Digi - 20-101-0494 Datasheet

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	Not For New Designs
Module/Board Type	MPU Core
Core Processor	Rabbit 2000
Co-Processor	-
Speed	22.1MHz
Flash Size	512KB
RAM Size	512KB
Connector Type	2 IDC Headers 2x13
Size / Dimension	1.6" x 2.3" (41mm x 58mm)
Operating Temperature	0°C ~ 70°C
Purchase URL	https://www.e-xfl.com/product-detail/digi-international/20-101-0494

RabbitCore RCM2200 User's Manual

Part Number 019-0097 • 090417–G • Printed in U.S.A.

Trademarks

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

Digi International Inc.

www.rabbit.com

To power down the Prototyping Board, unplug the power connector from J5. You should disconnect power before making any circuit adjustments in the prototyping area, changing any connections to the board, or removing the RabbitCore module from the board.

2.1.4 Alternate Power Supply Connections

Development kits sold outside North America before 2009 included a header connector that could be connected to 3-pin header J5 on the Prototyping Board. The red and black wires from the connector could then be connected to the positive and negative connections on your power supply. The power supply should deliver 8 V–24 V DC at 8 W.

2.2 Run a Sample Program

Once the RCM2200 is connected as described in the preceding pages, start Dynamic C by double-clicking on the Dynamic C icon on your desktop or in your **Start** menu. Dynamic C uses the serial port specified during installation.

If you are using a USB port to connect your computer to the RCM2200 module, choose **Options > Project Options** and select “Use USB to Serial Converter” under the **Communications** tab, then click **OK**.

Find the file `PONG.C`, which is in the Dynamic C `SAMPLES` folder. To run the 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 **STDIO** window will open and will display a small square bouncing around in a box.

2.2.1 Troubleshooting

If Dynamic C cannot find the target system (error message `"No Rabbit Processor Detected."`):

- Check that the RCM2200 is powered correctly — the red power LED on the Prototyping Board should be lit when the RCM2200 is mounted on the Prototyping Board and the AC adapter is plugged in.
- Check both ends of the programming cable to ensure that they are firmly plugged into the PC and the **PROG** connector, not the **DIAG** connector, is plugged in to the programming port on the RCM2200 with the marked (colored) edge of the programming cable towards pin 1 of the programming header.
- Ensure that the RCM2200 module is firmly and correctly installed in its connectors on the Prototyping Board.
- Dynamic C uses the COM port specified during installation. Select a different COM port within Dynamic C. From the **Options** menu, select **Project Options**, then select **Communications**. Select another COM port from the list, then click OK. Press **<Ctrl-Y>** to force Dynamic C to recompile the BIOS. If Dynamic C still reports it is unable to locate the target system, repeat the above steps until you locate the COM port used by the programming cable.

3. Running Sample Programs

To develop and debug programs for the RCM2200 (and for all other Rabbit hardware), you must install and use Dynamic C. This chapter provides a tour of the sample programs for the RCM2200.

3.1 Sample Programs

To help familiarize you with the RCM2200 modules, several sample Dynamic C programs have been included. Loading, executing and studying these programs will give you a solid hands-on overview of the RC M2200's capabilities, as well as a quick start with Dynamic C as an application development tool. These programs are intended to serve as tutorials, but then can also be used as starting points or building blocks for your own applications.

NOTE: It is assumed in this section that you have at least an elementary grasp of ANSI C. If you do not, see the introductory pages of the ***Dynamic C User's Manual*** for a suggested reading list.

Each sample program has comments that describe the purpose and function of the program.

Before running any of these sample program, make sure that your RCM2200 is connected to the Prototyping Board and to your PC as described in Section 2.1, "Connections." To run a sample program, open it with the **File** menu (if it is not already open), then compile and run it by pressing **F9** or by selecting **Run** in the **Run** menu.

