Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

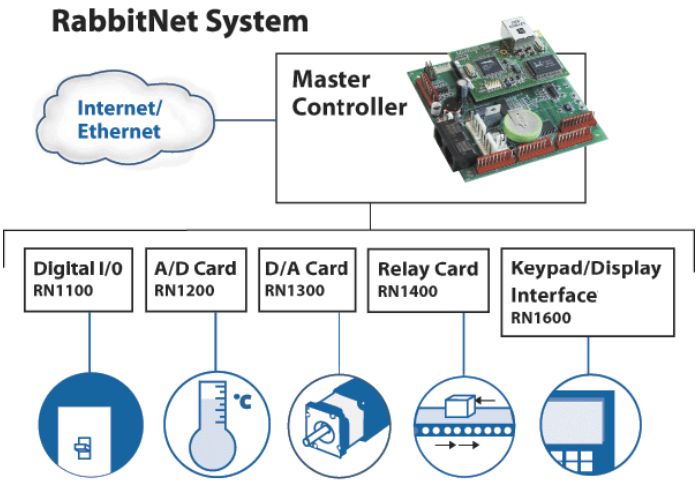
With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

PRODUCT MANUAL



RabbitNet Peripheral Cards

RabbitNet LAN Cards

User's Manual

019–0146 • 070629–D

RabbitNet Peripheral Cards User's Manual

Part Number 019-0146 • 07029–D • Printed in U.S.A.

Trademarks

Rabbit and Dynamic C are registered trademarks of Rabbit Semiconductor Inc.

RabbitNet is a trademark of Rabbit Semiconductor Inc.

The latest revision of this manual is available on the Rabbit Semiconductor Web site, www.rabbit.com, for free, unregistered download.

Rabbit Semiconductor Inc.

www.rabbit.com

TABLE OF CONTENTS

Chapter 1. The RabbitNet Protocol 1

1.1 General RabbitNet Description....1
1.1.1 RabbitNet Connections....1
1.1.2 RabbitNet Peripheral Cards....2
1.1.3 Connectivity Tools....3
1.1.4 DIN Rail Mounting....4
1.2 Physical Implementation....5
1.2.1 Control and Routing....5
1.3 Dynamic C....6
1.3.1 Dynamic C Libraries....6
1.3.1.1 Accessing and Downloading Dynamic C Libraries....7
1.3.2 Sample Programs....8
1.3.3 General RabbitNet Operation....8
1.3.4 General RabbitNet Function Calls....9
1.3.5 Status Byte....15

Chapter 2. Digital I/O Card 17

2.1 Features....18
2.1.1 Software....18
2.2 Connections....19
2.2.1 Power Supply....20
2.3 Pinout....22
2.3.1 Headers....22
2.3.2 Indicator LED....22
2.4 Digital I/O....23
2.4.1 Digital Inputs....23
2.4.2 Digital Outputs....24
2.5 Analog Inputs....26
2.5.1 Single-Ended Inputs....27
2.5.2 Differential Inputs....27
2.5.3 Calibrating the Analog Inputs....27
2.5.3.1 Calibration Constants....28
2.5.3.2 Calibration Recommendations....28
2.5.3.3 Factory Calibration....29
2.6 Software....30
2.6.1 Dynamic C Libraries....30
2.6.2 Sample Programs....30
2.6.2.1 Digital I/O....30
2.6.2.2 Analog Inputs....32
2.6.3 Digital I/O Card Function Calls....34
2.6.3.1 Digital Input Function Calls....34
2.6.3.2 Digital Output Function Calls....35
2.6.3.3 Analog Input Function Calls....37
2.6.4 Status Byte....44
2.7 Specifications....45
2.7.1 Electrical and Mechanical Specifications....45
2.7.1.1 Physical Mounting....47
2.7.2 Jumper Configurations....48

