

# **Relay Card (RN1400)**

RabbitNet LAN Card

## **User's Manual**

019-0140 • 040202-A

## **RN1400 User's Manual**

Part Number 019-0140 • 040202-A • Printed in U.S.A.

©2004 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

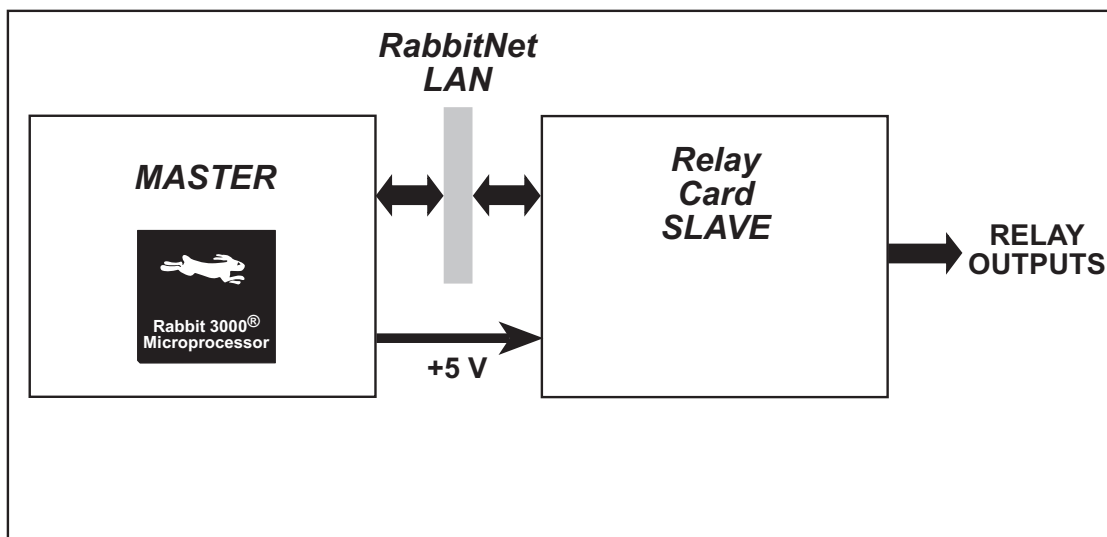
<b>Chapter 1. Overview</b>	<b>1</b>
1.1 Relay Card Features	2
1.1.1 Software	2
1.1.2 Connectivity Tools	2
1.1.3 DIN Rail Mounting	3
1.2 Connecting Peripheral Cards	4
1.2.1 Power-Supply Connections	5
1.3 Pinout	7
1.3.1 Headers	7
1.4 Relay Outputs	8
<b>Chapter 2. Relay Cards Software</b>	<b>9</b>
2.1 Dynamic 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 Relay Outputs	12
2.3 Relay Card Function Calls	13
2.3.1 Status Byte	16
<b>Appendix A. Relay Card Specifications</b>	<b>17</b>
A.1 Electrical and Mechanical Specifications	17
A.1.1 Physical Mounting	19
<b>Appendix B. RabbitNet</b>	<b>21</b>
B.1 General RabbitNet Description	21
B.2 Physical Implementation	23
B.2.1 Control and Routing	23
B.3 Function Calls	24
B.3.1 Status Byte	30
<b>Notice to Users</b>	<b>31</b>
<b>Index</b>	<b>33</b>
<b>Schematics</b>	<b>35</b>



# 1. OVERVIEW

Chapter 1 describes the features and the use of the Relay 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 cards. Figure 1 shows a conceptual view of the Relay Card connected to a master.



*Figure 1. Relay Card (Slave) Connected to Master*

## 1.1 Relay Card Features

- 6 SPDT relays rated at 250 V AC, 1200 V·A (100 V DC up to 240 W) with built-in snubbers
- 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, can be up to 10 m (33 ft) away from master

### 1.1.1 Software

The Relay 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

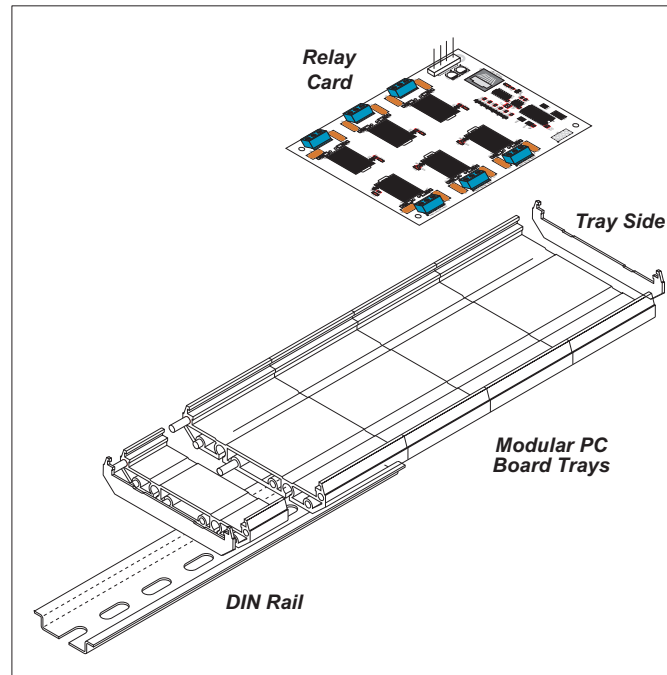
Z-World also has available additional tools and parts to allow you to make your own wiring assemblies to interface with the friction-lock connectors on the Relay Card.