Sample programs are provided in the Dynamic C **`SAMPLES`** folder. Two folders contain sample programs that illustrate features unique to the RCM2200.

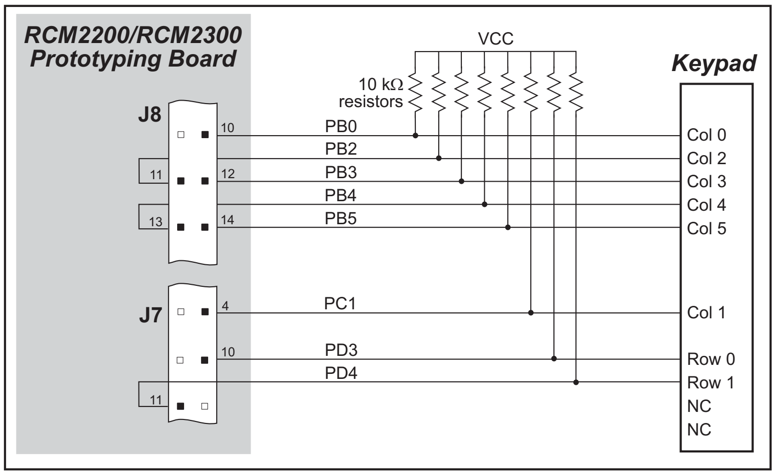
- **`RCM2200`**—Demonstrates the basic operation and the Ethernet functionality of the RCM2200.
- **`TCPIP`**—Demonstrates more advanced TCP/IP programming for Rabbit's Ethernet-enabled Rabbit-based boards.

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

- **`KEYLCD.C`**—demonstrates a simple setup for a 2 × 6 keypad and a 2 × 20 LCD.

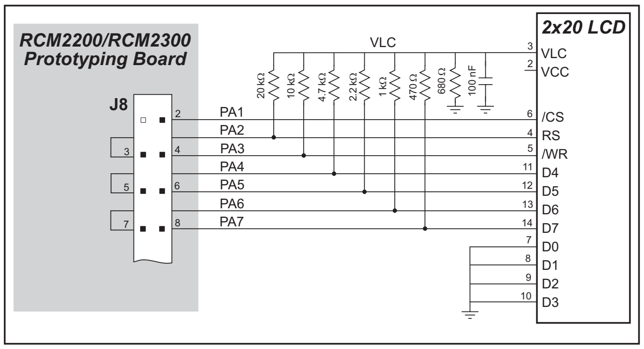
  Connect the keypad to Parallel Ports B, C, and D.

  PB0—Keypad Col 0
  PC1—Keypad Col 1
  PB2—Keypad Col 2
  PB3—Keypad Col 3
  PB4—Keypad Col 4
  PB5—Keypad Col 5
  PD3—Keypad Row 0
  PD4—Keypad Row 1



  Connect the LCD to Parallel Port A.

  PA0—backlight (if connected)
  PA1—LCD /CS
  PA2—LCD RS (High = Control, Low = Data) / LCD Contrast 0
  PA3—LCD /WR/ LCD Contrast 1
  PA4—LCD D4 / LCD Contrast 2
  PA5—LCD D5 / LCD Contrast 3
  PA6—LCD D6 / LCD Contrast 4
  PA7—LCD D7 / LCD Contrast 5

