Welcome to E-XFL.COM

Understanding Embedded - Microcontroller, Microprocessor, FPGA Modules

Embedded - Microcontroller, Microprocessor, and FPGA Modules are fundamental components in modern electronic systems, offering a wide range of functionalities and capabilities. Microcontrollers are compact integrated circuits designed to execute specific control tasks within an embedded system. They typically include a processor, memory, and input/output peripherals on a single chip. Microprocessors, on the other hand, are more powerful processing units used in complex computing tasks, often requiring external memory and peripherals. FPGAs (Field Programmable Gate Arrays) are highly flexible devices that can be configured by the user to perform specific logic functions, making them invaluable in applications requiring customization and adaptability.

Applications of Embedded - Microcontroller,

Details	
Product Status	Obsolete
Module/Board Type	MPU Core
Core Processor	Rabbit 3000
Co-Processor	-
Speed	44.2MHz
Flash Size	512KB (Internal), 32MB (External)
RAM Size	1MB
Connector Type	2 IDC Headers 2x17, 1 IDC Header 2x5. 1 xD-Picture Card
Size / Dimension	1.85" x 2.73" (47mm x 69mm)
Operating Temperature	0°C ~ 70°C
Purchase URL	https://www.e-xfl.com/product-detail/digi-international/101-1051

RabbitCore RCM3365/RCM3375 User's Manual

Part Number 019-0150 • 080528–G • Printed in U.S.A.

Trademarks

Rabbit, RabbitCore, and Dynamic C are registered trademarks of Digi International Inc.

Rabbit 3000 is a trademark of Digi International Inc.

xD-Picture Card is a trademark of Fuji Photo Film Co., Olympus Corporation, and Toshiba Corporation.

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

Rabbit Semiconductor Inc.

www.rabbit.com

1.1 RCM3365 and RCM3375 Features

- Small size: 1.85" x 2.73" x 0.86"
  (47 mm x 69 mm x 22 mm)
- Microprocessor: Rabbit 3000 running at 44.2 MHz
- 52 parallel 5 V tolerant I/O lines: 44 configurable for I/O, 4 fixed inputs, 4 fixed outputs
- Three additional digital inputs, two additional digital outputs
- External reset
- Alternate I/O bus can be configured for 8 data lines and 6 address lines (shared with parallel I/O lines), plus I/O read/write
- Ten 8-bit timers (six cascadable) and one 10-bit timer with two match registers
- 512K flash memory, 512K program execution SRAM, 512K data SRAM
- Fixed and hot-swappable mass-storage flash-memory options, which may be used with the standardized directory structure supported by the Dynamic C FAT File System module.
- Real-time clock
- Watchdog supervisor
- Provision for customer-supplied backup battery via connections on header J4
- 10-bit free-running PWM counter and four pulse-width registers
- Two-channel Input Capture (shared with parallel I/O ports) can be used to time input signals from various port pins
- Two-channel Quadrature Decoder accepts inputs from external incremental encoder modules
- Five or six 3.3 V CMOS-compatible serial ports with a maximum asynchronous baud rate of 5.525 Mbps. Three ports are configurable as a clocked serial port (SPI), and two ports are configurable as SDLC/HDLC serial ports (shared with parallel I/O ports).
- Supports 1.15 Mbps IrDA transceiver
- Supports Dynamic C RabbitSys, which supports Ethernet access for remote application updates, and remote monitoring and control of a RabbitSys-enabled RCM3365

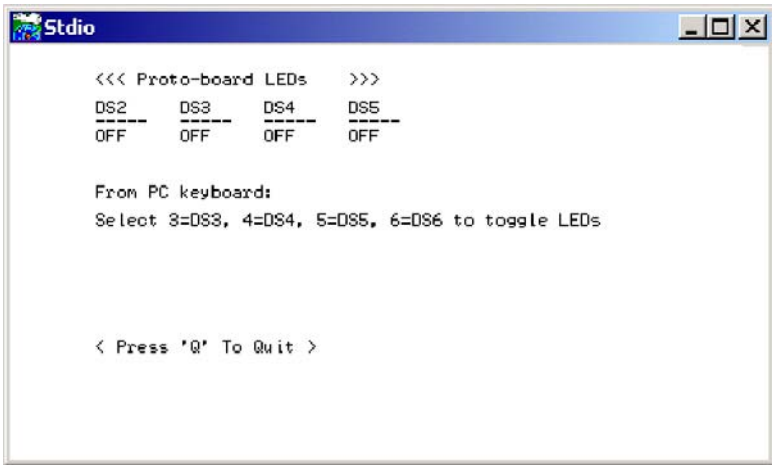
The RCM3900/RCM3910 and RCM3365/RCM3375 RabbitCore modules are similar to the RCM3305/RCM3315 and RCM3309/RCM3319, but they use fixed NAND or removable media for their mass-storage memories instead of the fixed serial flash options of the RCM3305/RCM3315 and the RCM3309/RCM3319.

