linear constant or gain,
- a voltage offset.

**NOTE:** Typical calibration constants are loaded at the factory. This function should be used when you need more precise calibration or recalibration, or the calibration constants were corrupted in the device.

#### PARAMETERS

**channel** is the analog input channel number (0 to 7) corresponding to AIN0–AIN7

channel	SINGLE	DIFF	mAMP
0	+AIN0	+AIN0 -AIN1	+AIN0
1	+AIN1	—	+AIN1
2	+AIN2	+AIN2 -AIN3	+AIN2
3	+AIN3	—	+AIN3
4	+AIN4	+AIN4 -AIN5	+AIN4*
5	+AIN5	—	+AIN5*
6	+AIN6	+AIN6 -AIN7	+AIN6*
7	+AIN7	—	+AIN7*

\* These channels need to be configured for current measurements as explained in Section 1.4.1.

**opmode** is the mode of operation for the specified channel:

- RNSINGLE**—single-ended input line
- RNDIFF**—differential input line
- RNmAMP**—4–20 mA input line

**gaincode** is the gain code of 0 to 7 (use a gain code of 4 for 4–20 mA operation)

Gain Code	Multiplier	Voltage Range	
		Single-Ended	Differential
0	×1	0–20 V	± 20 V
1	×2	0–10 V	± 10 V
2	×4	0–5 V	± 5 V
3	×5	0–4 V	± 4 V
4	×8	0–2.5 V	± 2.5 V
5	×10	0–2 V	± 2 V
6	×16	0–1.25 V	± 1.25 V
7	×20	0–1 V	± 1 V

**value1** is the first raw data value read from the A/D converter channel

**volts1** is the voltage corresponding to the first input value (minimum and maximum voltage with respect to the limits of the analog input)

**value2** is the second raw data value read from the A/D converter channel

**volts2** is the voltage corresponding to the second input value ((minimum and maximum voltage with respect to the limits of the analog input)

**rn\_AinData \*adata** is a pointer to the structure where the calibration constants, gain, and offset are written to after being calculated

**RETURN VALUE**

- 0, if successful.
- 1 if not able to make calibration constants.

**SEE ALSO**

**rn\_anaInWrCalib**

```
int rn_anaInWrCalib(int handle, int channel,
    int opmode, int gaincode, rn_AinData adata,
    int reserved);
```

Writes the calibration constants, gain, and offset previously calculated by `rn_anaInCalib()` into the analog device flash memory. A hardware reset (`rn_reset()`) and a read reset register (`rn_rst_status()`) must be issued after this function is called.

**NOTE:** Typical calibration constants are loaded at the factory. This function should be used when you need more precise calibration or recalibration, or the calibration constants were corrupted in the device.

#### PARAMETERS

**handle** is an address index to device information. Use `rn_device()` or `rn_find()` to establish the handle.

**channel** is the analog input channel number (0 to 7) corresponding to AIN0–AIN7

channel	SINGLE	DIFF	mAMP
0	+AIN0	+AIN0 -AIN1	+AIN0
1	+AIN1	—	+AIN1
2	+AIN2	+AIN2 -AIN3	+AIN2
3	+AIN3	—	+AIN3
4	+AIN4	+AIN4 -AIN5	+AIN4*
5	+AIN5	—	+AIN5*
6	+AIN6	+AIN6 -AIN7	+AIN6*
7	+AIN7	—	+AIN7*

\* These channels need to be configured for current measurements as explained in Section 1.4.1.

**opmode** is the mode of operation for the specified channel:

**RNSINGLE**—single-ended input line

**RNDIFF**—differential input line

**RNmAMP**—4–20 mA input line

**gaincode** is the gain code of 0 to 7 (use a gain code of 4 for 4–20 mA operation)

Gain Code	Multiplier	Voltage Range	
		Single-Ended	Differential
0	×1	0–20 V	± 20 V
1	×2	0–10 V	± 10 V
2	×4	0–5 V	± 5 V
3	×5	0–4 V	± 4 V
4	×8	0–2.5 V	± 2.5 V
5	×10	0–2 V	± 2 V
6	×16	0–1.25 V	± 1.25 V
7	×20	0–1 V	± 1 V

**rn\_AinData adata** is a pointer to the structure where the calibration constants, gain, and offset are written to after being calculated

**reserved** is reserved for future use. Set to 0.

#### RETURN VALUE

The status byte from the previous command and a return pointer to the raw input data. -1 means that device information indicates the A/D Converter Card is not connected to the master.