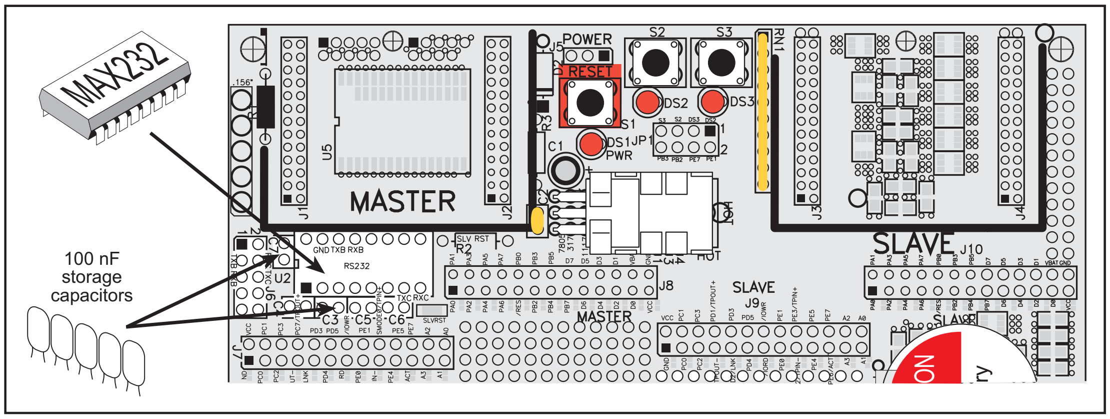


  Once the connections have been made and the sample program is running, the LCD will display two rows of 6 dots, each dot representing the corresponding key. When a key is pressed, the corresponding dot will become an asterisk.

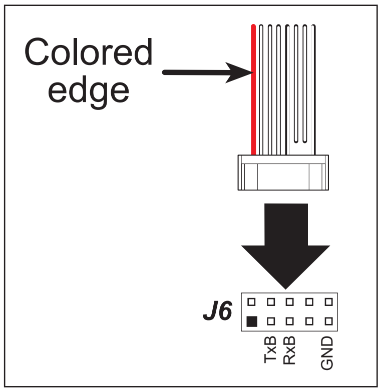
3.1.2 Serial Communication

The following sample programs can be found in the **`SAMPLES\RCM2200`** folder.

One sample programs, **`PUTS.C`** is available to illustrate RS-232 communication. To run this sample program, you will have to add an RS-232 transceiver such as the MAX232 at location U2 and five 100 nF capacitors at C3–C7 on the Prototyping Board. Also install a 2 × 5 IDC header with a pitch of 0.1" at J6 to interface the RS-232 signals. The diagram shows the connections.



Once the sample program is running, you may use a 10-pin header to DB9 cable (for example, Part No. 540-0009) to connect header J6 to your PC COM port (you will have to disconnect the programming cable from both the RCM2200 and the PC if you only have one PC COM port, then press the **RESET** button on the Prototyping Board). Line up the colored edge of the cable with pin 1 on header J6 as shown in the diagram (pin 1 is indicated by a small square on the Prototyping Board silkscreen).



This program writes a null terminated string over Serial Port B. Use a serial utility such as HyperTerminal or Tera Term to view the string. Use the following configuration for your serial utility.

Bits per second: 19200

Data bits: 8

Parity: None

Stop bits: 1

Flow control: None

3.1.4 Sample Program Descriptions

3.1.4.1 FLASHLED.C

This program is about as simple as a Dynamic C application can get—the equivalent of the traditional "Hello, world!" program found in most basic programming tutorials. If you are familiar with ANSI C, you should have no trouble reading through the source code and understanding it.

The only new element in this sample application should be Dynamic C's handling of the Rabbit microprocessor's parallel ports. The program:

4. Initializes the pins of Port A as outputs.
5. Sets all of the pins of Port A high, turning off the attached LEDs.
6. Starts an endless loop with a `for(;;)` expression, and within that loop:
    - Writes a bit to turn bit 1 off, lighting LED DS3;
    - Waits through a delay loop;
    - Writes a bit to turn bit 1 on, turning off the LED;
    - Waits through a second delay loop;

   These steps repeat as long as the program is allowed to run.

You can change the flash rate of the LED by adjusting the loop values in the two `for` expressions. The first loop controls the LED's "off" time; the second loop controls its "on" time.

> **NOTE:** Since the variable `j` is defined as type `int`, the range for `j` must be between 0 and 32767. To permit larger values and thus longer delays, change the declaration of `j` to `unsigned int` or `long`.

