

# **D/A Converter Card (RN1300)**

RabbitNet LAN Card

## **User's Manual**

019-0134 • 030725-A

# **RN1300 User's Manual**

Part Number 019-0134 • 030725-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

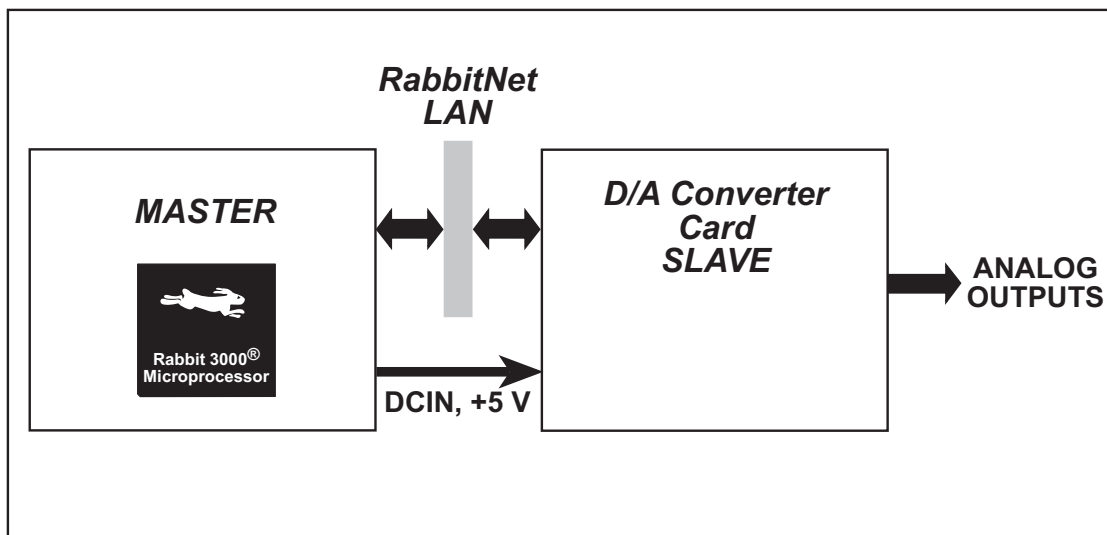
<b>Chapter 1. Overview</b>	<b>1</b>
1.1 D/A Converter Card Features .....	2
1.1.1 Software .....	2
1.1.2 Connectivity Tools .....	2
1.1.3 DIN Rail Mounting .....	3
1.2 Connecting Peripheral Cards .....	4
1.2.1 Power-Supply Connections .....	5
1.3 Pinout .....	6
1.3.1 Headers .....	6
1.3.2 Indicator LED .....	6
1.4 D/A Converter Outputs .....	7
1.4.1 Calibration .....	8
<b>Chapter 2. A/D Converter Card Software</b>	<b>9</b>
2.1 Dynamic C Libraries .....	9
2.1.1 Accessing and Downloading Dynamic C Libraries .....	10
2.2 Sample Programs .....	11
2.2.1 General RabbitNet Operation .....	11
2.2.2 D/A Converter Inputs .....	12
2.3 D/A Converter Card Function Calls .....	13
2.3.1 Analog Output Function Calls .....	13
<b>Appendix A. D/A Converter Card Specifications</b>	<b>21</b>
A.1 Electrical and Mechanical Specifications .....	21
A.1.1 Physical Mounting .....	23
<b>Appendix B. RabbitNet</b>	<b>25</b>
B.1 General RabbitNet Description .....	25
B.2 Physical Implementation .....	27
B.2.1 Control and Routing .....	27
B.3 Function Calls .....	28
B.3.1 Status Byte .....	34
<b>Notice to Users</b>	<b>35</b>
<b>Index</b>	<b>37</b>
<b>Schematics</b>	<b>39</b>



# 1. OVERVIEW

Chapter 1 describes the features and the use of the D/A 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 D/A Converter Card connected to a master.



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

## 1.1 D/A Converter Card Features

- 8 channels of 12-bit analog outputs
- 2 channels are software-configurable for output voltage ranges of 0–2.5 V, 0–5 V, 0–10 V, or 0–20 V, remaining 6 channels have software-configurable output voltage ranges of 0–10 V or 0–20 V
- 2.5 kHz update rate
- output impedance 8  $\Omega$
- 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 D/A 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