- 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 two Relay Cards (only eight 0.156" crimp terminals are actually used).
- 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 Relay Card may be mounted in 100 mm DIN rail trays as shown in Figure 2.



**Figure 2. Mounting Digital I/O 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 Relay 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

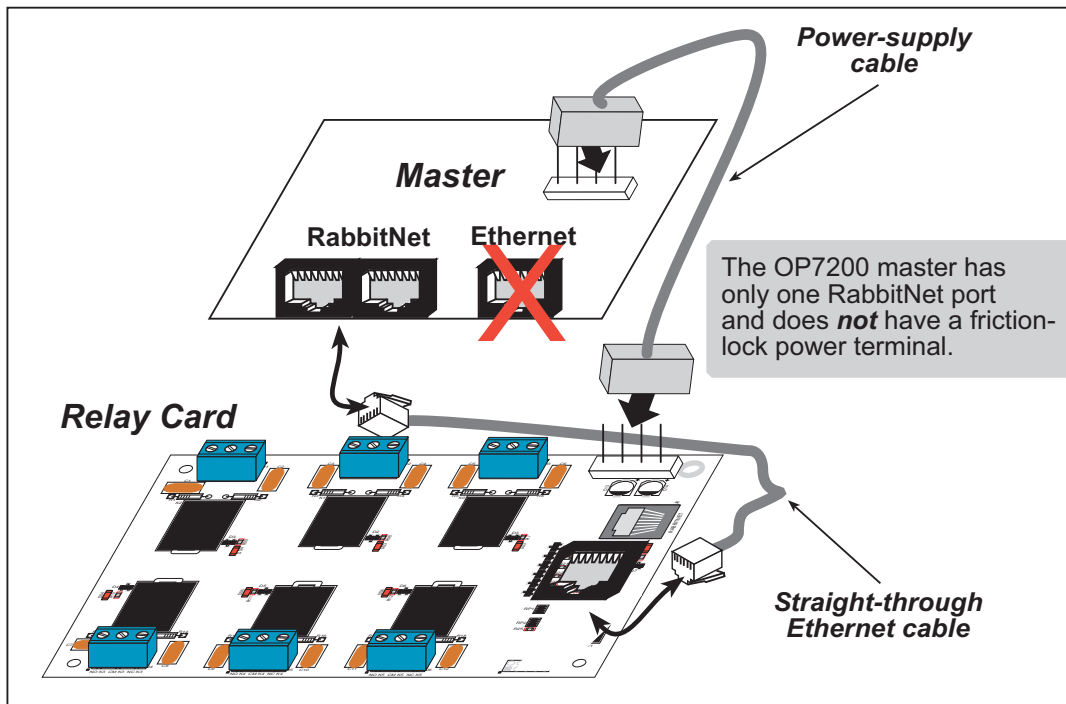
\* Length of DIN rail tray in cm

**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 Relay Card. 108 mm plastic trays may be used with spacers.

## 1.2 Connecting Peripheral Cards

Use a straight-through Ethernet cable to connect the Relay 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 cards. The *RabbitNet* jacks do not support Ethernet connections.



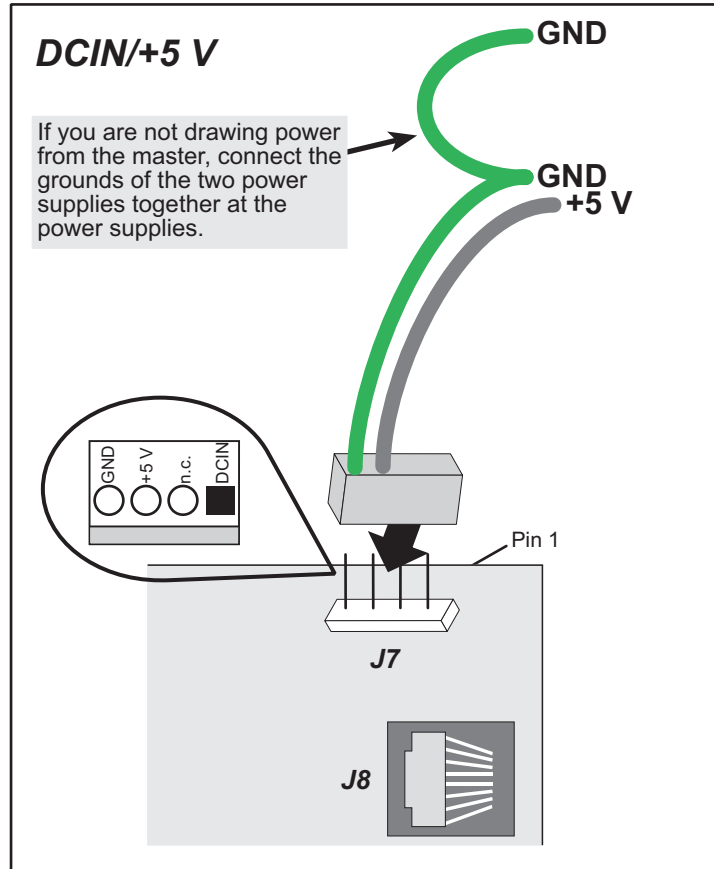
**Figure 3. Connect Digital I/O Card to Master**

You will also have to provide a separate +5 V DC power supply to your Relay Card. This power supply is connected via the polarized friction-lock terminal at header J7. 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. See Section 1.2.1 for detailed wiring diagrams.