Chapter 3. A/D Converter Card 51
3.1 Features ... 52
3.1.1 Software ... 52
3.2 Connections ... 53
3.2.1 Power Supply ... 54
3.3 Pinout ... 55
3.3.1 Headers ... 55
3.3.2 Indicator LED ... 55
3.4 Analog Inputs ... 56
3.4.1 Analog Current Measurements ... 58
3.4.2 Calibrating the A/D Converter Chip ... 59
3.4.2.1 Modes ... 59
3.4.2.2 Calibration Constants ... 59
3.4.2.3 Calibration Recommendations ... 60
3.4.2.4 Factory Calibration ... 61
3.5 Software ... 62
3.5.1 Dynamic C Libraries ... 62
3.5.2 Sample Programs ... 62
3.5.3 A/D Converter Card Function Calls ... 65
3.5.4 Status Byte ... 77
3.6 Specifications ... 78
3.6.1 Electrical and Mechanical Specifications ... 78
3.6.2 Physical Mounting ... 80
3.7 Jumper Configurations ... 81

Chapter 4. D/A Converter Card 83
4.1 Features ... 84
4.1.1 Software ... 84
4.2 Connections ... 85
4.2.1 Power Supply ... 86
4.3 Pinout ... 87
4.3.1 Headers ... 87
4.3.2 Indicator LED ... 87
4.4 D/A Converter Outputs ... 88
4.4.1 Calibration ... 89
4.5 Software ... 90
4.5.1 Dynamic C Libraries ... 90
4.5.2 Sample Programs ... 90
4.5.3 D/A Converter Card Function Calls ... 92
4.5.4 Status Byte ... 99
4.6 Specifications ... 100
4.6.1 Electrical and Mechanical Specifications ... 100
4.6.2 Physical Mounting ... 102

Chapter 5. Relay Card 103
5.1 Features ... 104
5.1.1 Software ... 104
5.2 Connections ... 105
5.2.1 Power Supply ... 106
5.3 Pinout ... 108
5.3.1 Headers ... 108
5.3.2 Indicator LEDs ... 108
5.4 Relay Outputs ... 109
5.5 Software ... 110
5.5.1 Dynamic C Libraries ... 110
5.5.2 Sample Programs ... 110
5.5.3 Relay Card Function Calls ... 112
5.5.4 Status Byte ... 115

5.6 Specifications 116
5.6.1 Electrical and Mechanical Specifications 116
5.6.2 Physical Mounting 118

Chapter 6. Keypad/Display Interface 119
6.1 Features 120
6.1.1 Software 120
6.2 Connections 121
6.2.1 Power Supply 122
6.3 Key RabbitNet Keypad/Display Interface Components 123
6.3.1 Headers and Jacks 123
6.3.1.1 Keypads 123
6.3.1.2 Liquid Crystal Displays 124
6.3.2 LEDs 124
6.3.3 Buzzer 124
6.4 Liquid Crystal Display Backlights 125
6.5 Display Contrast 127
6.6 Software 128
6.6.1 Dynamic C Libraries 128
6.6.2 Sample Programs 128
6.6.3 RabbitNet Keypad/Display interface Function Calls 130
6.6.3.1 Buzzer 130
6.6.3.2 LEDs 131
6.6.3.3 Keypad 132
6.6.3.4 Display 135
6.6.4 Status Byte 141
6.7 Specifications 142
6.8 Electrical and Mechanical Specifications 142
6.8.1 Physical Mounting 144

Appendix A. Keypad/Display Interface Expansion Kit 145
A.1 Keypads 146
A.1.1 Keypad Templates 147
A.2 LCD Displays 150
A.2.1 2 × 20 Character LCD 151
A.2.2 4 × 20 Character LCD 151
A.3 ZMENU.C 152
A.4 Configuring Key Code Indexes and Physical Keypad Arrangement 156
A.4.1 Basics of Assigning Key Code Indexes 156
A.4.2 Expansion Kit Keypads 158
A.4.2.1 3 × 4 Keypad 158
A.4.2.2 2 × 6 Keypad 159
A.4.2.3 4 × 10 Keypad 160
A.5 2 × 6 Keypad Datasheet 161
A.6 3 × 4 Keypad Datasheet 162
A.7 4 × 10 Keypad Datasheet 163
A.8 2 × 20 Character LCD Datasheet 164
A.9 4 × 20 Character LCD Datasheet 188

Index 195

Schematics 199

1. THE RABBITNET PROTOCOL

1.1 General RabbitNet Description

RabbitNet is a high-speed synchronous protocol developed by Rabbit Semiconductor to connect peripheral cards to a master and to allow them to communicate with each other.