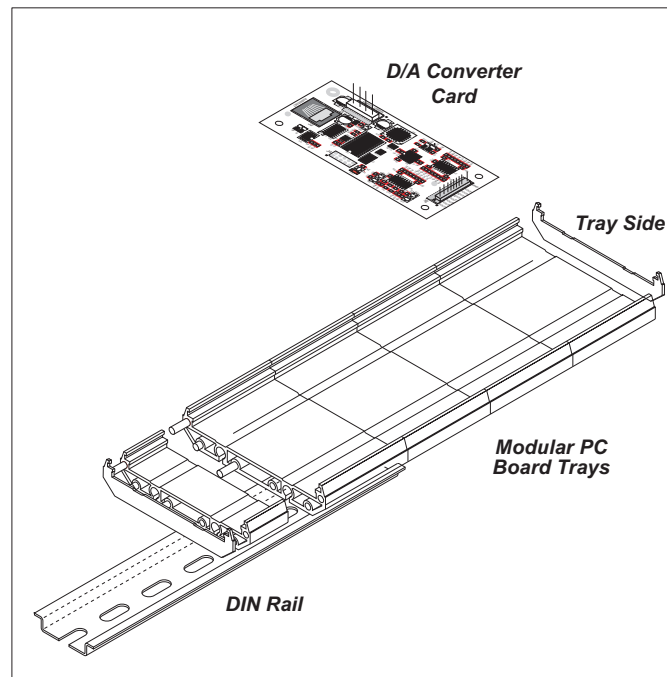
Z-World also has available additional tools and parts to allow you to make your own wiring assemblies to interface with the friction-lock connectors on the D/A Converter Card.

- Connectivity Kit (Z-World Part No. 101-0581)—Six 1 × 10 friction-lock connectors (0.1" pitch) with sixty 0.1" crimp terminals; and two 1 × 4 friction-lock connectors (0.156" pitch) and two 1 × 2 friction-lock connectors (0.156" pitch) with fifteen 0.156" crimp terminals. Each kit contains sufficient parts to interface with at least two D/A 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 D/A Converter Card may be mounted in 100 mm DIN rail trays as shown in Figure 2.



**Figure 2. Mounting D/A 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 D/A 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

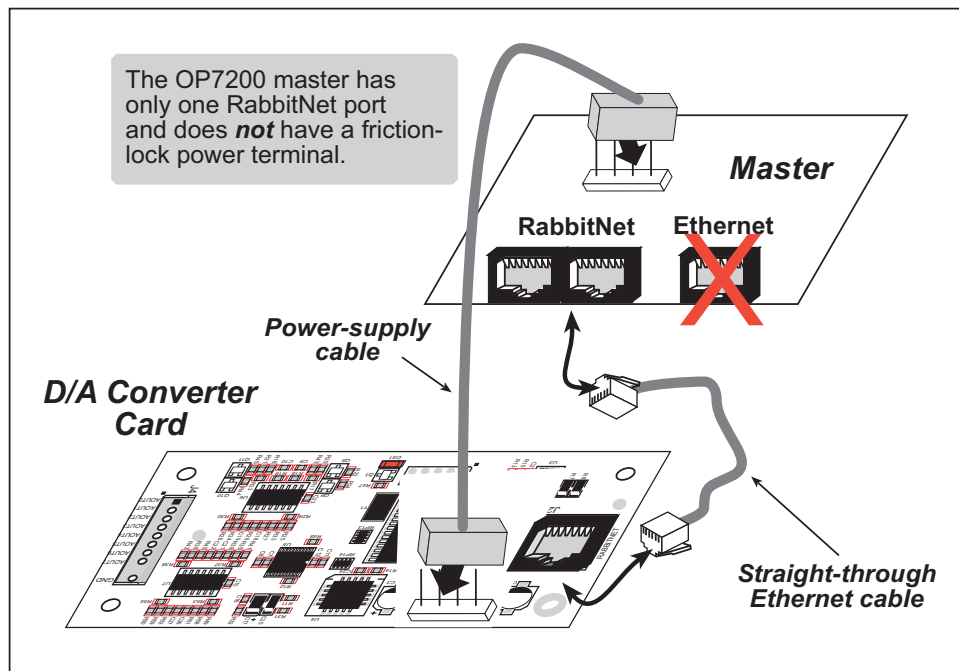
\* Length of DIN rail tray in cm

**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 D/A 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 D/A 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 D/A Converter Card to Master**

You will also have to provide two separate DC power supplies to your D/A Converter Card: +5 V and a DCIN of 9–32 V. These power supplies are connected via the polarized friction-lock terminal at header J1. You may assemble a suitable cable using the friction-lock connectors from the Connectivity Kit described in Section 1.1.2. If you are using a BL2500 as your master, you may draw this power from the BL2500 as shown in Figure 3.