At the present time, the number of peripheral cards you can use with one master is limited by the number of RabbitNet ports on the master. A router is being developed to allow additional peripheral cards 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

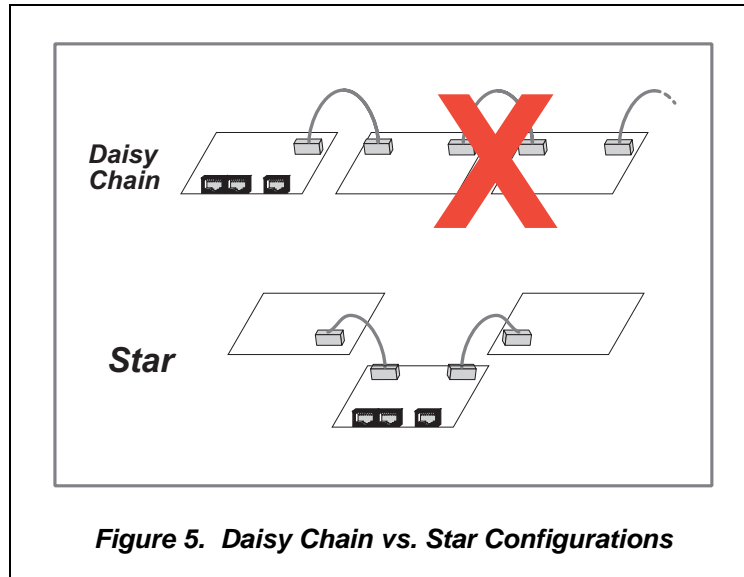
Figure 4 illustrates the assembled friction-lock connector wiring diagram for the power supplies used to supply power to the Relay Card.



**Figure 4. Power-Supply Connections**

**NOTE:** If you are using a separate DC power supply for +5 V to the Relay Card because you are not drawing this power from the master, note that the crimp pins used in the friction-lock connector assembly can only hold one wire each. Connect the one GND wire from the friction-lock connector assembly to the ground on one of the two power supplies, then use a separate wire to connect the power-supply grounds together.

Use 18-gauge (AWG) wire (1 mm<sup>2</sup>) for power-supply connections up to 10 m away from the master or router. If the wire length is less than 3 m, 22 gauge (AWG) wire (0.4 mm<sup>2</sup>) is acceptable. Do not daisy-chain the power supply connections between different peripheral cards, but use a star configuration from the master or router when there are several peripheral cards.