3.2 Sample Programs

Of the many sample programs included with Dynamic C, several are specific to the RCM3365 and the RCM3375. Sample programs illustrating the general operation of the RCM3365/RCM3375, serial communication, and the NAND flash are provided in the **SAMPLES\RCM3360** folder. Each sample program has comments that describe the purpose and function of the program. Follow the instructions at the beginning of the sample program. Note that the RCM3365/RCM3375 must be installed on the Prototyping Board when using the sample programs described in this chapter.

- **CONTROLLED.C**—Demonstrates use of the digital inputs by having you turn the LEDs on the Prototyping Board on or off from the **STDIO** window on your PC.

  Once you compile and run **CONTROLLED.C**, the following display will appear in the Dynamic C **STDIO** window.



  Press “2” or “3” or “4”or “5”on your keyboard to select LED DS3 or DS4 or DS5 or DS6 on the Prototyping Board. Then follow the prompt in the Dynamic C **STDIO** window to turn the LED on or off.

- **FLASHLED.C**—Demonstrates assembly-language program by flashing the USR LED on the RCM3365/RCM3375 and LEDs DS3, DS4, DS5, and DS6 on the Prototyping Board.
- **SWRELAY.C**—Demonstrates the relay-switching function call using the relay installed on the Prototyping Board through screw-terminal header J17.
- **TOGGLESWITCH.C**—Uses costatements to detect switches S2 and S3 using debouncing. The corresponding LEDs (DS3 and DS4) will turn on or off.

Once you have loaded and executed these four programs and have an understanding of how Dynamic C and the RCM3365/RCM3375 modules interact, you can move on and try the other sample programs, or begin building your own.

3.2.1 Use of NAND Flash

The following sample programs can be found in the **SAMPLES\RCM3360\NANDFlash** folder. As you run most of these sample programs, you will be prompted in the Dynamic C **STDIO** window to select either the soldered-in NAND flash (RCM3365 model only) or the socketed *xD-Picture Card* (0 = soldered, 1 = socketed).

- **NFLASH_DUMP.c**—This program is a utility for dumping the nonerased contents of a NAND flash chip to the Dynamic C **STDIO** window, and the contents may be redirected to a serial port.

  When the sample program starts running, it attempts to communicate with the user-selected NAND flash chip. If this communication is successful and the main page size is acceptable, the nonerased page contents (non 0xFF) from the NAND flash page are dumped to the Dynamic C **STDIO** win.for inspection.

  Note that an error message might appear when the first 32 pages (0x20 pages) are "dumped." You may ignore the error message.

- **NFLASH_INSPECT.c**—This program is a utility for inspecting the contents of a NAND flash chip. When the sample program starts running, it attempts to communicate with the NAND flash chip selected by the user. Once a NAND flash chip is found, the user can execute various commands to print out the contents of a specified page, clear (set to zero) all the bytes in a specified page, erase (set to FF), or write to specified pages.

  CAUTION: When you run this sample program, enabling the **#define NFLASH_CANERASEBADBLOCKS** macro makes it possible to write to bad blocks. The first two blocks on the *xD-Picture Card* are marked bad to protect the configuration data needed to use the card in a digital camera or a PC. You will only be able to use the *xD-Picture Card* in Rabbit-based systems if either of the first two blocks is written to.

- **NFLASH_LOG.c**—This program runs a simple Web server and stores a log of hits in the NAND flash. As long as the *xD-Picture Card* is plugged in to its connector J6, this sample program will log hits to the *xD-Picture Card*. Remove the *xD-Picture Card* if you wish to log hits on the soldered-in NAND flash (RCM3365 model only).

  This log can be viewed and cleared from a browser by connecting the RJ-45 jack on the RCM3365 to your PC as described in Section 6.1. The sidebar on the next page explains how to set up your PC or notebook to view this log.