#### SEE ALSO

**rn\_anaInRdCalib**, **rn\_anaInCalib**

```
int rn_anaInRdCalib(int handle, int channel,
    int opmode, int gaincode, rn_AinData *adata,
    int reserved);
```

Reads the calibration constants, gain, and offset from a device with analog inputs.

#### PARAMETERS

**handle** is an address index to device information. Use `rn_device()` or `rn_find()` to establish the handle.

**channel** is the analog input channel number (0 to 7) corresponding to AIN0–AIN7

channel	SINGLE	DIFF	mAMP
0	+AIN0	+AIN0 -AIN1	+AIN0
1	+AIN1	—	+AIN1
2	+AIN2	+AIN2 -AIN3	+AIN2
3	+AIN3	—	+AIN3
4	+AIN4	+AIN4 -AIN5	+AIN4*
5	+AIN5	—	+AIN5*
6	+AIN6	+AIN6 -AIN7	+AIN6*
7	+AIN7	—	+AIN7*

\* These channels need to be configured for current measurements as explained in Section 1.4.1.

**opmode** is the mode of operation for the specified channel:

**RNSINGLE**—single-ended input line

**RNDIFF**—differential input line

**RNmAMP**—4–20 mA input line

**gaincode** is the gain code of 0 to 7 (use a gain code of 4 for 4–20 mA operation)

Gain Code	Multiplier	Voltage Range	
		Single-Ended	Differential
0	×1	0–20 V	± 20 V
1	×2	0–10 V	± 10 V
2	×4	0–5 V	± 5 V
3	×5	0–4 V	± 4 V
4	×8	0–2.5 V	± 2.5 V
5	×10	0–2 V	± 2 V
6	×16	0–1.25 V	± 1.25 V
7	×20	0–1 V	± 1 V

**rn\_AinData \*adata** is a pointer to the structure where the calibration constants, gain, and offset are written to after being calculated

**reserved** is reserved for future use. Set to 0.

**RETURN VALUE**

The status byte from the previous command and a return pointer to the raw input data. -1 means that device information indicates the A/D Converter Card is not connected to the master.

**SEE ALSO**

`rn_anaInWrCalib`, `rn_anaInCalib`

### 2.3.2 Status Byte

Unless otherwise specified, functions returning a status byte for A/D Converter Cards will have the following format for each designated bit.

7	6	5	4	3	2	1	0	
×	×							00 = Reserved 01 = Ready 02 = Busy 03 = Device not connected
		×						0 = Device 1 = Router
			×					0 = No error 1 = Communication error*
				×				Reserved
					×			Reserved
						×		0 = Last command accepted 1 = Last command unexecuted
							×	0 = Not expired 1 = HW or SW watchdog timer expired†

\* Use the function `rn_comm_status()` to determine which error occurred.

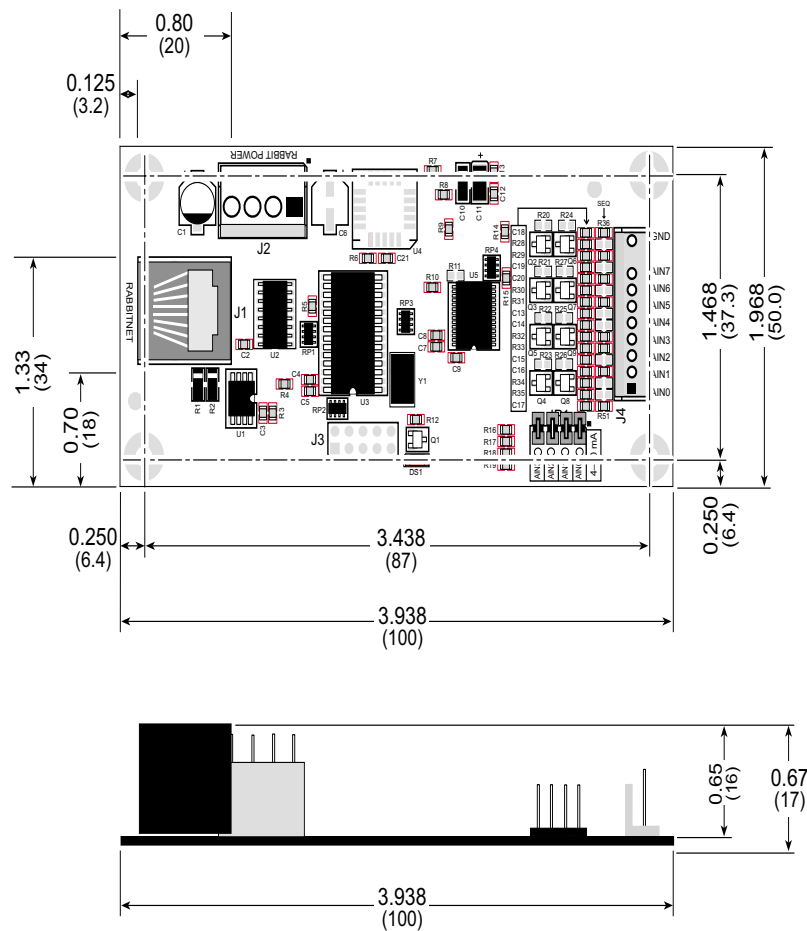
† Use the function `rn_rst_status()` to determine which timer expired.