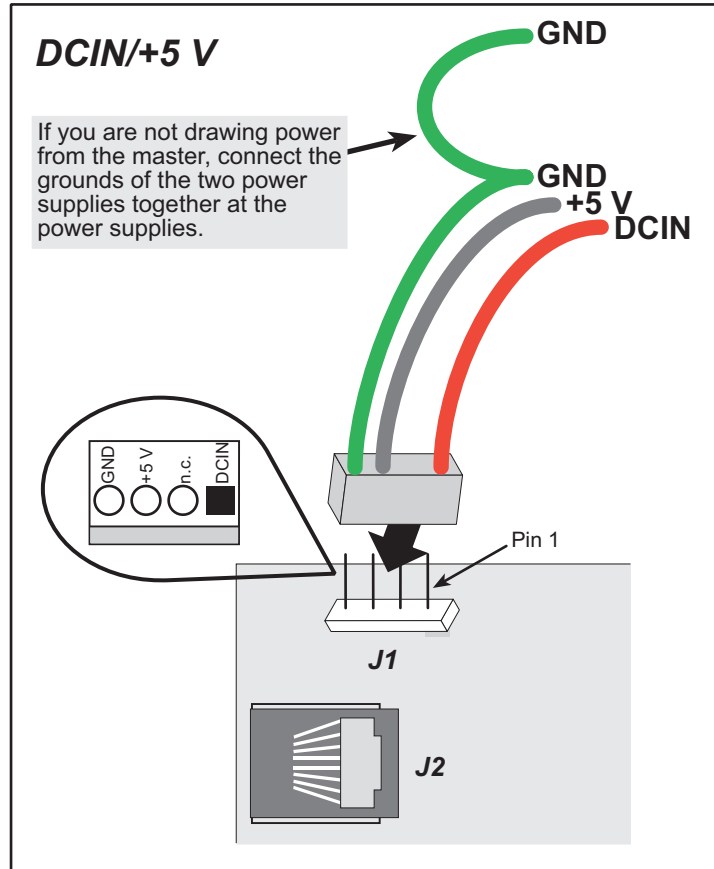
When selecting DCIN, note that DCIN must be at least 3 V more than the voltage of the D/A converter output. For example, if AOUT0 is configured for 0–20 V, DCIN must be at least 23 V, otherwise the maximum D/A converter output will be  $DCIN - 3$  V.

**NOTE:** Even if you are not drawing power from a master, you will need to connect the D/A Converter Card ground to the ground on your master. The GND pin on header J3 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 D/A Converter Card.



**Figure 4. Power-Supply Connections**

## 1.3 Pinout

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

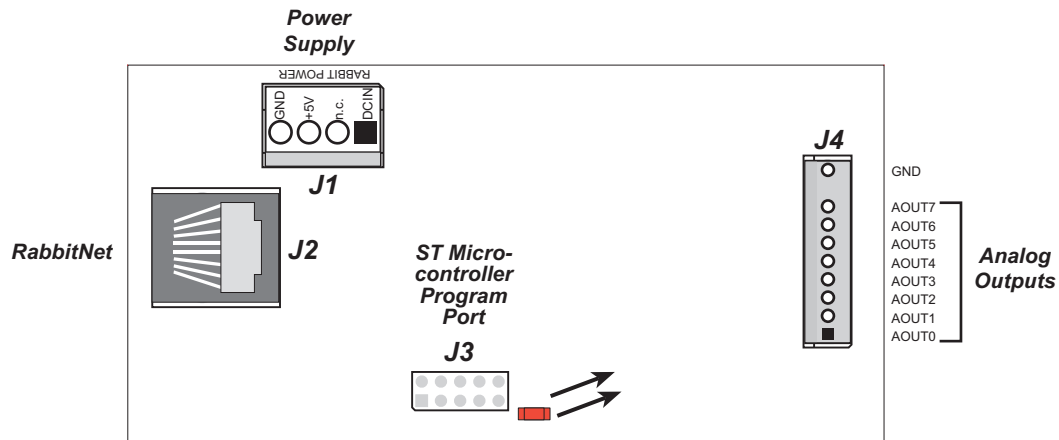


Figure 5. D/A Converter Card Pinouts

### 1.3.1 Headers

D/A Converter Cards are equipped with one polarized  $1 \times 9$  friction-lock terminals at J4, a  $1 \times 4$  friction-lock terminal at J1 (DCIN and +5 V power supplies), and an RJ-45 RabbitNet jack.