Figure 10 shows how to insert or remove the *xD-Picture Card*. While you remove or insert the *xD-Picture Card*, take care to avoid touching the electrical contacts on the bottom of the card to prevent electrostatic discharge damage to the card and to keep any moisture or other contaminants off the contacts. Do ***not*** remove or insert the *xD-Picture Card* while it is being accessed.

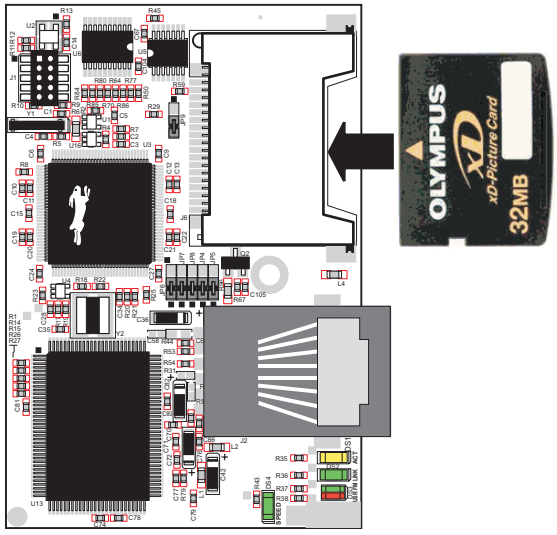


Figure 10. Insertion/Removal of xD-Picture Card

It is possible to hot-swap *xD-Picture Cards* without removing power from the RCM3365/RCM3375 module. The file system must be closed before the cards can be hot-swapped. The chip selects associated with the NAND flash and the *xD-Picture Card* must be set to their inactive state, and read/write operations addressed to the NAND flash area cannot be allowed to occur. These operations can be initiated in software by sensing an external switch actuated by the user, and the *xD-Picture Card* can then be removed and replaced with a different one. Once the application program detects a new card, the file system can be opened. These steps allow the *xD-Picture Card* to be installed or removed without affecting either the program, which continues to run on the RCM3365/RCM3375 module, or the data stored on the *xD-Picture Card*.

The **FAT_HOT_SWAP_336x0.C** sample program in the **SAMPLES\FileSystem** folder illustrates this hot-swapping procedure.

Rabbit recommends that you use header J6 only for the *xD-Picture Card* since other devices are not supported. Be careful to remove and insert the *xD-Picture Card* as shown, and be careful ***not*** to insert any foreign objects, which may short out the contacts and lead to the destruction of your *xD-Picture Card*.

Sample programs in the **SAMPLES\RCM3360\NANDFlash** folder illustrate the use of the NAND flash. These sample programs are described in Section 3.2.1, “Use of NAND Flash.” Pay careful attention to the sample programs to see how to close files and secure any data on the *xD-Picture Card* before you remove it.

for additional information if you are using a Dynamic C release prior to v. 9.60 under Windows Vista. Programs can be downloaded at baud rates of up to 460,800 bps after the program compiles.

Dynamic C has a number of standard features.

- Full-feature source and/or assembly-level debugger, no in-circuit emulator required.
- Royalty-free TCP/IP stack with source code and most common protocols.
- Hundreds of functions in source-code libraries and sample programs:
  - Exceptionally fast support for floating-point arithmetic and transcendental functions.
  - RS-232 and RS-485 serial communication.
  - Analog and digital I/O drivers.
  - I^2C, SPI, GPS, file system.
  - LCD display and keypad drivers.
- Powerful language extensions for cooperative or preemptive multitasking
- Loader utility program to load binary images into Rabbit targets in the absence of Dynamic C.
- Provision for customers to create their own source code libraries and augment on-line help by creating "function description" block comments using a special format for library functions.
- Standard debugging features:
  - Breakpoints—Set breakpoints that can disable interrupts.
  - Single-stepping—Step into or over functions at a source or machine code level, µC/OS-II aware.
  - Code disassembly—The disassembly window displays addresses, opcodes, mnemonics, and machine cycle times. Switch between debugging at machine-code level and source-code level by simply opening or closing the disassembly window.
  - Watch expressions—Watch expressions are compiled when defined, so complex expressions including function calls may be placed into watch expressions. Watch expressions can be updated with or without stopping program execution.
  - Register window—All processor registers and flags are displayed. The contents of general registers may be modified in the window by the user.
  - Stack window—shows the contents of the top of the stack.
  - Hex memory dump—displays the contents of memory at any address.
  - **STDIO** window—`printf` outputs to this window and keyboard input on the host PC can be detected for debugging purposes. `printf` output may also be sent to a serial port or file.