# APPENDIX A. A/D CONVERTER CARD SPECIFICATIONS

## A.1 Electrical and Mechanical Specifications

Figure A-1 shows the mechanical dimensions for the A/D Converter Card.



**Figure A-1. A/D Converter Card Dimensions**

**NOTE:** All diagram and graphic measurements are in inches followed by millimeters enclosed in parentheses.

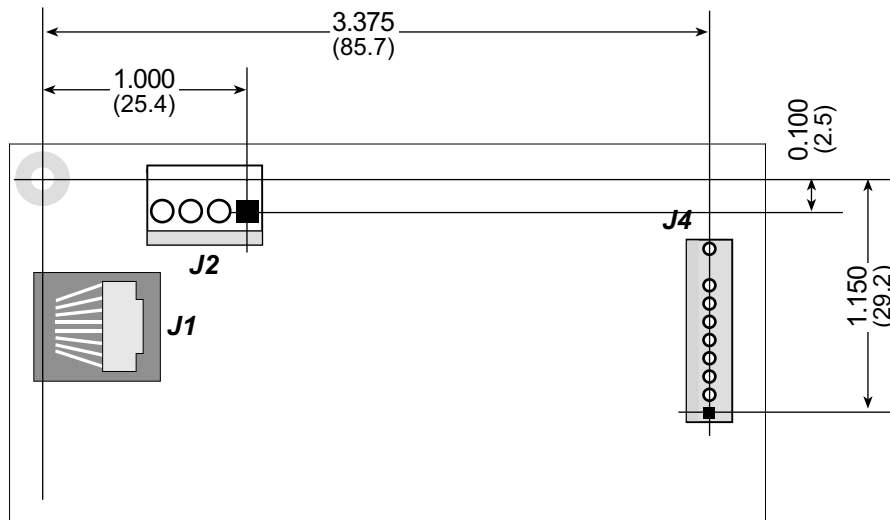
Table A-1 lists the electrical, mechanical, and environmental specifications for the A/D Converter Card.

**Table A-1. A/D Converter Card Specifications**

Feature	Specification
Analog Inputs	8 single-ended 11-bit or 4 differential 12-bit analog inputs, 1 M $\Omega$ input impedance, 2.5 ksamples/s sampling rate, all 8 channels can be configured as 11-bit 4–20 mA analog inputs <ul style="list-style-type: none"> <li>• software-controlled ranges:                0–1 V, 2 V, 5 V, 10 V, 20 V DC (single-ended) or  <math>\pm 1</math> V, <math>\pm 2</math> V, <math>\pm 5</math> V, <math>\pm 10</math> V, <math>\pm 20</math> V DC (differential)</li> </ul>
RabbitNet™ Serial Port	RS-422 SPI, 1 Mbits/s
Power	V <sub>cc</sub> : +5 V DC, 100 mA
Temperature	-40°C to +70°C
Humidity	5% to 95%, noncondensing
Connectors	Friction-lock connectors: one polarized 9-position terminals with 0.1" pitch one 4-position terminal with 0.156" pitch One RJ-45 RabbitNet™ jack
Board Size	1.97" $\times$ 3.94" $\times$ 0.67" (50 mm $\times$ 100 mm $\times$ 17 mm)