1.1.1 RabbitNet Connections

All RabbitNet connections are made point to point. 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.

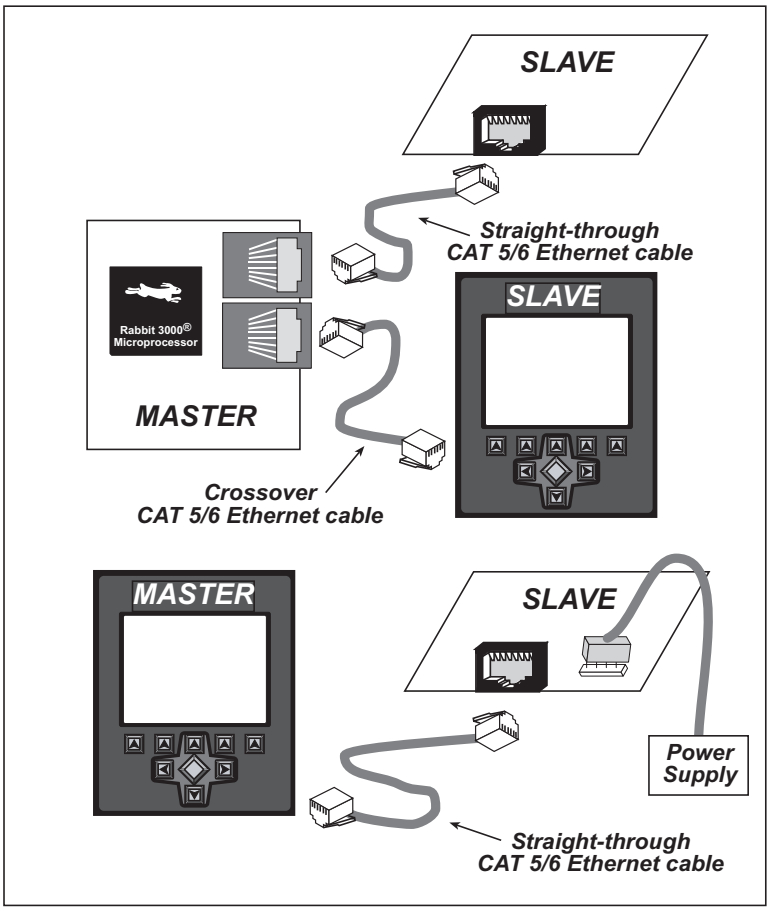


Figure 1. Connecting Peripheral Cards to a Master

A typical RabbitNet™ system consists of a master single-board computer and one or two peripheral cards. A high-performance Rabbit 3000® or Rabbit 2000® microprocessor on the master provides fast data processing, and a BL2500 or a BL2600 master also provides the DCIN and +5 V power for the peripheral cards. Use a straight-through CAT 5/6 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 CAT 5/6 Ethernet cable to connect an OP7200 that is being used as a slave.

NOTE: Even though CAT 5/6 Ethernet cables are used for the RabbitNet connections, ***never*** connect a RabbitNet port to an Ethernet network. Doing so could destroy the RabbitNet SPI driver.

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

Table 1 lists Rabbit Semiconductor's single-board computers and other devices that can be used as the master in a RabbitNet system.

Table 1. RabbitNet Master Capabilities

RabbitNet Masters	Master Supplies Power to Peripheral Cards	Number of RabbitNet Ports
BL2500	Yes	2
BL2600	Yes	2
OP7200	No	1
RCM3300/RCM3360 Prototyping Board	No	1
PowerCore FLEX Prototyping Board	No	1

1.1.2 RabbitNet Peripheral Cards

- Digital I/O Card

  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 Card

  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 Card

  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 interface

  Allows you to connect your own keypad with up to 64 keys and one character liquid crystal display from 1 × 8 to 4 × 20 characters with or without backlight using standard 1 × 16 or 2 × 8 connectors. The following connectors are used:

  Signal = 0.1" headers or sockets
  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

The OP7200 operator interface may serve as a RabbitNet peripheral card "display" in a RabbitNet system.