5.2 Dynamic C Functions

5.2.1 Digital I/O

The RCM3365/RCM3375 was designed to interface with other systems, and so there are no drivers written specifically for the I/O. The general Dynamic C read and write functions allow you to customize the parallel I/O to meet your specific needs. For example, use

```
WrPortI(PEDDR, &PEDDRShadow, 0x00);
```

to set all the Port E bits as inputs, or use

```
WrPortI(PEDDR, &PEDDRShadow, 0xFF);
```

to set all the Port E bits as outputs.

When using the external I/O bus on the Rabbit 3000 chip, add the line

```
#define PORTA_AUX_IO    // required to enable external I/O bus
```

to the beginning of any programs using the external I/O bus.

The sample programs in the Dynamic C **SAMPLES/RCM3360** folder provide further examples.

5.2.2 SRAM Use

The RCM3365/RCM3375 have a battery-backed data SRAM and a program-execution SRAM. Dynamic C provides the **protected** keyword to identify variables that are to be placed into the battery-backed SRAM. The compiler generates code that maintains two copies of each protected variable in the battery-backed SRAM. The compiler also generates a flag to indicate which copy of the protected variable is valid at the current time. This flag is also stored in the battery-backed SRAM. When a protected variable is updated, the "inactive" copy is modified, and is made "active" only when the update is 100% complete. This assures the integrity of the data in case a reset or a power failure occurs during the update process. At power-on the application program uses the active copy of the variable pointed to by its associated flag.

The sample code below shows how a protected variable is defined and how its value can be restored.

```
protected nf_device nandFlash;
int main() {
   ...
   _sysIsSoftReset();    // restore any protected variables
```

The **bbram** keyword may also be used instead if there is a need to store a variable in battery-backed SRAM without affecting the performance of the application program. Data integrity is ***not*** assured when a reset or power failure occurs during the update process.

Additional information on **bbram** and **protected** variables is available in the ***Dynamic C User's Manual***.

5.2.6 Prototyping Board Functions

The functions described in this section are for use with the Prototyping Board features. The source code is in the Dynamic C **SAMPLES\RCM3300\RCM33xx.LIB** library if you need to modify it for your own board design.

The **RCM33xx.LIB** library is supported by the **RN_CFG_RCM33.LIB**—library, which is used to configure the RCM3365/RCM3375 for use with RabbitNet peripheral boards on the Prototyping Board.

Other generic functions applicable to all devices based on Rabbit microprocessors are described in the ***Dynamic C Function Reference Manual***.

5.2.6.1 Board Initialization

```
void brdInit (void);
```

Call this function at the beginning of your program. This function initializes Parallel Ports A through G for use with the Prototyping Board.

Summary of Initialization

1. I/O port pins are configured for Prototyping Board operation.
2. Unused configurable I/O are set as tied inputs or outputs.
3. External I/O are disabled.
4. The LCD/keypad module is disabled.
5. RS-485 is not enabled.
6. RS-232 is not enabled.
7. LEDs are off.
8. Ethernet select is disabled.
9. Mass-storage flash select is disabled.
10. Motor control is disabled.
11. The RabbitNet SPI interface is disabled.
12. The relay is set to normally closed positions.

RETURN VALUE

None.

5.2.6.3 Switches, LEDs, and Relay

int switchIn(int swin);

Reads the state of a switch input.

PARAMETERS

swin is the switch input to read:

2—S2
3—S3

RETURN VALUE

State of the switch input:

1 = open
0 = closed

SEE ALSO

brdInit

void ledOut(int led, int value);

Controls LEDs on the Prototyping Board and on the RCM3365/RCM3375.

PARAMETERS

led is the LED to control:

0 = red User LED on RCM3365/RCM3375
3 = DS3 on Prototyping Board
4 = DS4 on Prototyping Board
5 = DS5 on Prototyping Board
6 = DS6 on Prototyping Board

value is the value used to control the LED:

0 = off
1 = on

RETURN VALUE

None.

SEE ALSO

brdInit

void rn_sp_close(int port);

Deactivates the RCM3365/RCM3375 RabbitNet port as a clocked serial port. This call is also used by **rn_init()**.

PARAMETERS

portnum = 0

RETURN VALUE

None

void rn_sp_enable(int portnum);

This is a macro that enables or asserts the RCM3365/RCM3375 RabbitNet port chip select prior to data transfer.