### A.1.1 Physical Mounting

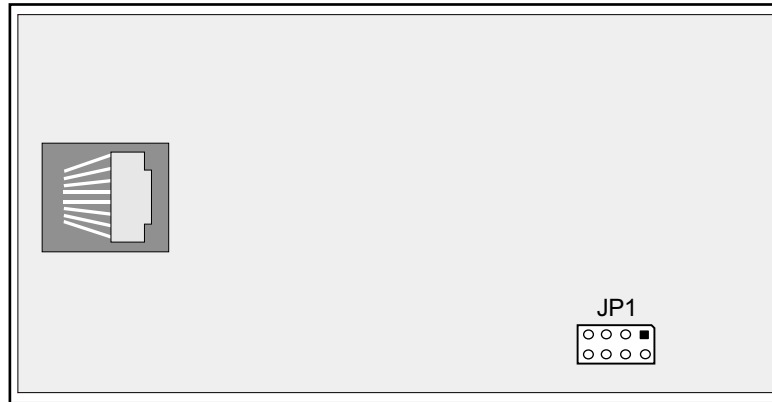
Figure A-2 shows position information to assist with interfacing other boards with the A/D Converter Card.



**Figure A-2. User Board Footprint for A/D Converter Card**

## A.2 Jumper Configurations

Figure A-3 shows the header and jumper locations used to configure the various A/D Converter Card options.



**Figure A-3. Location of A/D Converter Card Configurable Positions**

Table A-2 lists the configuration options. Standard pluggable jumpers are used.

**Table A-2. A/D Converter Card Jumper Configurations**

Header	Description	Pins Connected		Factory Default
JP1	Analog Voltage/4–20 mA Options	1–2	Connect for 4–20 mA option on AIN0	n.c.
		3–4	Connect for 4–20 mA option on AIN1	n.c.
		5–6	Connect for 4–20 mA option on AIN2	n.c.
		7–8	Connect for 4–20 mA option on AIN3	n.c.

# APPENDIX B. RABBITNET

## B.1 General RabbitNet Description

RabbitNet is a high-speed synchronous protocol developed by Z-World to connect peripheral cards to a master and to allow them to communicate with each other. A communication path is established and controlled by the master, and each master can, in theory, control up to 196 peripheral cards. All RabbitNet connections are made point to point, and until a router is made available, a RabbitNet master port can only be connected directly to a peripheral card, and the number of peripheral cards is limited by the number of available RabbitNet ports on the master.

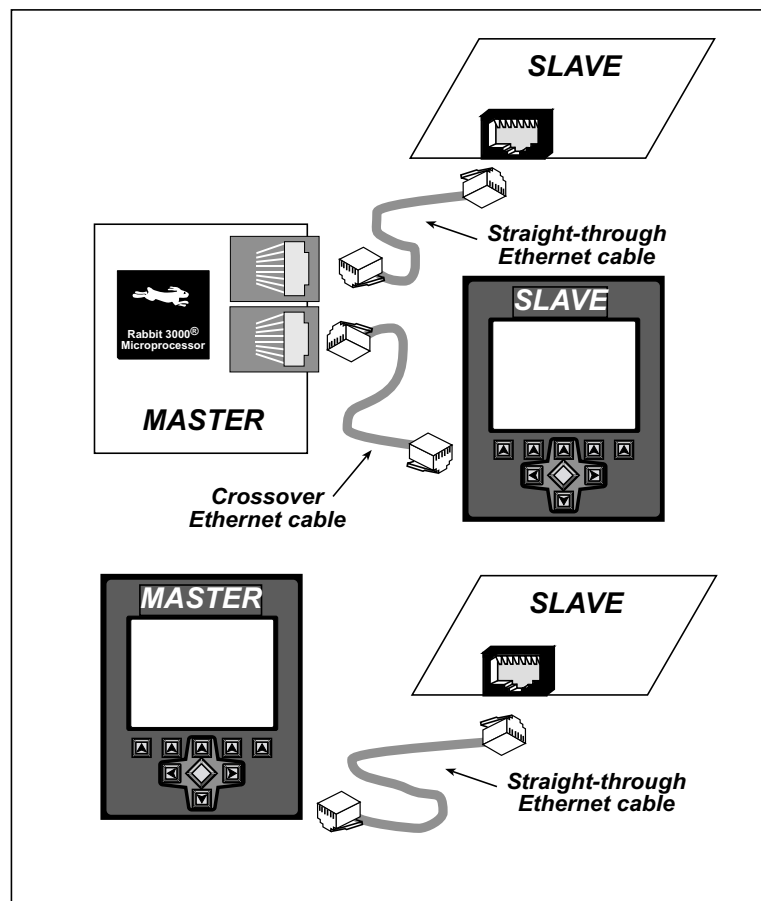


Figure B-1. Connecting Peripheral Cards to a Master

Use a straight-through Ethernet cable to connect the master to slave peripheral cards, unless you are using a device such as the OP7200 that could be used either as a master or a slave. In this case you would use a crossover cable to connect an OP7200 that is being used as a slave.