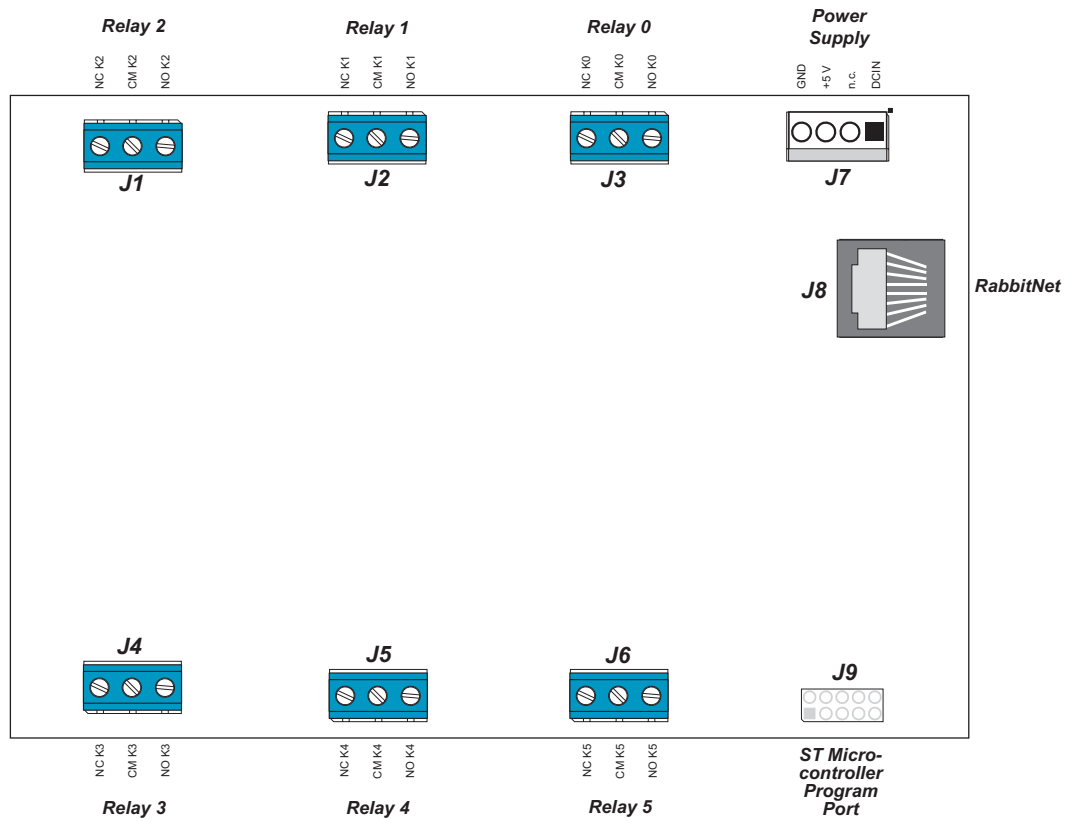


It is best to use a type of cable where the wires for the ground and positive(s) of any power supply are bound together or twisted, and ideally the power-supply wires should not be bundled with other wires.

Large transient currents flow in the ground and positive supply wires when the relay output drivers are switched on/off, and it is imperative that any ground differential resulting from resistive or inductive loss in the ground wire be kept as low as possible (<4 V). Use the GND pin on header J7 on the Relay Card if you have separate power supplies. Z-World also recommends that you have a physical ground connection between the Relay Card and the master, which you will have if the power to header J7 on the Relay Card already comes from the master.

## 1.3 Pinout

The Relay Card pinouts are shown in Figure 6.



**Figure 6. Relay Card Pinouts**

### 1.3.1 Headers

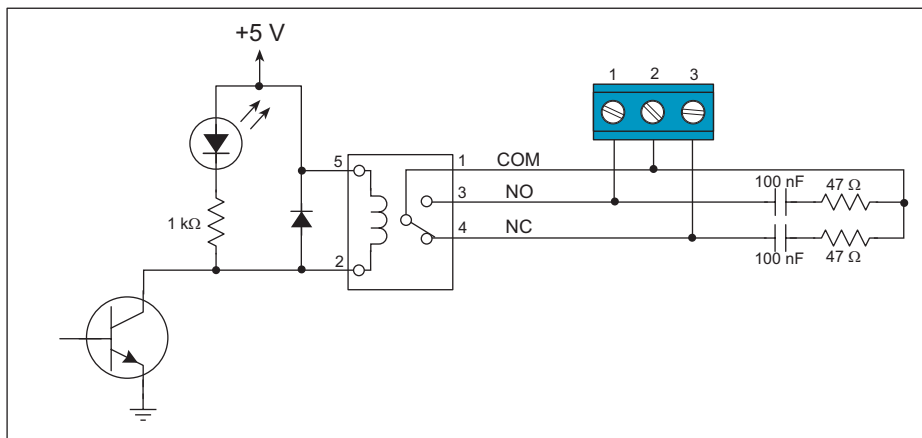
Relay Cards are equipped with six screw-terminal headers (J1–J6), a 1 × 4 friction-lock terminal (J7—DCIN and +5 V power supplies), and an RJ-45 RabbitNet jack.

No header is installed at J9, which is used to program the Relay Card at the factory.

## 1.4 Relay Outputs

The Relay Card has six SPDT relays, each of which is rated to handle up to 250 V AC, 1200 V·A (max. 10 A) or up to 100 V, DC 240 W (max. 5 A). Each relay draws approximately 83 mA from the +5 V power supply when energized. This current draw can be reduced by approximately a factor of two by using the `rn_RelayPwr` function call to engage the power-save mode once a relay is energized.

Figure 7 illustrates one of the six relay output circuits. An LED is associated with each relay, and is on while the relay is energized.



**Figure 7. Relay Output Circuit**



**CAUTION:** Voltages up to 250 V AC may be present on the screw-terminal headers. Exercise appropriate care when handling a wired Relay Card.

Since a wired Relay Card is likely to be installed as part of an assembly inside an enclosure, Z-World recommends that appropriate warning labels be placed on the enclosure to alert the end-user of the high-voltage hazard and to refer any repairs or maintenance to a qualified service technician.

Each relay has built-in snubbers, which consist of a resistor and a capacitor in parallel with the contacts to reduce arcing. Although the original role of the snubbers was to preserve the life of the relay contacts by reducing arcing, snubbers are particularly beneficial in circuits driving inductive loads, where they limit voltage transients and reduce electromagnetic interference.

Depending on the reactive load you plan to operate with the Relay Card, you may want to change the resistor and capacitor values used for the built-in snubber circuit. This can be done easily since all the resistors and capacitors used in the snubber circuits are through-hole parts.

## 2. RELAY CARDS 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 Relay 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 Relay Card.

- **RN\_CFG\_BL25.LIB**—used to configure the BL2500 for use with RabbitNet peripheral cards. 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 cards. 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\_RELAY.LIB**—provides functions unique to the Relay 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 Relay 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 Relay 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 cards. 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 card 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 Relay 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 cards connected to the master. The last peripheral card found will run the sample program. The sample program will also display the serial number(s) of the peripheral cards connected to which RabbitNet port on the master using the **STDIO** window, or that no card is connected to a particular port.