Visit our 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.

1.1.3 Connectivity Tools

Rabbit Semiconductor 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 RabbitNet peripheral cards.

- Connectivity Kit (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 one or more RabbitNet peripheral cards.
- Crimp tool (Part No. 998-0013) to secure wire in crimp terminals.

Contact your authorized Rabbit Semiconductor distributor or your Rabbit Semiconductor Sales Representative for more information.

1.1.4 DIN Rail Mounting

RabbitNet peripheral cards and the BL2500 master may be mounted in 100 mm DIN rail trays as shown in Figure 2.

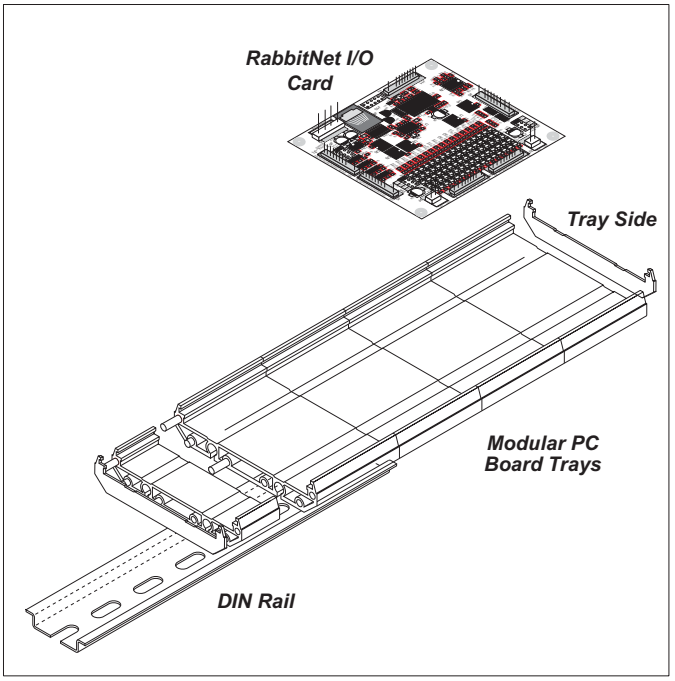


Figure 2. Mounting RabbitNet Peripheral Card in DIN Rail Trays

DIN rail trays are typically mounted on DIN rails with "feet." Table 2 lists Phoenix Contact part numbers for the DIN rail trays, rails, and feet. The tray side elements are used to keep the RabbitNet peripheral 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 2. 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 a RabbitNet peripheral card. 108 mm plastic trays may be used with spacers.

1.2 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, Rabbit Semiconductor recommends a practical limit of 10 m (33 ft).

Connections between peripheral cards and masters are done using standard 8-conductor CAT 5/6 Ethernet cables. Masters and peripheral cards are equipped with RJ-45 8-pin female connectors. The cables may be swapped end for end without affecting functionality.

1.2.1 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.

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. 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.

1.3 Dynamic C

Dynamic C is an integrated development system for writing embedded software. It runs on an IBM-compatible PC and is designed for use with Rabbit Semiconductor single-board computers and other devices based on the Rabbit microprocessor.

1.3.1 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 RabbitNet peripheral cards.

- **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_BL26.LIB**—used to configure the BL2600 for use with RabbitNet peripheral cards. Function calls for this library are discussed in the ***Wolf (BL2600) 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***.
- **RN_CFG_PowerCoreFLEX.LIB**—used to configure the PowerCore FLEX modules for use with RabbitNet peripheral boards on the PowerCore FLEX Prototyping Board. Function calls for this library are discussed in the ***PowerCore FLEX User's Manual***.
- **RN_CFG_RCM33.LIB**—used to configure the RCM3300, RCM3310, RCM3360, and RCM3370 for use with RabbitNet peripheral boards on the RCM3300 Prototyping Board. Function calls for this library are discussed in the ***RCM3300/RCM3310 User's Manual*** and in the ***RCM3360/RCM3370 User's Manual***.
- **RNET.LIB**—provides functions unique to the RabbitNet protocol. Function calls for this library are presented below.
- **RNET_DRIVER.LIB**—provides background functions unique to the RabbitNet data transmission protocol.