More Information

See the section on primitive data types, and the entries for the library functions `WrPortI( )` and `BitWrPortI( )` in the ***Dynamic C User's Manual***.

tion of how Dynamic C handles multitasking with costatements and cofunctions, see Chapter 5, “Multitasking with Dynamic C,” and Chapter 6, “The Virtual Driver,” in the ***Dynamic C User’s Manual***.

3.1.4.3 TOGGLELED.C

One of Dynamic C’s unique and powerful aspects is its ability to efficiently multitask using *cofunctions* and *costatements*. This simple application demonstrates how these program elements work.

This sample program uses two costatements to set up and manage the two tasks. Costatements must be contained in a loop that will “tap” each of them at regular intervals. This program:

1. Initializes the pins of Port A as outputs.
2. Sets all the pins of Port A high, turning off the attached LEDs.
3. Sets the toggled LED status variable **`vswitch`** to 0 (LED off).
4. Starts an endless loop using a **`while(1)`** expression, and within that loop:
    - Executes a costatement that flashes LED DS3;
    - Executes a costatement that checks the state of switch S2 and toggles the state of **`vswitch`** if it is pressed;
    - Turns LED DS2 on or off, according to the state of **`vswitch`**.

    These steps repeat as long as the program is allowed to run.

The first costatement is a compressed version of **`FLASHLED.c`**, with slightly different flash timing. It also uses the library function **`DelayMs()`** to deliver more accurate timing than the simple delay loops of the previous program.

The second costatement does more than check the status of S2. Switch contacts often “bounce” open and closed several times when the switch is actuated, and each bounce can be interpreted by fast digital logic as an independent press. To clean up this input, the code in the second costatement “debounces” the switch signal by waiting 50 milliseconds and checking the state of the switch again. If it is detected as being closed both times, the program considers it a valid switch press and toggles **`vswitch`**.

Unlike most C statements, the two costatements are not executed in their entirety on each iteration of the **`while(1)`** loop. Instead, the list of statements within each costatement is initiated on the first loop, and then executed one “slice” at a time on each successive iteration. This mode of operation is known as a *state machine*, a powerful concept that permits a single processor to efficiently handle a number of independent tasks.

The ability of Dynamic C to manage state machine programs enables you to create very powerful and efficient embedded systems with much greater ease than other programming methods.

More Information

See the entries for the **`DelayMs()`** function, as well as Section 5, “Multitasking with Dynamic C,” in the ***Dynamic C User’s Manual***.

4.3.2 Standalone Operation of the RCM2200

The RCM2200 must be programmed via the RCM2200/RCM2300 Prototyping Board or via a similar arrangement on a customer-supplied board. Once the RCM2200 has been programmed successfully, remove the programming cable from the programming connector and reset the RCM2200. The RCM2200 may be reset by cycling the power off/on or by pressing the **RESET** button on the Prototyping Board. The RCM2200 module may now be removed from the Prototyping Board for end-use installation.

CAUTION: Power to the Prototyping Board or other boards should be disconnected when removing or installing your RCM2200 module to protect against inadvertent shorts across the pins or damage to the RCM2200 if the pins are not plugged in correctly. Do not reapply power until you have verified that the RCM2200 module is plugged in correctly.

Developing software with Dynamic C is simple. Users can write, compile, and test C and assembly code without leaving the Dynamic C development environment. Debugging occurs while the application runs on the target. Alternatively, users can compile a program to an image file for later loading. Dynamic C runs on PCs under Windows 95 or later. 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:

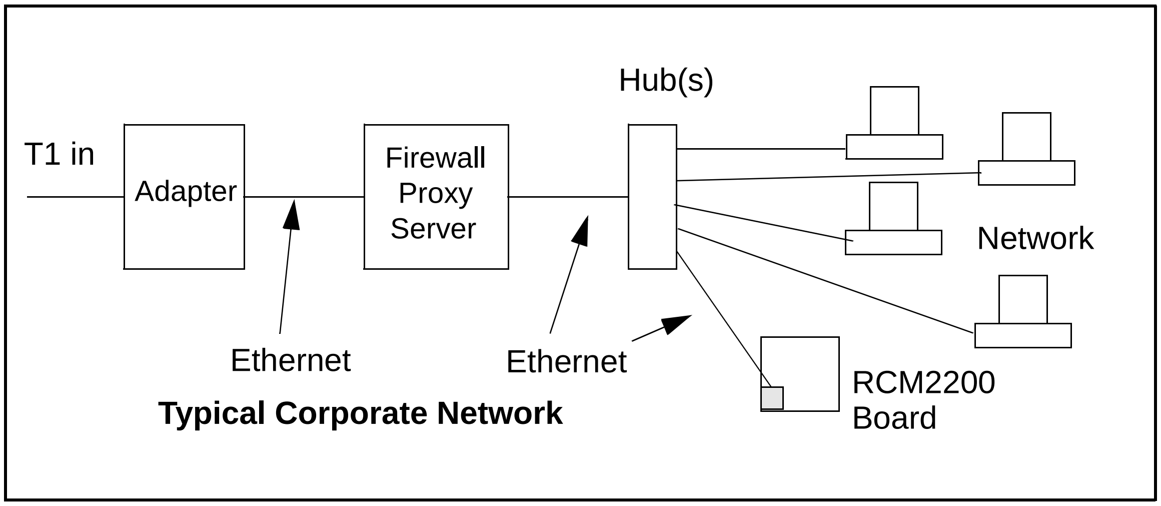
- Full-feature source and/or assembly-level debugger, no in-circuit emulator required.
- Royalty-free TCP/IP stack with source code and most common protocols.
- Hundreds of functions in source-code libraries and sample programs:
  - ▶ Exceptionally fast support for floating-point arithmetic and transcendental 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.

The following IP addresses are set aside for local networks and are not allowed on the Internet: 10.0.0.0 to 10.255.255.255, 172.16.0.0 to 172.31.255.255, and 192.168.0.0 to 192.168.255.255.

The RCM2200 board uses a 10Base-T type of Ethernet connection, which is the most common scheme. The RJ-45 connectors are similar to U.S. style telephone connectors, except they are larger and have 8 contacts.

An alternative to the direct connection using a crossover cable is a direct connection using a hub. The hub relays packets received on any port to all of the ports on the hub. Hubs are low in cost and are readily available. The RCM2200 board uses 10 Mbps Ethernet, so the hub or Ethernet adapter must be either a 10 Mbps unit or a 10/100 unit that adapts to either 10 or 100 Mbps.

In a corporate setting where the Internet is brought in via a high-speed line, there are typically machines between the outside Internet and the internal network. These machines include a combination of proxy servers and firewalls that filter and multiplex Internet traffic. In the configuration below, the RCM2200 board could be given a fixed address so any of the computers on the local network would be able to contact it. It may be possible to configure the firewall or proxy server to allow hosts on the Internet to directly contact the controller, but it would probably be easier to place the controller directly on the external network outside of the firewall. This avoids some of the configuration complications by sacrificing some security.



If your system administrator can give you an Ethernet cable along with its IP address, the netmask and the gateway address, then you may be able to run the sample programs without having to setup a direct connection between your computer and the RCM2200 board. You will also need the IP address of the nameserver, the name or IP address of your mail server, and your domain name for some of the sample programs.