- **ECHOCHAR.C**—Demonstrates a simple character echo to any RabbitNet peripheral card. A character is sent to the RabbitNet peripheral card connected at a physical node address of 0x00 or 000 octal. If a peripheral card is connected, the character will be returned back along with the status of the peripheral card. Otherwise, the status byte will indicate there is no connection.
- **ECHOTERM.C**—Demonstrates a simple character echo to any RabbitNet peripheral card through a serial terminal on the master. A character is sent to the RabbitNet peripheral card connected at a physical-node address of 0x00 or 000 octal. If a card is connected, the character will be returned back along with the status of the peripheral card. Otherwise, the status byte will indicate there is no connection.
- **HWATCHDOG.C**—Demonstrates setting the hardware watchdog on a RabbitNet peripheral card. This sample program will first look for a peripheral card 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 card using costatements. This program will first look for a peripheral card 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 Relay Outputs

The `SAMPLES\RABBITNET\RN1400` subdirectory contains the following sample programs. When running these sample programs, the Relay 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_find()` and the product RN1400 as the search criteria to first find any Relay Cards connected to the master. The first Relay Card found will run the sample program.

- **RELAY\_ALL.C**—Demonstrates how to activate all the relays in parallel using the `rn_RelayAll` function call.



**CAUTION:** Activating several relays in a short period of time may cause a power surge that may exceed the peak power rating of your power supply. Be sure that your power supply can handle at least 500 mA when using this sample program.

- **RELAY\_LOW\_PWR.C**—Demonstrates how to configure the relays to operate in the power-save mode. A relay is first activated normally for 50 ms, and is then pulsed every millisecond with a 50% duty-cycle square wave, which essentially cuts the power required to keep the relay energized in half. Since the operation of a relay in the power-save mode will reduce the relay-holding force, this mode is not recommended when the relay may be subject to shock and vibration.
- **RELAY\_SEQUENCE.C**—Demonstrates writing values to a bank of outputs by using the Demonstration Board whose LEDs are toggled ON/OFF via the outputs.

## 2.3 Relay Card Function Calls



**CAUTION:** Activating several relays in a short period of time may cause a power surge that may exceed the peak power rating of your power supply. It is ultimately the responsibility of the application designer to ensure that the power supply meets the requirements for the intended application.

Also note that the power-save mode will reduce the holding force for the relay contacts. Z-World recommends that you *not* use the power-save mode when the Relay Card is expected to be subject to shock and vibration.

```
int rn_Relay(int handle, int relay, int value,  
             int reserved);
```

Sets the state of a given relay by connecting the relay common contact to either the relay normally closed contact or to the relay normally open contact.

### PARAMETERS

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

**relay** is the selected relay (0–5).

**value** is used to set a given relay connection as follows:

0 = common connected to normally closed contact

1 = common connected to normally open contact

**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 Relay Card is not connected to the master.

### SEE ALSO

`rnRelayAll`, `rnRelayPwr`

```
int rn_RelayAll(int handle, int control,  
int reserved);
```

Sets the state of all the relays with the given bitwise control value. Connects the relay common contact to either the relay normally closed contact or to the relay normally open contact.

#### PARAMETERS

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

**control** establishes the bitwise control of Relays 0–5. The bit positions 0–5 correspond directly to Relays 0–5, with the bit value controlling the relay as follows:

0 = common connected to normally closed contact

1 = common connected to normally open contact

**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 Relay Card is not connected to the master.

#### SEE ALSO

**rnRelayPwr**, **rnRelay**

#### EXAMPLE

```
rn_RelayAll(handle, 0x05, 0);  
  
// Sets the relays to have the following connections:  
Relay0...Common connected to Normally Open contact  
Relay1...Common connected to Normally Closed contact  
Relay2...Common connected to Normally Open contact  
Relay3...Common connected to Normally Closed contact  
Relay4...Common connected to Normally Closed contact  
Relay5...Common connected to Normally-Closed contact
```

```
void rn_RelayPwr(int handle, int control,  
int reserved);
```

Sets the specified relays to be in a power reduction/save mode. The power-save mode is activated after the relay has been active for at least 50 ms, after which the relay will be pulsed every 1 ms with a 50% duty cycle square wave, which should provide a power reduction of 50% for the given relay.

If this function isn't called, the relays will operate without going into the power-save mode of operation.