Function calls specific to individual RabbitNet peripheral cards are described in the chapters specific to the individual RabbitNet peripheral card. Other functions applicable to all devices based on Rabbit microprocessors are described in the ***Dynamic C Function Reference User's Manual***. More complete information on Dynamic C is provided in the ***Dynamic C User's Manual***.

1.3.1.1 Accessing and Downloading Dynamic C Libraries

The libraries needed to run the RabbitNet peripheral cards are available on the CD included with the Development Kit for the master single-board computer, or they may be downloaded from http://www.rabbit.com/support/downloads/ on Rabbit Semiconductor'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 RabbitNet peripheral card and you will continue to be able to use this installation with all the other Rabbit Semiconductor products you used before.

1.3.2 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), then compile and run it by pressing **F9** or 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***.

1.3.3 General RabbitNet Operation

The **SAMPLES\RABBITNET** subdirectory contains the following sample programs. When running these sample programs, the RabbitNet peripheral 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 0000 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 0000 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.

1.3.4 General RabbitNet 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

pna 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 pna                // physical node address
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. -1 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 *recdata,
   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.

recdata 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 wdttype);

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.

wdttype

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.

1.3.5 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.

2. DIGITAL I/O CARD

Chapter 2 describes the features and the use of the Digital I/O Card, one of the peripheral cards designed for use with the RabbitNet expansion ports on selected Rabbit Semiconductor single-board computers, operator interfaces, and RabbitCore Prototyping Boards.

Figure 3 shows a conceptual view of the Digital I/O Card connected to a master.

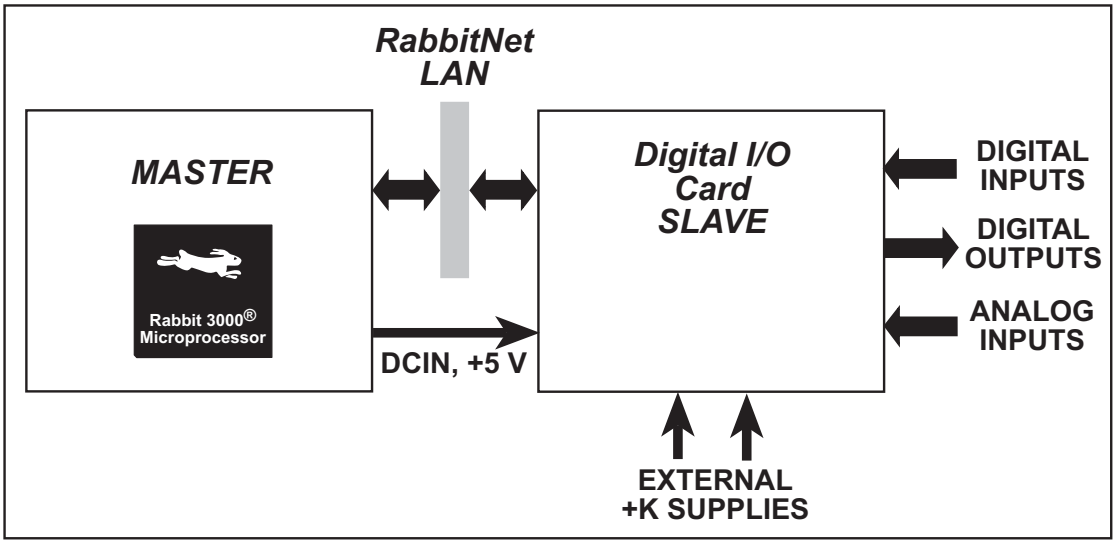


Figure 3. Digital I/O Card (Slave) Connected to Master

NOTE: The OP7200 master and the RabbitCore Prototyping Boards do ***not*** supply any power to the slave.