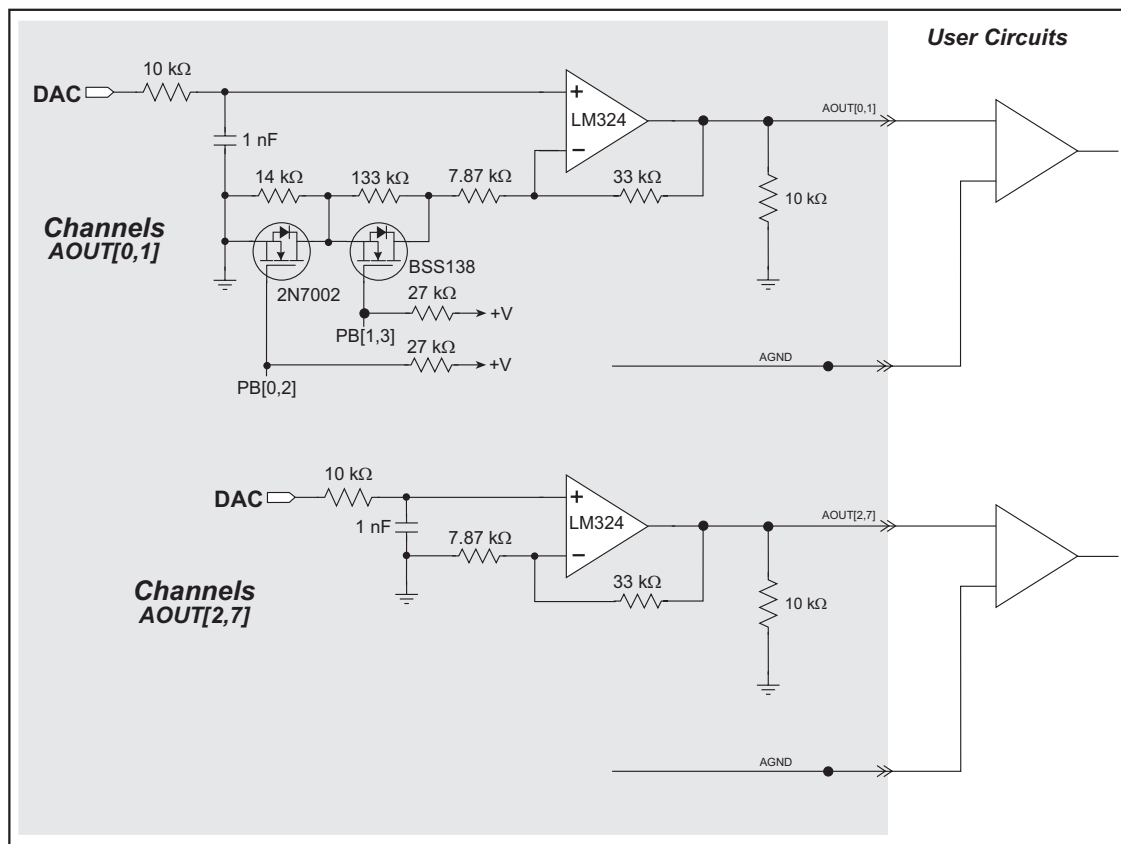
No header is installed at J3, which is used to program the D/A Converter Card at the factory.

### 1.3.2 Indicator LED

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

## 1.4 D/A Converter Outputs

Figure 6 shows the D/A converter outputs.



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

The D/A converter outputs are buffered and scaled to provide outputs that are software-configurable for voltage ranges of 0–10 V and 0–20 V. Channels AOUT0 and AOUT1 can also be configured in software to provide output ranges of 0–5 V.

**NOTE:** The D/A converter output voltage depends on the original power-supply voltage, DCIN, which must be at least 3 V more than the voltage of the D/A converter output. For example, if AOUT0 is configured for 0–20 V, DCIN must be at least 23 V, otherwise the maximum D/A converter output will be DCIN – 3 V.

While each D/A converter channel can output up to 10 mA, the total power dissipation by the LM324 op-amp at any instant must be kept below 400 mW, or (400 mW)/(DCIN × 4) mA per channel.

The D/A Converter Card outputs can be updated asynchronously as raw data are processed and written, or they may be updated simultaneously by executing the `rn_anaOutStrobe` function call with the `opmode` parameter in the `rn_anaOutConfig` function call set for synchronous operation. Further details are provided in Section 2.3.1, “Analog Output Function Calls.”

### 1.4.1 Calibration

The D/A converter outputs are factory-calibrated for the 0–10 V output range, and typical calibration constants are stored in the flash memory for the other voltage ranges. You may recalibrate the D/A converter outputs at a later time using the `rn_anaOutCalib` software function described in Section 2.3.1, “Analog Output Function Calls.”

The calibration constants are stored in flash memory in a table form. When you recalibrate your D/A Converter Card, only the calibration constants related to the the voltage range you recalibrated will be overwritten.

The `DAC_CAL.C` sample program illustrates how to perform the calibration and save the calibration data. The sample program is found in the in the `SAMPLES\RABBIT-NET\RN1300` directory. See Section 2.2, “Sample Programs,” for more information on sample programs and how to use them.

## 2. D/A 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 D/A 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 D/A 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\_AOUT.LIB**—provides functions unique to the analog inputs on the D/A Converter Card and the D/A 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 cards are described in the manual specific to the peripheral card.