Distances between a master unit and peripheral cards can be up to 10 m or 33 ft.

The following low-cost peripheral cards are currently available or are being developed.

- Digital I/O

24 inputs, 16 push/pull outputs, 4 channels of 10-bit A/D conversion with ranges of 0 to 10 V, 0 to 1 V, and -0.25 to +0.25 V. The following connectors are used:

Signal = 0.1" friction-lock connectors

Power = 0.156" friction-lock connectors

RabbitNet = RJ-45 connector

- A/D converter

8 channels of programmable-gain 12-bit A/D conversion, configurable as current measurement and differential-input pairs. 2.5 V reference voltage is available on the connector. The following connectors are used:

Signal = 0.1" friction-lock connectors

Power = 0.156" friction-lock connectors

RabbitNet = RJ-45 connector

- D/A converter

8 channels of 0–10 V 12-bit D/A conversion. The following connectors are used:

Signal = 0.1" friction-lock connectors

Power = 0.156" friction-lock connectors

RabbitNet = RJ-45 connector

- Display/Keypad card

122 × 32 graphic display with 1 × 7 keypad and optional bezel, similar to the LCD/keypad module sold by Z-World for use with OP6800, Smart Star, BL2100, and select RabbitCore modules. Only one RabbitNet Display per master is supported at this time. The following connectors are used:

Power = 0.156" friction-lock connectors

RabbitNet = RJ-45 connector

Visit [Z-World's Web site](#) for up-to-date information about additional cards and features as they become available. The Web site also has the latest revision of this user's manual.

## B.2 Physical Implementation

There are four signaling functions associated with a RabbitNet connection. From the master's point of view, the transmit function carries information and commands to the peripheral card. The receive function is used to read back information sent to the master by the peripheral card. A clock is used to synchronize data going between the two devices at high speed. The master is the source of this clock. A slave select (SS) function originates at the master, and when detected by a peripheral card causes it to become selected and respond to commands received from the master.

The signals themselves are differential RS-422, which are series-terminated at the source. With this type of termination, the maximum frequency is limited by the round-trip delay time of the cable. Although a peripheral card could theoretically be up to 45 m (150 ft) from the master for a data rate of 1 MHz, Z-World recommends a practical limit of 10 m (33 ft).

Connections between peripheral cards and/or routers are done using standard 8-conductor Ethernet cables. Masters, peripheral cards, and routers are equipped with RJ-45 8-pin female connectors. The cables may be swapped end for end without affecting functionality.

### B.2.1 Control and Routing

Control starts at the master when the master asserts the slave select signal (SS). Then it simultaneously sends a serial command and clock. The first byte of a command contains the 8-bit address of the peripheral card if more than one peripheral card is connected to a port via a router.

A peripheral card assumes it is selected as soon as it receives the select signal. For direct master-to-peripheral-card connections, this is as soon as the master asserts the select signal. When a router or routers are in the path, an address must be sent first. The first router encountered decodes its assigned portion of the address byte and activates the addressed downstream port by asserting the associated select signal. The downstream port is activated only after the address is processed. The command can pass through one or more routers in this way before reaching the peripheral card. The connection is established once the select signal reaches the addressed slave. At this point communication between the master and the selected peripheral card is established, and data can flow in both directions simultaneously. The connection is maintained so long as the master asserts the select signal.

## B.3 Function Calls

The function calls described in this section are used with all RabbitNet peripheral cards, and are available in the `RNET.LIB` library in the Dynamic C `RABBITNET` folder.

```
int rn_init(char portflag, char servicetype);
```

Resets, initializes, or disables a specified RabbitNet port on the master single-board computer. During initialization, the network is enumerated and relevant tables are filled in. If the port is already initialized, calling this function forces a re-enumeration of all devices on that port.

Call this function first before using other RabbitNet functions.

### PARAMETERS

**portflag** is a bit that represents a RabbitNet port on the master single-board computer (from 0 to the maximum number of ports). A set bit requires a service. If **portflag** = 0x03, both RabbitNet ports 0 and 1 will need to be serviced.

**servicetype** enables or disables each RabbitNet port as set by the port flags.

0 = disable port

1 = enable port

### RETURN VALUE

0

```
int rn_device(char pna);
```

Returns an address index to device information from a given physical node address. This function will check device information to determine that the peripheral card is connected to a master.