PARAMETERS

portnum = 0

RETURN VALUE

None

void rn_sp_disable(int portnum);

This is a macro that disables or deasserts the RCM3365/RCM3375 RabbitNet port chip select to invalidate data transfer.

PARAMETERS

portnum = 0

RETURN VALUE

None.

6. USING THE TCP/IP FEATURES

6.1 TCP/IP Connections

Programming and development can be done with the RCM3365/RCM3375 modules without connecting the Ethernet port to a network. However, if you will be running the sample programs that use the Ethernet capability or will be doing Ethernet-enabled development, you should connect the RCM3365/RCM3375 module's Ethernet port at this time.

Before proceeding you will need to have the following items.

- If you don't have Ethernet access, you will need at least a 10Base-T Ethernet card (available from your favorite computer supplier) installed in a PC.
- Two RJ-45 straight-through Ethernet cables and a hub, or an RJ-45 crossover Ethernet cable.

A straight-through and a crossover Ethernet cable are included in the RCM3365/RCM3375 Development Kit. Figure 11 shows how to identify the two cables based on the wires in the transparent RJ-45 connectors.

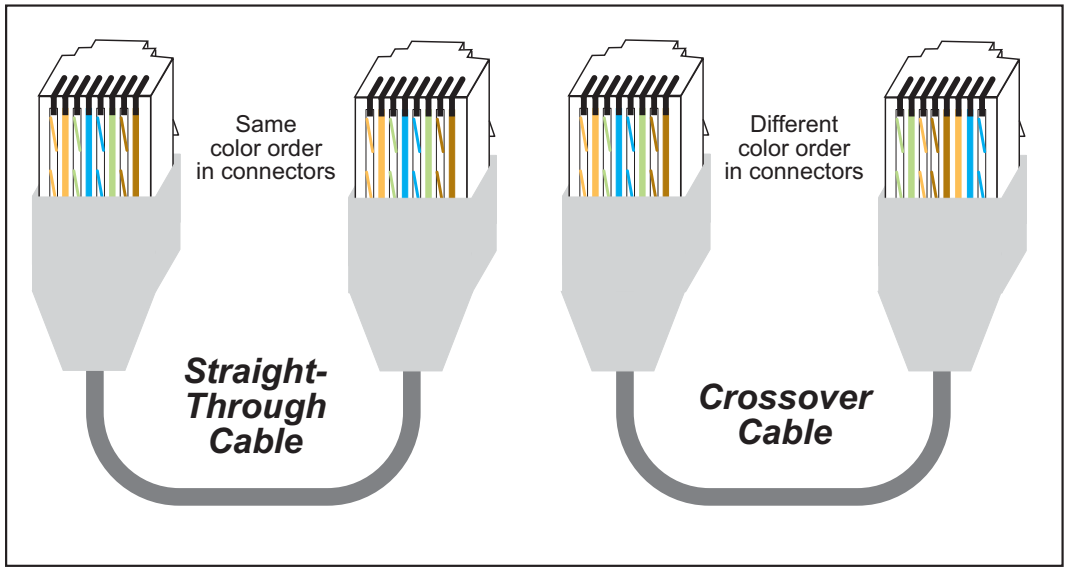


Figure 11. How to Identify Straight-Through and Crossover Ethernet Cables

Ethernet cables and a 10Base-T Ethernet hub are available from Rabbit in a TCP/IP tool kit. More information is available at www.rabbit.com.

Now you should be able to make your connections.

1. Connect the AC adapter and the serial programming cable as shown in Chapter 2, "Getting Started."
2. Ethernet Connections

   There are four options for connecting the RCM3365/RCM3375 module to a network for development and runtime purposes. The first two options permit total freedom of action in selecting network addresses and use of the "network," as no action can interfere with other users. We recommend one of these options for initial development.

- **No LAN —** The simplest alternative for desktop development. Connect the RCM3365/RCM3375 module's Ethernet port directly to the PC's network interface card using an RJ-45 *crossover cable*. A crossover cable is a special cable that flips some connections between the two connectors and permits direct connection of two client systems. A standard RJ-45 network cable will not work for this purpose.
- **Micro-LAN —** Another simple alternative for desktop development. Use a small Ethernet 10Base-T hub and connect both the PC's network interface card and the RCM3365/RCM3375 module's Ethernet port to it using standard network cables.