### 2.1.1 Accessing and Downloading Dynamic C Libraries

The libraries needed to run the D/A 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 **RabbitNetExpansionCards.exe** 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 D/A 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 D/A 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 D/A Converter Inputs

The `SAMPLES\RABBITNET\RN1300` subdirectory contains the following sample programs.

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

**DAC\_ASYNC.C**—This sample program outputs a voltage that can be read with a voltmeter. The output voltage is calculated using the calibration constants located on the D/A Converter Card EEPROM (simulated in flash memory).

The D/A Converter Card is set up for the asynchronous mode of operation, which updates a D/A converter output at the time it is being accessed via the `anaOutVolts` or `anaOut` functions. (i.e., the `anaOutStrobe` function is not used to update the D/A converter outputs).

The sample program **DAC\_SYNC.C** illustrates the synchronous mode of operation.

**NOTE:** This sample program must be compiled to flash.

**DAC\_SYNC.C**—This sample program outputs a voltage that can be read with a voltmeter. The output voltage is calculated using the calibration constants located on the D/A Converter Card EEPROM (simulated in flash memory).

The D/A Converter Card is set up for the synchronous mode of operation, which updates all D/A converter outputs at the same time when the `anaOutStrobe` function executes. The outputs are all updated with values previously written using the `anaOutVolts` and/or `anaOut` functions.

**NOTE:** This sample program must be compiled to flash.

**DAC\_CAL.C**—This program demonstrates how to recalibrate a D/A converter channel using two known voltages, and defines the two coefficients, gain, and offset, that will be rewritten into the D/A converter card's EEPROM (simulated in flash memory).

This program will first look for a device using `rn_find()` and the product RN1300 as the search criteria, and will use the first D/A Converter Card found.

**NOTE:** The calibration constants set at the factory will be overwritten when you run this sample program.

## 2.3 D/A Converter Card Function Calls

### 2.3.1 Analog Output Function Calls

```
int rn_anaOutConfig(int handle, char config,  
int opmode, int reserved);
```

Configures the D/A Converter Card to the desired voltage range. Once the D/A Converter Card has been configured, use `rn_anaOut`, `rn_anaOutVolts`, and `rn_anaOutStrobe` to control the D/A Converter Card outputs.

#### PARAMETERS

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

`config` is a configuration code used to set the voltage ranges for the D/A Converter Card channels

Configuration Code	Voltage Ranges	
	Channels 0–1	Channels 2–7
0	0–2.5 V	0–10 V
1	0–5 V	0–10 V
2*	0–10 V	0–10 V
3	0–5 V	0–20 V
4	0–10 V	0–20 V
5	0–20 V	0–20 V

\* Default setting after reset

`opmode` is the mode of operation

0 = asynchronous—outputs are updated at the time raw data are written (default mode after reset)

1 = synchronous—outputs are updated when `rn_anaOutStrobe` is executed

`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 D/A Converter Card is not connected to the master.

#### SEE ALSO

`rn_anaOut`, `rn_anaOutVolts`, `rn_anaOutStrobe`

```
int rn_anaOut(int handle, int channel, int rawdata,  
int reserved);
```

Sets an analog output channel to a voltage that corresponds to the raw data value.

If the D/A Converter Card is set to the asynchronous mode of operation (the default mode), the output channel will be updated at the time the raw data are being written.

If the D/A Converter Card is set to the synchronous mode of operation, all the D/A converter outputs will be updated with the raw data values previously written (or default value of zero) when the **rn\_anaOutStrobe** function is executed.

The voltage range of the D/A converter outputs will be 0–10 V (default) or one of the other voltage range options previously set with the **rn\_anaInConfig** function.

Configuration Code	Voltage Ranges	
	Channels 0–1	Channels 2–7
0	0–2.5 V	0–10 V
1	0–5 V	0–10 V
2*	0–10 V	0–10 V
3	0–5 V	0–20 V
4	0–10 V	0–20 V
5	0–20 V	0–20 V

\* Default setting after reset

#### 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 AOUT0–AOUT7

**rawdata** is the raw-data value (2 bytes)

**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 D/A Converter Card is not connected to the master.

#### SEE ALSO

**rn\_anaOutConfig**, **rn\_anaOutVolts**, **rn\_anaOutStrobe**

```
int rn_anaOutVolts(int handle, int channel,
float voltage, struct rn_dacCalTable *tables,
int reserved);
```

Sets an analog output channel to a voltage using previously set calibration constants to obtain the desired voltage. Remember to run the **rn\_anaOutRdCalib** function before executing this function so the calibration table will contain valid data. Here's an example.

```
for(channel=0; channel < 8; channel++) {
    rn_anaOutRdCalib(device0, channel, &DacCalTable1, 0);
}
```

If the D/A Converter Card is set to the asynchronous mode of operation (the default mode), the output channel will be updated at the time the raw data are being written.