### PARAMETER

**pn**a is the physical node address, indicated as a byte.

7,6—2-bit binary representation of the port number on the master

5,4,3—Level 1 router downstream port

2,1,0—Level 2 router downstream port

### RETURN VALUE

Pointer to device information. This is used by other functions as a handle to identify the peripheral card.

-1 indicates that the peripheral card either cannot be identified or is not connected to the master.

### SEE ALSO

`rn_find`

```
int rn_find(rn_search *srch);
```

Locates the first active device that matches the search criteria.

#### PARAMETER

**srch** is the search criteria structure **rn\_search**:

```
unsigned int flags;    // status flags see MATCH macros below
unsigned int ports;   // port bitmask
char productid;      // product id
char productrev;     // product rev
char coderev;        // code rev
long serialnum;      // serial number
```

Use a maximum of 3 macros for the search criteria:

```
RN_MATCH_PORT        // match port bitmask
RN_MATCH_PNA         // match physical node address
RN_MATCH_HANDLE      // match instance (reg 3)
RN_MATCH_PRDID       // match id/version (reg 1)
RN_MATCH_PRDREV      // match product revision
RN_MATCH_CODEREV     // match code revision
RN_MATCH_SN          // match serial number
```

For example:

```
rn_search newdev;
newdev.flags = RN_MATCH_PORT|RN_MATCH_SN;
newdev.ports = 0x03; //search ports 0 and 1
newdev.serialnum = E3446C01L;
handle = rn_find(&newdev);
```

#### RETURN VALUE

Returns the handle of the first device matching the criteria. 0 indicates no such devices were found.

#### SEE ALSO

**rn\_device**

```
int rn_echo(int handle, char sendecho,
char *recdata);
```

The peripheral card sends back the character the master sent. This function will check device information to determine that the peripheral card is connected to a master.

#### PARAMETERS

**handle** is an address index to device information. Use **rn\_device()** or **rn\_find()** to establish the handle.

**sendecho** is the character to echo back.

**recdata** is a pointer to the return address of the character from the device.

#### RETURN VALUE

The status byte from the previous command. -1 means that device information indicates the peripheral card is not connected to the master.

```
int rn_write(int handle, int regno, char *data,  
int datalen);
```

Writes a string to the specified device and register. Waits for results. This function will check device information to determine that the peripheral card is connected to a master.

#### PARAMETERS

**handle** is an address index to device information. Use `rn_device()` or `rn_find()` to establish the handle.

**regno** is the command register number as designated by each device.

**data** is a pointer to the address of the string to write to the device.

**datalen** is the number of bytes to write (0–15).

**NOTE:** A data length of 0 will transmit the one-byte command register number.

#### RETURN VALUE

The status byte from the previous command. -1 means that device information indicates the peripheral card is not connected to the master, and -2 means that the data length was greater than 15.

#### SEE ALSO

`rn_read`

```
int rn_read(int handle, int regno, char *reodata,  
int datalen);
```

Reads a string from the specified device and register. Waits for results. This function will check device information to determine that the peripheral card is connected to a master.

#### PARAMETERS

**handle** is an address index to device information. Use `rn_device()` or `rn_find()` to establish the handle.

**regno** is the command register number as designated by each device.

**reodata** is a pointer to the address of the string to read from the device.

**datalen** is the number of bytes to read (0–15).

**NOTE:** A data length of 0 will transmit the one-byte command register number.

#### RETURN VALUE

The status byte from the previous command. -1 means that device information indicates the peripheral card is not connected to the master, and -2 means that the data length was greater than 15.

#### SEE ALSO

`rn_write`

```
int rn_reset(int handle, int resettype);
```

Sends a reset sequence to the specified peripheral card. The reset takes approximately 25 ms before the peripheral card will once again execute the application. Allow 1.5 seconds after the reset has completed before accessing the peripheral card. This function will check peripheral card information to determine that the peripheral card is connected to a master.

#### PARAMETERS

**handle** is an address index to device information. Use `rn_device()` or `rn_find()` to establish the handle.

**resettype** describes the type of reset.

0 = hard reset—equivalent to power-up. All logic is reset.

1 = soft reset—only the microprocessor logic is reset.

#### RETURN VALUE

The status byte from the previous command. -1 means that device information indicates the peripheral card is not connected to the master.

```
int rn_sw_wdt(int handle, float timeout);
```

Sets software watchdog timeout period. Call this function prior to enabling the software watchdog timer. This function will check device information to determine that the peripheral card is connected to a master.