#### PARAMETERS

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

**control** establishes the bitwise control of Relays 0–5. The bit positions 0–5 correspond directly to Relays 0–5, with the bit value controlling the relay as follows:

0 = set relay for normal operation

1 = set relay for power-save mode

**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 Relay Card is not connected to the master.

#### SEE ALSO

`rnRelayAll`, `rnRelay`

#### EXAMPLE

```
rn_RelayPwr(handle, 0x05, 0);  
  
// Sets the relays for the following operation:  
Relay0.....Set to power-save mode  
Relay1.....Set for normal operation  
Relay2.....Set to power-save mode  
Relay3.....Set for normal operation  
Relay4.....Set for normal operation  
Relay5.....Set for normal operation
```

### 2.3.1 Status Byte

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

7	6	5	4	3	2	1	0	
×	×							00 = Reserved 01 = Ready 10 = Busy 11 = Device not connected
		×						0 = Device 1 = Router
			×					0 = No error 1 = Communication error*
				×				0 = No update occurred 1 = A/D converter update occurred
					×			Reserved
						×		0 = Last command accepted 1 = Last command unexecuted
							×	0 = Not expired 1 = HW or SW watchdog timer expired†

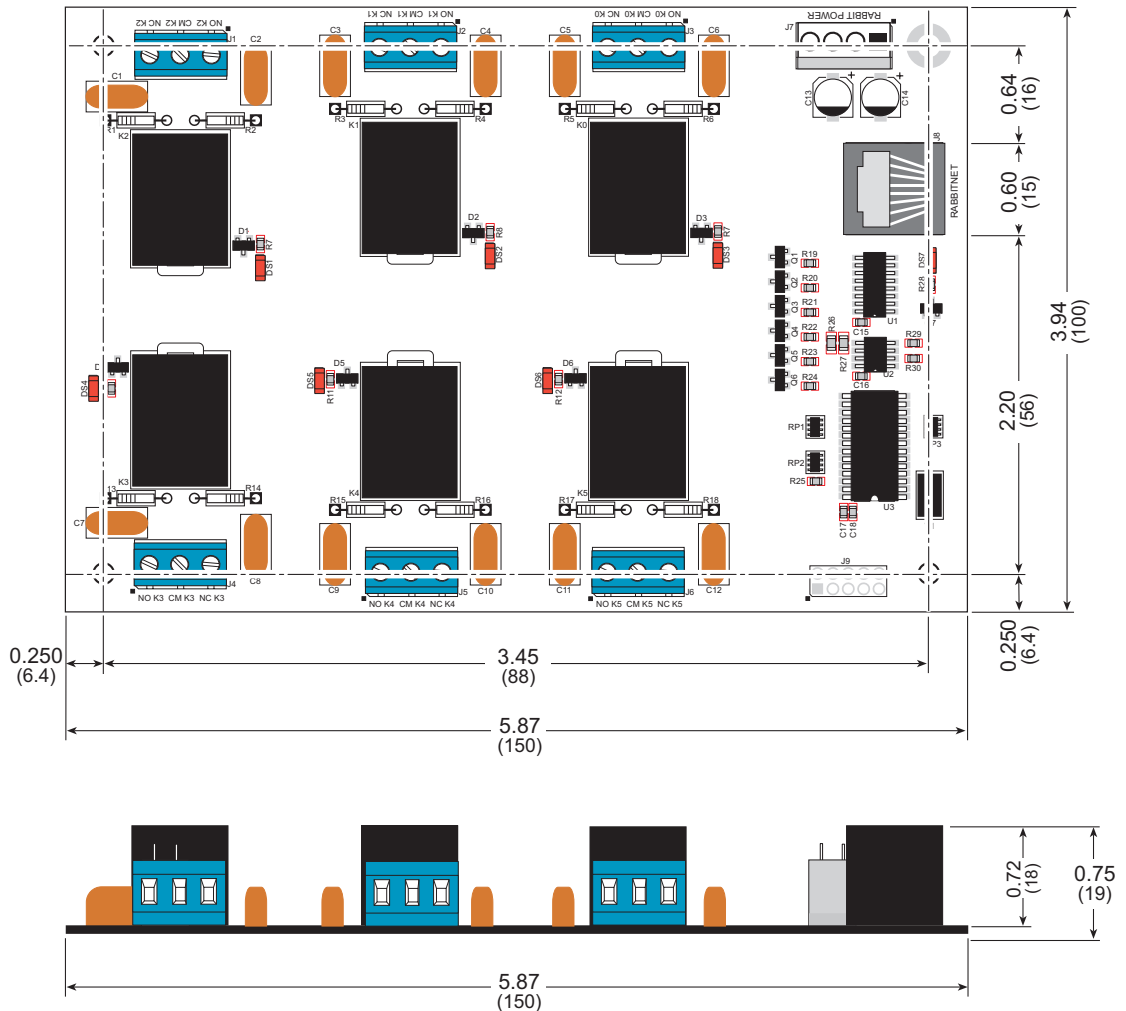
\* Use the function **rn\_comm\_status()** to determine which error occurred.

† Use the function **rn\_rst\_status()** to determine which timer expired.

# APPENDIX A. RELAY CARD SPECIFICATIONS

## A.1 Electrical and Mechanical Specifications

Figure A-1 shows the mechanical dimensions for the Relay Card.