If the D/A Converter Card is set to the synchronous mode of operation, all the D/A converter outputs will be updated with the raw data values previously written (or default value of zero) when the **rn\_anaOutStrobe** function is executed.

The voltage range of the D/A converter outputs will be 0–10 V (default) or one of the other voltage range options previously set with the **rn\_anaInConfig** function.

Configuration Code	Voltage Ranges	
	Channels 0–1	Channels 2–7
0	0–2.5 V	0–10 V
1	0–5 V	0–10 V
2*	0–10 V	0–10 V
3	0–5 V	0–20 V
4	0–10 V	0–20 V
5	0–20 V	0–20 V

\* Default setting after reset

## 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 AOUT0–AOUT7

**voltage** is the desired output voltage, which must be less than or equal to the maximum voltage in the voltage range specified by the **rn\_anaInConfig** function

**rn\_dacCalTable \*tables** is a pointer to a table structure that contains the calibration constants for channels 0–7 for the selected mode

**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 D/A Converter Card is not connected to the master.

## SEE ALSO

**rn\_anaOutConfig**, **rn\_anaOut**, **rn\_anaOutStrobe**

```
int rn_anaOutStrobe(int handle, int reserved);
```

Strobes the D/A Converter Card to update all the D/A converter outputs with the raw data values previously written (or a default value of zero).

**NOTE:** This function is only valid if the D/A Converter Card is set to the synchronous mode of operation using the **rn\_anaOutConfig** function call.

#### **PARAMETERS**

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

**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 D/A Converter Card is not connected to the master.

#### **SEE ALSO**

**rn\_anaOutConfig, rn\_anaOut, rn\_anaOutVolts**

```
int rn_anaOutCalib(int channel, int value1,  
float volts1, int value2, float volts2,  
DacCal *table, int reserved);
```

Calibrates the response of the desired analog output channel as a linear function using the two conversion points provided. Values are calculated and the results are sent to the analog output device using the function **anaOutWrCalib**.

Each channel will have the following information:

- linear constant or gain
- 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 in the device were corrupted.

#### PARAMETERS

**channel** is the channel number (0 to 7) corresponding to AOUT0–AOUT7

**value1** is the first raw analog output value (0–4095)

**volts1** is the voltage corresponding to the first output value

**value2** is the second raw analog output value (0–4095)

**volts2** is the voltage corresponding to the second output value

**DacCal \*table** is a pointer to a table structure that contains the calibration constants

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

#### RETURN VALUE

0, if successful.

-1 if not able to make calibration constants.

#### SEE ALSO

**rn\_anaOutWrCalib**, **rn\_anaOutRdCalib**

```
int rn_anaOutWrCalib(int handle, int channel,  
    DacCal *table, int reserved);
```

Writes the calibration constants, gain, and offset previously calculated by `rn_anaOutCalib()` into the device flash memory.

**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 in the device were corrupted.

#### 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 AOUT0–AOUT7

**DacCal \*table** is a pointer to a table structure that contains the calibration constants

**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 D/A Converter Card is not connected to the master.

#### SEE ALSO

`rn_anaOutRdCalib`, `rn_anaOutCalib`

```
int rn_anaOutRdCalib(int handle, int channel,  
    DacCal *table, int reserved);
```

Reads the calibration constants, gain, and offset into a calibration descriptor table **rn\_dacCalTable**.

#### **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 AOUT0–AOUT7

**DacCal \*table** is a pointer to a table structure that contains the calibration constants

**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 D/A Converter Card is not connected to the master.

#### **SEE ALSO**

**rn\_anaOutWrCalib, rn\_anaOutCalib**

### 2.3.2 Status Byte

Unless otherwise specified, functions returning a status byte for D/A 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†