6.5 Dynamically Assigned Internet Addresses

In many instances, there are no fixed IP addresses. This is the case when, for example, you are assigned an IP address dynamically by your dial-up Internet service provider (ISP) or when you have a device that provides your IP addresses using the Dynamic Host Configuration Protocol (DHCP). The RCM2200 can use such IP addresses to send and receive packets on the Internet, but you must take into account that this IP address may only be valid for the duration of the call or for a period of time, and could be a private IP address that is not directly accessible to others on the Internet. These private address can be used to perform some Internet tasks such as sending e-mail or browsing the Web, but usually cannot be used to participate in conversations that originate elsewhere on the Internet. If you want to find out this dynamically assigned IP address, under Windows XP you can run the **`ipconfig`** program while you are connected and look at the interface used to connect to the Internet.

Many networks use private IP addresses that are assigned using DHCP. When your computer comes up, and periodically after that, it requests its networking information from a DHCP server. The DHCP server may try to give you the same address each time, but a fixed IP address is usually not guaranteed.

If you are not concerned about accessing the RCM2200 from the Internet, you can place the RCM2200 on the internal network using a private address assigned either statically or through DHCP.

6.7 How to Set IP Addresses in the Sample Programs

We have provided a number of sample programs demonstrating various uses of TCP/IP for networking embedded systems. These programs require that you connect your PC and the Coyote 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.

With the introduction of Dynamic C 7.30 we have taken steps to make it easier to run many of our sample programs. You will see a **`TCPCONFIG`** macro. This macro tells Dynamic C to select your configuration from a list of default configurations. You will have three choices when you encounter a sample program with the **`TCPCONFIG`** macro.

1. You can replace the **`TCPCONFIG`** macro with individual **`MY_IP_ADDRESS`**, **`MY_NETMASK`**, **`MY_GATEWAY`**, and **`MY_NAMESERVER`** macros in each program.
2. You can leave **`TCPCONFIG`** at the usual default of 1, which will set the IP configurations to **`10.10.6.100`**, the netmask to **`255.255.255.0`**, and the nameserver and gateway to **`10.10.6.1`**. If you would like to change the default values, for example, to use an IP address of **`10.1.1.2`** for the Coyote board, and **`10.1.1.1`** for your PC, you can edit the values in the section that directly follows the "General Configuration" comment in the **`TCP_CONFIG.LIB`** library. You will find this library in the **`LIB\TCPIP`** directory.
3. You can create a **`CUSTOM_CONFIG.LIB`** library and use a **`TCPCONFIG`** value greater than 100. Instructions for doing this are at the beginning of the **`TCP_CONFIG.LIB`** library in the **`LIB\TCPIP`** directory.

There are some other "standard" configurations for **`TCPCONFIG`** that let you select different features such as DHCP. Their values are documented at the top of the **`TCP_CONFIG.LIB`** library in the **`LIB\TCPIP`** directory. More information is available in the ***Dynamic C TCP/IP User's Manual***.

IP Addresses Before Dynamic C 7.30

Most of the sample programs use macros to define the IP address assigned to the board and the IP address of the gateway, if there is a gateway. Instead of the **`TCPCONFIG`** macro, you will see a **`MY_IP_ADDRESS`** macro and other macros.

```
#define MY_IP_ADDRESS "10.10.6.170"
#define MY_NETMASK "255.255.255.0"
#define MY_GATEWAY "10.10.6.1"
#define MY_NAMESERVER "10.10.6.1"
```

In order to do a direct connection, the following IP addresses can be used for the Coyote:

```
#define MY_IP_ADDRESS "10.1.1.2"
#define MY_NETMASK "255.255.255.0"
// #define MY_GATEWAY "10.10.6.1"
// #define MY_NAMESERVER "10.10.6.1"
```

In this case, the gateway and nameserver are not used, and are commented out. The IP address of the board is defined to be **`10.1.1.2`**. The IP address of your PC can be defined as **`10.1.1.1`**.

APPENDIX A. RABBITCORE RCM2200 SPECIFICATIONS

Appendix A provides the specifications for the RCM2200, and describes the conformal coating.