2.1 Features

- 24 protected and filtered digital inputs
- 16 high-speed protected digital outputs, individually configurable as sinking or sourcing up to 200 mA at up to 36 V DC
- four 10-bit analog input channels:

  2 buffered, 0 – 10 V, single-ended

  1 buffered, 0 – 1 V, single-ended

  1 buffered, -0.25 – +0.25 V, differential
- can be mounted in standard 100 mm DIN rail trays sold by other suppliers
- interfaces with master through RabbitNet™ serial protocol at 1 Megabit per second using standard CAT 5/6 Ethernet cable, can be up to 10 m (33 ft) away from master

2.1.1 Software

The Digital I/O Card is a slave; the master to which it is connected is programmed using version 8.01 or later of Rabbit Semiconductor's Dynamic C. If you are using a master with an earlier version of Dynamic C, Rabbit Semiconductor recommends that you upgrade your Dynamic C installation. Contact your authorized Rabbit Semiconductor distributor or your Rabbit Semiconductor Sales Representative for more information on Dynamic C upgrades.

2.2 Connections

Use a straight-through CAT 5/6 Ethernet cable to connect the Digital I/O 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.

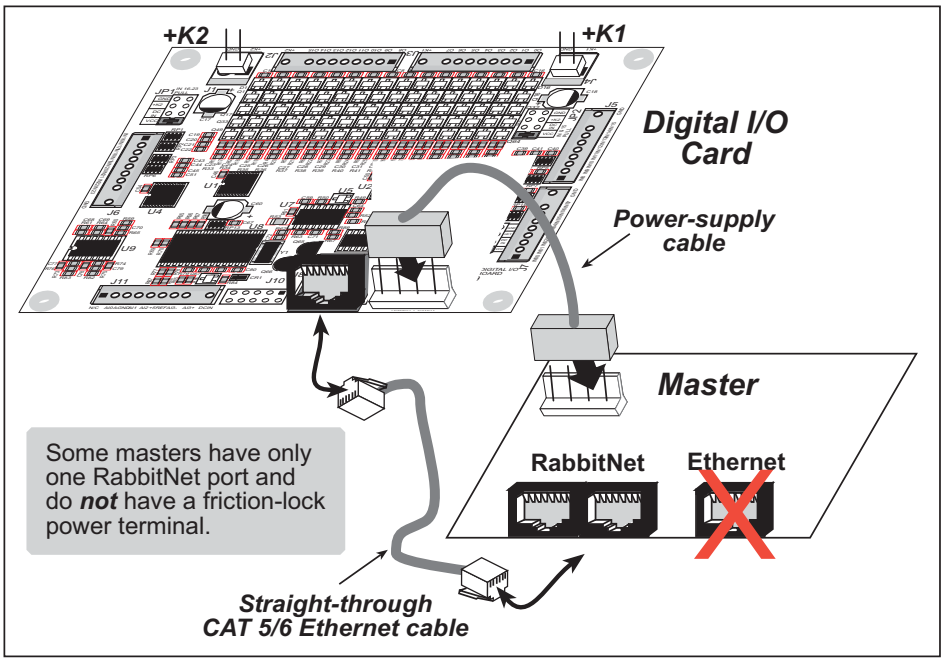


Figure 4. Connect Digital I/O Card to Master

You will also have to provide two separate DC power supplies to your Digital I/O Card: +5 V and a DCIN of 9–32 V. These power supplies are connected via the polarized friction-lock terminal at header J9. You may assemble a suitable cable using the friction-lock connectors from the Connectivity Kit described in Section 1.1.3. If you are using a BL2500 or BL2600 as your master, you may draw this power from the BL2500 or BL2600 as shown in Figure 4.

If you are using the digital outputs, you will need two additional external power supplies up to 36 V that can each handle up to 1.6 A for +K1 and +K2. The actual voltage and current depend on the requirements of the loads you plan to connect to the digital outputs. These power supplies are connected to friction-lock terminals J1 and J4 on the Digital I/O Card. You may assemble suitable cables using the friction-lock connectors from the Connectivity Kit described in Section 1.1.3. See Section 2.2.1 for detailed wiring diagrams.