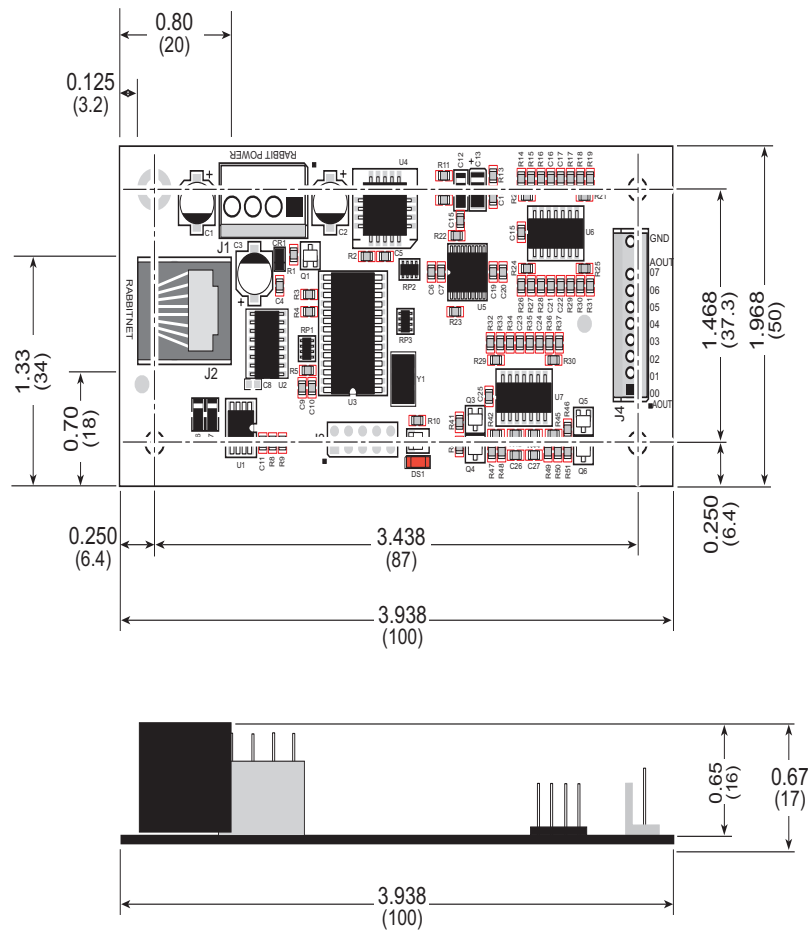
\* Use the function `rn_comm_status()` to determine which error occurred.

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

# APPENDIX A. D/A CONVERTER CARD SPECIFICATIONS

## A.1 Electrical and Mechanical Specifications

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



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

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

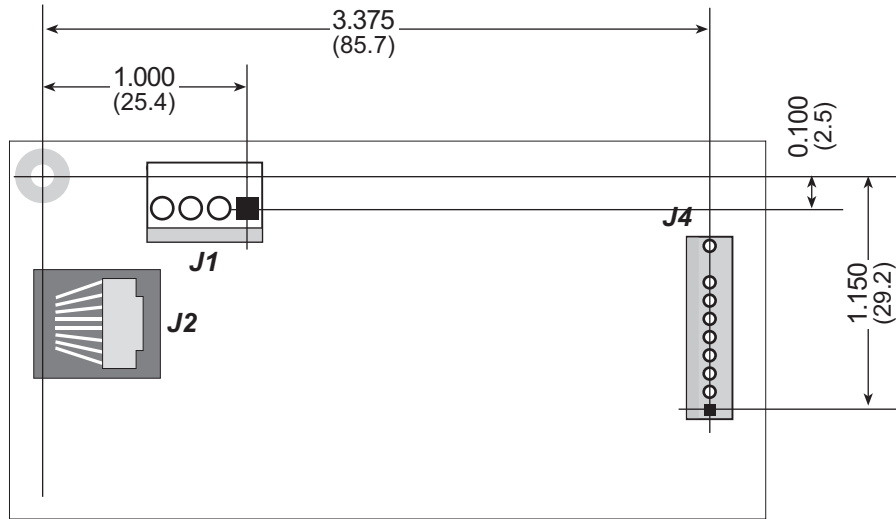
Table A-1 lists the electrical, mechanical, and environmental specifications for the D/A Converter Card.

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

Feature	Specification
Analog Outputs	8 channels of 12-bit analog outputs, 8 $\Omega$ output impedance, 2.5 kHz update rate <ul style="list-style-type: none"> <li>• software-controlled output-voltage ranges:                0–2.5 V, 0–5 V, 0–10 V, 0–20 V DC (channels AOUT0–AOUT1)                0–10 V, 0–20 V DC (channels AOUT2–AOUT7)</li> </ul>
RabbitNet™ Serial Port	RS-422 SPI, 1 Mbits/s
Power	Vcc: +5 V DC, 20 mA DCIN: 9–32 V, 100 mA
Temperature	–40°C to +85°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" × 3.94" × 0.67" (50 mm × 100 mm × 17 mm)