It is recommended that you allow for an “exclusion zone” of 0.04" (1 mm) around the RCM2200 in all directions when the RCM2200 is incorporated into an assembly that includes other printed circuit boards. An “exclusion zone” of 0.16" (4 mm) is recommended below the RCM2200 when the RCM2200 is plugged into another assembly using the shortest connectors for headers J4 and J5. Figure A-2 shows this “exclusion zone.”

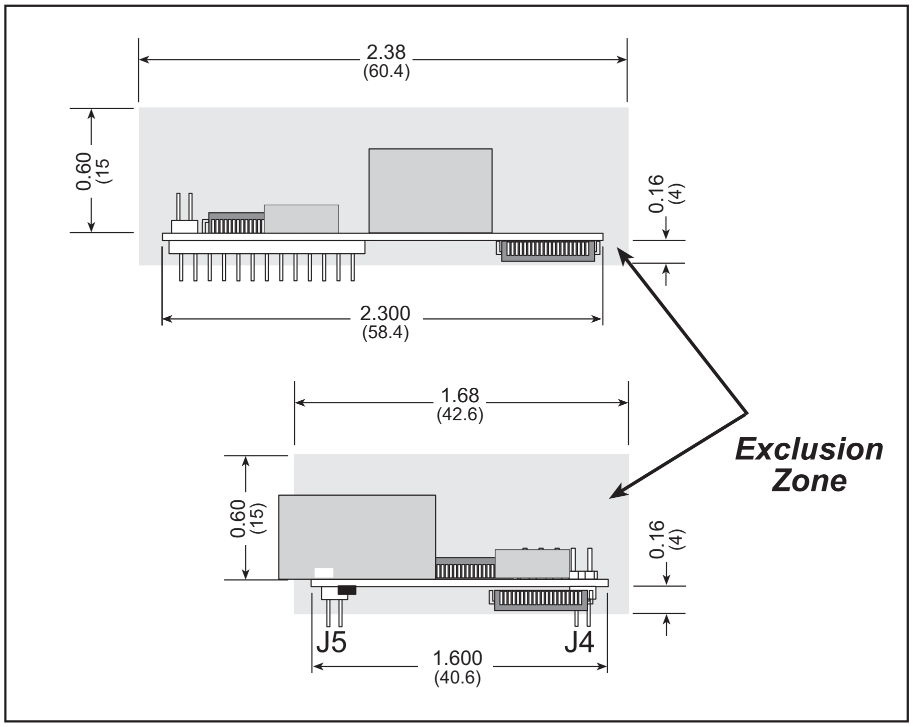


Figure A-2. RCM2200 “Exclusion Zone”

Table A-1 lists the electrical, mechanical, and environmental specifications for the RCM2200.

Table A-1. RabbitCore RCM2200 Specifications

Parameter	RCM2200	RCM2210	RCM2250	RCM2260
Microprocessor	Rabbit 2000® at 22.1 MHz			
Ethernet Port (10/100-compatible with 10Base-T interface)	RJ-45, 2 LEDs	Raw signals only	RJ-45, 2 LEDs	Raw signals only
Flash Memory	One 256K		Two 256K	Two 256K
SRAM	128K		512K	512K
Backup Battery	Connection for user-supplied backup battery (to support RTC and SRAM)			
General-Purpose I/O	26 parallel I/0 lines grouped in five 8-bit ports (shared with serial ports): • 16 configurable I/O • 7 fixed inputs • 3 fixed outputs			
Additional Inputs	2 startup mode, reset			
Additional Outputs	Status, reset			
Memory, I/O Interface	4 address lines, 8 data lines, I/O read/write			
Serial Ports	Four 5 V CMOS-compatible ports. Two ports are configurable as clocked ports, one is a dedicated RS-232 programming port.			
Serial Rate	Maximum burst rate = CLK/32 Maximum sustained rate = CLK/64			
Slave Interface	A slave port allows the RCM2200 to be used as an intelligent peripheral device slaved to a master processor, which may either be another Rabbit 2000 or any other type of processor			
Real-Time Clock	Yes			
Timers	Five 8-bit timers cascadable in pairs, one 10-bit timer with 2 match registers that each have an interrupt			
Watchdog/Supervisor	Yes			
Power	4.75 V to 5.25 V DC, 134 mA			
Operating Temperature	–40°C to +70°C			
Humidity	5% to 95%, noncondensing			
Connectors	Two IDC headers 2 × 13, 2 mm pitch			
Board Size	1.60" × 2.30" × 0.86" (41 mm × 59 mm × 22 mm)			