#### PARAMETERS

**handle** is an address index to device information. Use `rn_device()` or `rn_find()` to establish the handle.

**timeout** is a timeout period from 0.025 to 6.375 seconds in increments of 0.025 seconds. Entering a zero value will disable the software watchdog timer.

#### RETURN VALUE

The status byte from the previous command. -1 means that device information indicates the peripheral card is not connected to the master.

```
int rn_enable_wdt(int handle, int wdtttype);
```

Enables the hardware and/or software watchdog timers on a peripheral card. The software on the peripheral card will keep the hardware watchdog timer updated, but will hard reset if the time expires. The hardware watchdog cannot be disabled except by a hard reset on the peripheral card. The software watchdog timer must be updated by software on the master. The peripheral card will soft reset if the timeout set by `rn_sw_wdt()` expires. This function will check device information to determine that the peripheral card is connected to a master.

#### PARAMETERS

**handle** is an address index to device information. Use `rn_device()` or `rn_find()` to establish the handle.

#### **wdtttype**

- 0 enables both hardware and software watchdog timers
- 1 enables hardware watchdog timer
- 2 enables software watchdog timer

#### RETURN VALUE

The status byte from the previous command. -1 means that device information indicates the peripheral card is not connected to the master.

#### SEE ALSO

`rn_hitwd`, `rn_sw_wdt`

```
int rn_hitwd(int handle, char *count);
```

Hits software watchdog. Set the timeout period and enable the software watchdog prior to using this function. This function will check device information to determine that the peripheral card is connected to a master.

#### PARAMETERS

**handle** is an address index to device information. Use `rn_device()` or `rn_find()` to establish the handle.

**count** is a pointer to return the present count of the software watchdog timer. The equivalent time left in seconds can be determined from `count × 0.025` seconds.

#### RETURN VALUE

The status byte from the previous command. -1 means that device information indicates the peripheral card is not connected to the master.

#### SEE ALSO

`rn_enable_wdt`, `rn_sw_wdt`

```
int rn_rst_status(int handle, char *retdata);
```

Reads the status of which reset occurred and whether any watchdogs are enabled.

#### PARAMETERS

**handle** is an address index to device information. Use **rn\_device()** or **rn\_find()** to establish the handle.

**retdata** is a pointer to the return address of the communication byte. A set bit indicates which error occurred. This register is cleared when read.

- 7—HW reset has occurred
- 6—SW reset has occurred
- 5—HW watchdog enabled
- 4—SW watchdog enabled
- 3,2,1,0—Reserved

#### RETURN VALUE

The status byte from the previous command.

```
int rn_comm_status(int handle, char *retdata);
```

#### PARAMETERS

**handle** is an address index to device information. Use **rn\_device()** or **rn\_find()** to establish the handle.

**retdata** is a pointer to the return address of the communication byte. A set bit indicates which error occurred. This register is cleared when read.

- 7—Data available and waiting to be processed MOSI (master out, slave in)
- 6—Write collision MISO (master in, slave out)
- 5—Overrun MOSI (master out, slave in)
- 4—Mode fault, device detected hardware fault
- 3—Data compare error detected by device
- 2,1,0—Reserved

#### RETURN VALUE

The status byte from the previous command.

### B.3.1 Status Byte

Unless otherwise specified, functions returning a status byte will have the following format for each designated bit.

7	6	5	4	3	2	1	0	
×	×							00 = Reserved 01 = Ready 02 = Busy 03 = Device not connected
		×						0 = Device 1 = Router
			×					0 = No error 1 = Communication error*
				×				Reserved for individual peripheral cards
					×			Reserved for individual peripheral cards
						×		0 = Last command accepted 1 = Last command unexecuted
							×	0 = Not expired 1 = HW or SW watchdog timer expired†

\* Use the function `rn_comm_status()` to determine which error occurred.

† Use the function `rn_rst_status()` to determine which timer expired.



## NOTICE TO USERS

Z-WORLD PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE-SUPPORT DEVICES OR SYSTEMS UNLESS A SPECIFIC WRITTEN AGREEMENT REGARDING SUCH INTENDED USE IS ENTERED INTO BETWEEN THE CUSTOMER AND Z-WORLD PRIOR TO USE. Life-support devices or systems are devices or systems intended for surgical implantation into the body or to sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling and user's manual, can be reasonably expected to result in significant injury.

No complex software or hardware system is perfect. Bugs are always present in a system of any size. In order to prevent danger to life or property, it is the responsibility of the system designer to incorporate redundant protective mechanisms appropriate to the risk involved.