The following options require more care in address selection and testing actions, as conflicts with other users, servers and systems can occur:

- **LAN —** Connect the RCM3365/RCM3375 module's Ethernet port to an existing LAN, preferably one to which the development PC is already connected. You will need to obtain IP addressing information from your network administrator.
- **WAN —** The RCM3365/RCM3375 is capable of direct connection to the Internet and other Wide Area Networks, but exceptional care should be used with IP address settings and all network-related programming and development. We recommend that development and debugging be done on a local network before connecting a RabbitCore system to the Internet.

  TIP: Checking and debugging the initial setup on a micro-LAN is recommended before connecting the system to a LAN or WAN.

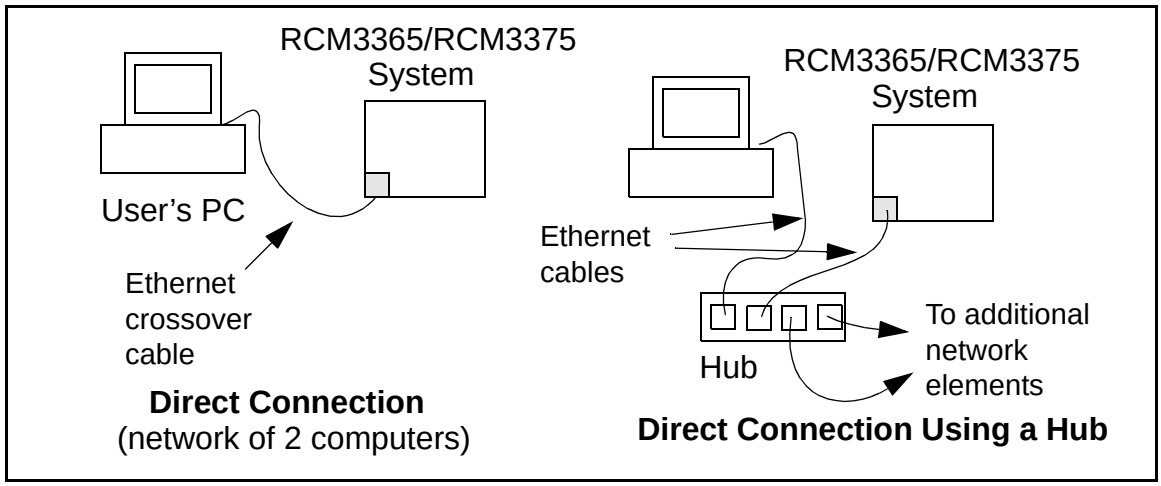
The PC running Dynamic C does not need to be the PC with the Ethernet card.

3. Apply Power

   Plug in the AC adapter. The RCM3365/RCM3375 module and Prototyping Board are now ready to be used.

6.4 Running TCP/IP Sample Programs

We have provided a number of sample programs demonstrating various uses of TCP/IP for networking embedded systems. These programs require you to connect your PC and the RCM3365/RCM3375 board together on the same network. This network can be a local private network (preferred for initial experimentation and debugging), or a connection via the Internet.



The sample programs described in this chapter may also be run with a RabbitSys-enabled RCM3365 operating in the RabbitSys mode. There is no change to the instructions when you use the serial programming cable. When you use an Ethernet cable, you may use CAT 5/6 straight-through Ethernet cables to connect the RCM3365 and your PC to a DHCP network. It is not necessary to use a crossover cable for a direct connection. Use the TCP/IP parameters such as the IP address that you identified with the rdiscover utility; if you are using an Ethernet crossover cable to connect the RCM3365 directly to your PC, use the TCP/IP parameters that you set up according to the instructions in Appendix E.

A.3 Rabbit 3000 DC Characteristics

Table A-5. Rabbit 3000 Absolute Maximum Ratings

Symbol	Parameter	Maximum Rating
T_A	Operating Temperature	-55° to +85°C
T_S	Storage Temperature	-65° to +150°C
	Maximum Input Voltage: • Oscillator Buffer Input • 5-V-tolerant I/O	 V_{DD} + 0.5 V 5.5 V
V_{DD}	Maximum Operating Voltage	3.6 V

Stresses beyond those listed in Table A-5 may cause permanent damage. The ratings are stress ratings only, and functional operation of the Rabbit 3000 chip at these or any other conditions beyond those indicated in this section is not implied. Exposure to the absolute maximum rating conditions for extended periods may affect the reliability of the Rabbit 3000 chip.

Table A-6 outlines the DC characteristics for the Rabbit 3000 at 3.3 V over the recommended operating temperature range from T_A = –55°C to +85°C, V_{DD} = 3.0 V to 3.6 V.