**Figure A-1. Relay 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 Relay Card.

**Table A-1. Relay Card Specifications**

Feature	Specification
Microprocessor	ST72F264G
Relay Outputs	Six SPDT relays with snubbers: <ul style="list-style-type: none"> <li>• max. contact settling time: 10 ms</li> <li>• max. switching voltage: 250 V AC, 100 V DC</li> <li>• max. switching current: 10 A AC, 5 A DC</li> <li>• max. switching capability: 1200 V·A AC, 240 W DC</li> <li>• snubbers: built-in 47 <math>\Omega</math>, 100 nF</li> <li>• terminal wire gauge: #14 AWG (1.628 mm dia.) max.</li> </ul>
RabbitNet™ Serial Port	RS-422, 1 Mbits/s
Power	V <sub>cc</sub> : +5 V DC, 500 mA (all relays energized)
Temperature	-40°C to +70°C
Humidity	5% to 95%, noncondensing
Connectors	Six screw-terminal headers Friction-lock connectors: <ul style="list-style-type: none"> <li>• one 4-position terminal with 0.156" pitch</li> </ul> One RJ-45 RabbitNet™ jack
Board Size	3.94" × 5.87" × 0.75" (100 mm × 150 mm × 19 mm)

## A.1.1 Physical Mounting

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

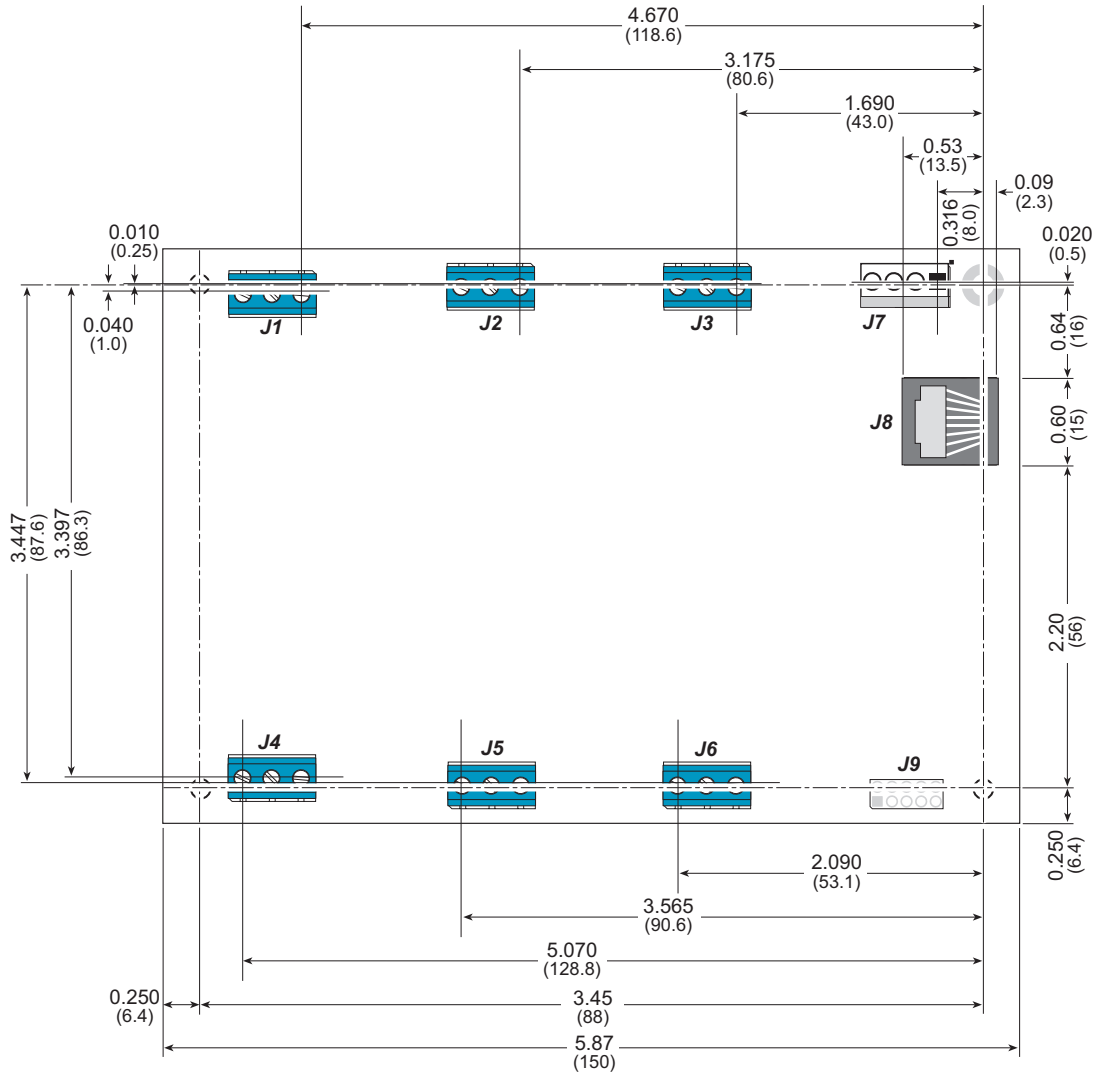


Figure A-2. User Board Footprint for Relay Card



# 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.

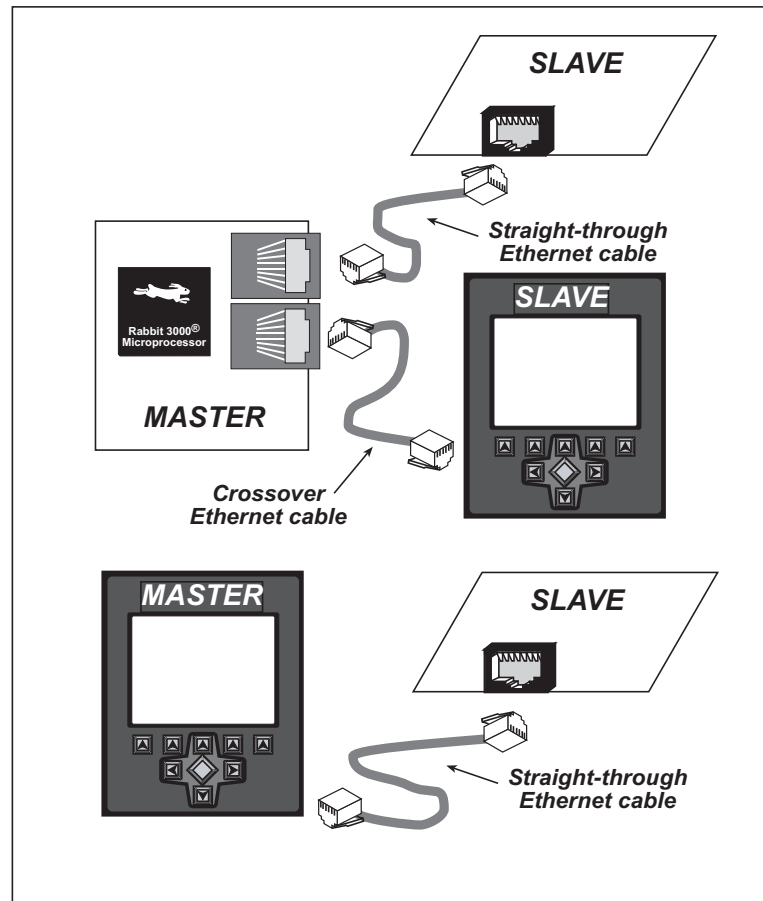


Figure B-1. Connecting Peripheral Cards to a Master

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
- Relay card
  - 6 relays rated at 250 V AC, 1200 V·A or 100 V DC up to 240 W. The following connectors are used:
    - Relay contacts = screw-terminal connectors
    - 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 10 = Busy 11 = 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

dimensions  
  Relay Card ..... 17  
DIN rail mounting ..... 3  
  components ..... 3  
Dynamic C ..... 2  
  downloading RabbitNet  
    libraries ..... 10  
  libraries ..... 9

## F

features ..... 2

## P

peripheral cards  