### A.1.1 Physical Mounting

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



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



# 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.

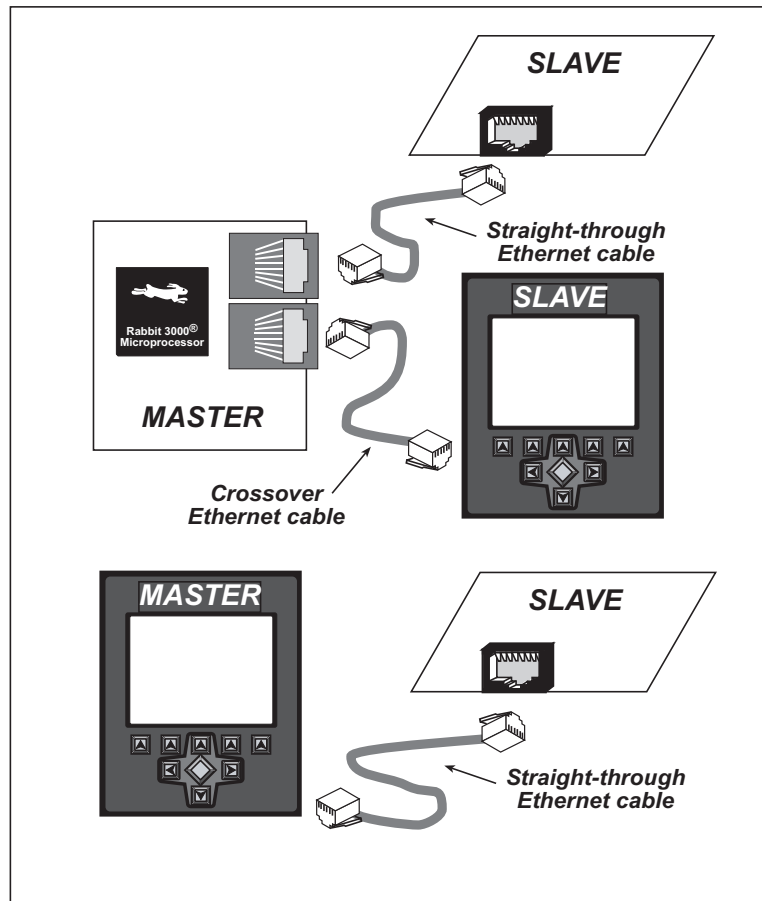


Figure B-1. Connecting Peripheral Cards to a Master

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 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. -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

## C

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

## D

D/A Converter Card  
  connection to master ..... 1, 4  
  power supplies ..... 4  
D/A converter inputs  
  calibration constants ..... 8  
dimensions  
  D/A Converter Card ..... 21  
DIN rail mounting ..... 3  
  components ..... 3  
Dynamic C ..... 2  
  downloading RabbitNet  
    libraries ..... 10  
    libraries ..... 9

## F

features ..... 2

## I

indicator LED ..... 6

## P

peripheral cards  
  connection to master ... 25, 26  
physical mounting  
  D/A Converter Card ..... 23  
pinout  
  D/A Converter headers ..... 6  
power supplies  
  D/A Converter Card ..... 4  
  wiring diagram ..... 5

## R

RabbitNet  
  Ethernet cables to connect  
    peripheral cards ..... 25, 26  
  general description ..... 25  
  peripheral cards ..... 26  
  physical implementation ... 27

## S

sample programs ..... 11  
  calibration constants  
    DAC\_CAL.C ..... 8  
  D/A converter outputs  
    DAC\_ASYNC.C ..... 12  
    DAC\_CAL.C ..... 12  
    DAC\_SYNC.C ..... 12  
  RabbitNet operation  
    ECHOCHAR.C ..... 11  
    ECHOTERM.C ..... 11  
    HWATCHDOG.C ..... 11  
    SWATCHDOG.C ..... 11  
software ..... 2, 9  
  analog output functions ..... 13  
    rn\_anaOut ..... 14  
    rn\_anaOutCalib ..... 17  
    rn\_anaOutConfig ..... 13  
    rn\_anaOutRdCalib ..... 19  
    rn\_anaOutStrobe ..... 16  
    rn\_anaOutVolts ..... 15  
    rn\_anaOutWrCalib ..... 18  
  downloading RabbitNet  
    libraries ..... 10  
  libraries ..... 9  
  RN\_CFG\_BL25.LIB ..... 9  
  RN\_CFG\_OP72.LIB ..... 9  
  RNET.LIB ..... 9, 28  
  RNET\_AOUT.LIB ..... 9  
  RNET\_DRIVER.LIB ..... 9

## RNET.LIB

  rn\_comm\_status ..... 33  
  rn\_device ..... 11, 28  
  rn\_echo ..... 29  
  rn\_enable\_wdt ..... 32  
  rn\_find ..... 29  
  rn\_hitwd ..... 32  
  rn\_init ..... 28  
  rn\_read ..... 30  
  rn\_reset ..... 31  
  rn\_rst\_status ..... 33  
  rn\_sw\_wdt ..... 31  
  rn\_write ..... 30  
  sample programs ..... 11  
specifications  
  D/A Converter Card ..... 21  
  dimensions ..... 21  
  electrical ..... 22  
  header footprint ..... 23  
  physical mounting ..... 23  
  relative pin 1 locations .. 23  
  temperature ..... 22  
status byte ..... 34





# SCHEMATICS

## **090-0179 D/A Converter Card Schematic**

[www.zworld.com/documentation/schemat/090-0179.pdf](http://www.zworld.com/documentation/schemat/090-0179.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.