Table A-6. 3.3 Volt DC Characteristics

Symbol	Parameter	Test Conditions	Min	Typ	Max	Units
V_{DD}	Supply Voltage		3.0	3.3	3.6	V
V_{IH}	High-Level Input Voltage		2.0			V
V_{IL}	Low-Level Input Voltage				0.8	V
V_{OH}	High-Level Output Voltage	I_{OH} = 6.8 mA, V_{DD} = V_{DD} (min)	0.7 x V_{DD}			V
V_{OL}	Low-Level Output Voltage	I_{OL} = 6.8 mA, V_{DD} = V_{DD} (min)			0.4	V
I_{IH}	High-Level Input Current (absolute worst case, all buffers)	V_{IN} = V_{DD}, V_{DD} = V_{DD} (max)			10	µA
I_{IL}	Low-Level Input Current (absolute worst case, all buffers)	V_{IN} = V_{SS}, V_{DD} = V_{DD} (max)	-10			µA
I_{OZ}	High-Impedance State Output Current (absolute worst case, all buffers)	V_{IN} = V_{DD} or V_{SS}, V_{DD} = V_{DD} (max), no pull-up	-10		10	µA

B.4.3 CMOS Digital Outputs

If the stepper-motor option is not used, eight CMOS-level digital outputs are available at J10, and can each handle up to 25 mA.

B.4.4 Sinking Digital Outputs

Four sinking digital outputs shared with LEDs DS3–DS6 are available at J12, and can each handle up to 500 mA. Figure B-6 shows a wiring diagram for a typical sinking output.

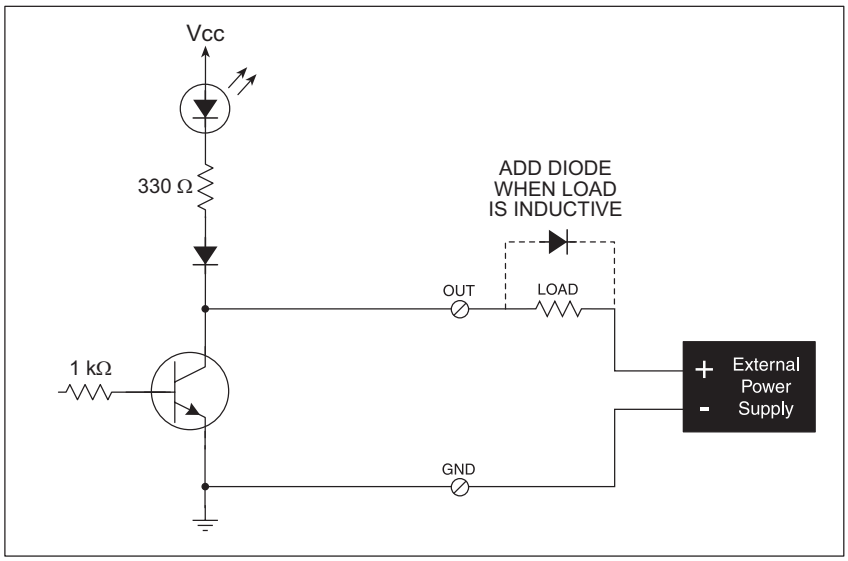


Figure B-6. Prototyping Board Sinking Digital Outputs

B.4.5 Relay Outputs

Figure B-7 shows the contact connections for the relay on the Prototyping Board. A diode across the coil provides a return path for inductive spikes, and snubbers across the relay contacts protect the relay contacts from inductive spikes.

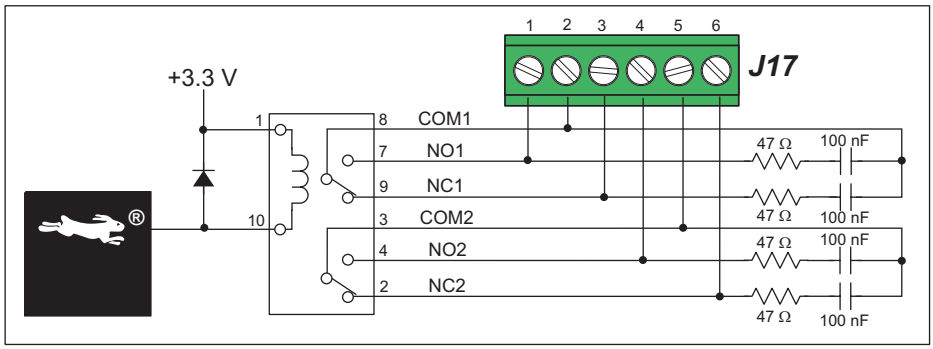


Figure B-7. Prototyping Board Relay Output Contact Connections

The relay is driven by pin PA4 of the RCM3365/RCM3375 module via U8, and is controlled by PE7 and PG5 as shown in the sample applications.