All Z-World products are 100 percent functionally tested. Additional testing may include visual quality control inspections or mechanical defects analyzer inspections. Specifications are based on characterization of tested sample units rather than testing over temperature and voltage of each unit. Z-World products may qualify components to operate within a range of parameters that is different from the manufacturer's recommended range. This strategy is believed to be more economical and effective. Additional testing or burn-in of an individual unit is available by special arrangement.



# INDEX

## A

- A/D Converter Card
  - connection to master ..... 1, 4
  - power supplies ..... 4
- A/D converter inputs
  - 4–20 mA current measurements ..... 9
  - calibration ..... 10, 11, 12
  - differential measurements ... 8
  - sample programs ..... 12
  - single-ended measurements 7

## C

- connectivity tools
  - Connectivity Kit ..... 2
  - crimp tool ..... 2

## D

- dimensions
  - A/D Converter Card ..... 31
- DIN rail mounting ..... 3
  - components ..... 3
- Dynamic C ..... 2
  - downloading RabbitNet libraries ..... 14
  - libraries ..... 13

## F

- features ..... 2

## I

- indicator LED ..... 6

## J

- jumper configurations
  - A/D Converter Card ..... 34
    - JP1 (analog voltage/4–20 mA measurement options) ..... 34
  - jumper locations ..... 34

## P

- peripheral cards
  - connection to master ... 35, 36
- physical mounting
  - A/D Converter Card ..... 33
- pinout
  - A/D Converter headers ..... 6
- power supplies
  - A/D Converter Card ..... 4
  - wiring diagram ..... 5

## R

- RabbitNet
  - Ethernet cables to connect peripheral cards ..... 35, 36
  - general description ..... 35
  - peripheral cards ..... 36
  - physical implementation ... 37

## S

- sample programs ..... 15
  - A/D converter inputs ..... 12
    - AIN\_CALDIFF\_CH.C ..... 12, 16
    - AIN\_CALMA\_CH.C ..... 12, 16
    - AIN\_CALSE\_ALL.C ..... 12, 16
    - AIN\_CALSE\_CH.C ..... 12, 16
    - AIN\_RDDIFF\_CH.C ..... 12, 16
    - AIN\_RDMA\_CH.C ..... 12, 16
    - AIN\_RDSE\_CH.C .. 12, 16
- RabbitNet operation
  - ECHOCHAR.C ..... 15
  - ECHOTERM.C ..... 15
  - HWATCHDOG.C ..... 15
  - SWATCHDOG.C ..... 15

- software ..... 2, 13
  - analog input functions ..... 17
    - rn\_anaIn ..... 19
    - rn\_anaInCalib ..... 23
    - rn\_anaInConfig ..... 17
    - rn\_anaInDiff ..... 21
    - rn\_anaInmAmps ..... 22
    - rn\_anaInRdCalib ..... 27
    - rn\_anaInVolts ..... 20
    - rn\_anaInWrCalib ..... 25
  - downloading RabbitNet
    - libraries ..... 14
  - libraries ..... 13
    - RN\_CFG\_BL25.LIB .... 13
    - RN\_CFG\_OP72.LIB .... 13
    - RNET.LIB ..... 13, 38
    - RNET\_AIN.LIB ..... 13
    - RNET\_DRIVER.LIB .... 13
  - RNET.LIB
    - rn\_comm\_status ..... 43
    - rn\_device ..... 15, 38
    - rn\_echo ..... 39
    - rn\_enable\_wdt ..... 42
    - rn\_find ..... 39
    - rn\_hitwd ..... 42
    - rn\_init ..... 38
    - rn\_read ..... 40
    - rn\_reset ..... 41
    - rn\_rst\_status ..... 43
    - rn\_sw\_wdt ..... 41
    - rn\_write ..... 40
  - sample programs ..... 15
  - specifications
    - A/D Converter Card ..... 31
      - dimensions ..... 31
      - electrical ..... 32
      - header footprint ..... 33
      - physical mounting ..... 33
      - relative pin 1 locations .. 33
      - temperature ..... 32
    - status byte ..... 44
      - A/D Converter Card ..... 29





# SCHEMATICS

## **090-0178 A/D Converter Card Schematic**

[www.zworld.com/documentation/schemat/090-0178.pdf](http://www.zworld.com/documentation/schemat/090-0178.pdf)

The schematics included with the printed manual were the latest revisions available at the time the manual was last revised. The online versions of the manual contain links to the latest revised schematic on the Web site. You may also use the URL information provided above to access the latest schematics directly.