  connection to master ... 21, 22  
physical mounting  
  Relay Card ..... 19  
pinout  
  Relay Card headers ..... 7  
power supplies  
  Relay Card ..... 4  
  wiring diagram ..... 5

## R

RabbitNet  
  Ethernet cables to connect  
    peripheral cards ..... 21, 22  
  general description ..... 21  
  peripheral cards ..... 22  
  physical implementation ... 23  
Relay Card  
  connection to master ..... 1, 4  
  power supplies ..... 4  
  relay output ..... 8

## S

sample programs ..... 11  
  RabbitNet operation  
    ECHOCHAR.C ..... 11  
    ECHOTERM.C ..... 11  
    HWATCHDOG.C ..... 11  
    SWATCHDOG.C ..... 11  
  relay outputs  
    RELAY\_ALL.C ..... 12  
    RELAY\_LOW\_PWR.C 12  
    RELAY\_SEQUENCE.C 12  
  software ..... 2, 9  
    downloading RabbitNet  
      libraries ..... 10  
  libraries ..... 9  
    RN\_CFG\_BL25.LIB ..... 9  
    RN\_CFG\_OP72.LIB ..... 9  
    RNET.LIB ..... 9, 24  
    RNET\_DRIVER.LIB ..... 9  
    RNET\_RELAY.LIB ..... 9

relayoutputs  
  m\_Relay ..... 13  
  m\_RelayAll ..... 14  
  m\_RelayPwr ..... 8, 15  
RNET.LIB  
  m\_comm\_status ..... 29  
  m\_device ..... 11, 24  
  m\_echo ..... 25  
  m\_enable\_wdt ..... 28  
  m\_find ..... 12, 25  
  m\_hitwd ..... 28  
  m\_init ..... 24  
  m\_read ..... 26  
  m\_reset ..... 27  
  m\_rst\_status ..... 29  
  m\_sw\_wdt ..... 27  
  m\_write ..... 26  
  sample programs ..... 11  
specifications  
  Relay Card ..... 17  
    dimensions ..... 17  
    electrical ..... 18  
    header footprint ..... 19  
    physical mounting ..... 19  
    relative pin 1 locations .. 19  
    temperature ..... 18  
  status byte ..... 30  
  Relay Card ..... 16





# SCHEMATICS

## **090-0184 Relay Card Schematic**

[www.zworld.com/documentation/schemat/090-0184.pdf](http://www.zworld.com/documentation/schemat/090-0184.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.