3. Fasten the unit with the four 4-40 screws and washers included with the LCD/keypad module. If your panel is thick, use a 4-40 screw that is approximately 3/16" (5 mm) longer than the thickness of the panel.

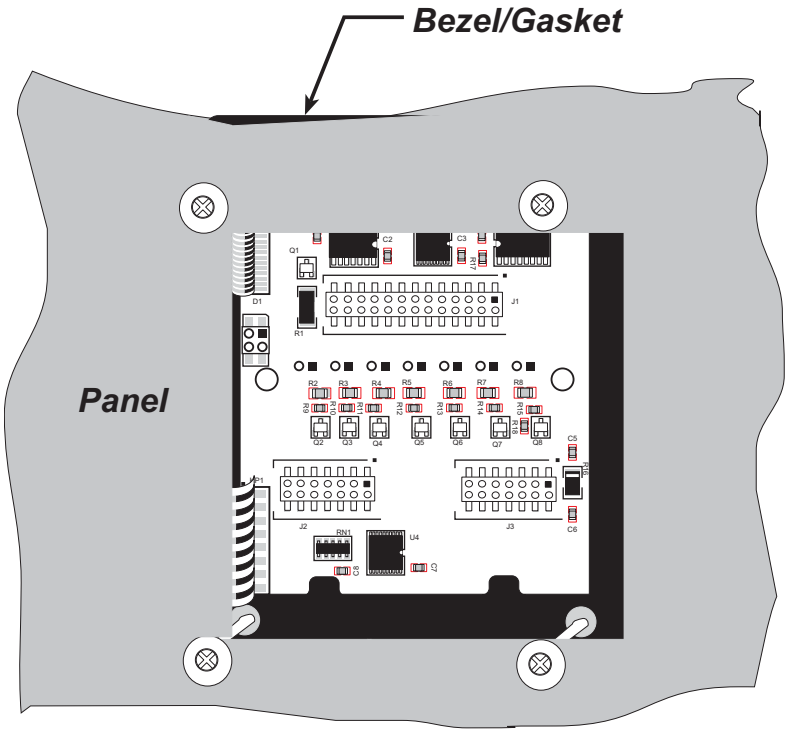


Figure C-9. LCD/Keypad Module Mounted in Panel (rear view)

Carefully tighten the screws until the gasket is compressed and the plastic bezel face-plate is touching the panel.

Do not tighten each screw fully before moving on to the next screw. Apply only one or two turns to each screw in sequence until all are tightened manually as far as they can be so that the gasket is compressed and the plastic bezel faceplate is touching the panel.

void keypadDef();

Configures the physical layout of the keypad with the default ASCII return key codes.

Keypad physical mapping 1 x 7

0	4	1	5	2	6	3
['L']		['U']		['D']		['R']
	['–']		['+']		['E']	

where

'D' represents Down Scroll
'U' represents Up Scroll
'R' represents Right Scroll
'L' represents Left Scroll
'–' represents Page Down
'+' represents Page Up
'E' represents the ENTER key

Example: Do the followingfor the above physical vs. ASCII return key codes.

```
keyConfig ( 3,'R',0, 0, 0, 0, 0 );
keyConfig ( 6,'E',0, 0, 0, 0, 0 );
keyConfig ( 2,'D',0, 0, 0, 0, 0 );
keyConfig ( 4,'-',0, 0, 0, 0, 0 );
keyConfig ( 1,'U',0, 0, 0, 0, 0 );
keyConfig ( 5,'+',0, 0, 0, 0, 0 );
keyConfig ( 0,'L',0, 0, 0, 0, 0 );
```

Characters are returned upon keypress with no repeat.

RETURN VALUE

None.

SEE ALSO

keyConfig, keyGet, keyProcess

void keyScan(char *pcKeys);

Writes "1" to each row and reads the value. The position of a keypress is indicated by a zero value in a bit position.

PARAMETER

***pcKeys** is a pointer to the address of the value read.

RETURN VALUE

None.

SEE ALSO

keyConfig, keyGet, keypadDef, keyProcess

F.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

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