Figure A-4 shows a typical timing diagram for the Rabbit 2000 microprocessor external I/O read and write cycles.

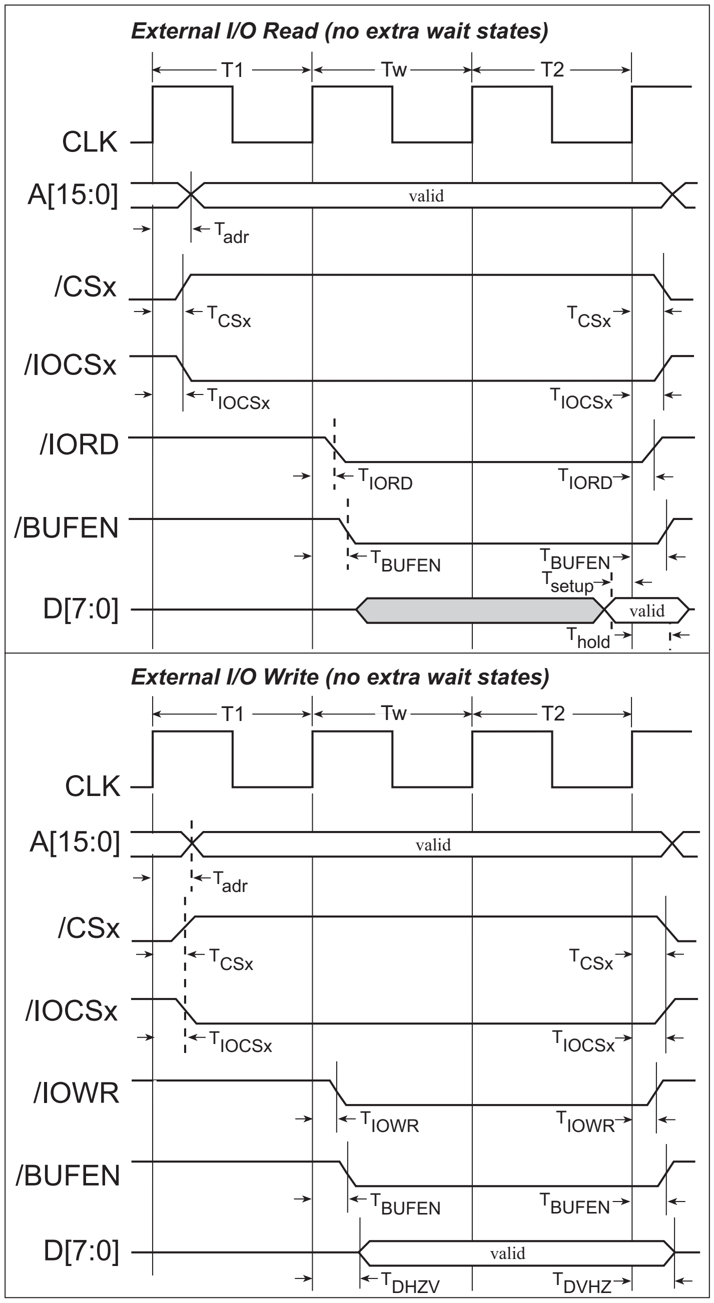


Figure A-4. External I/O Read and Write Cycles—No Extra Wait States

T_{adr} is the time required for the address output to reach 0.8 V. This time depends on the bus loading. T_{setup} is the data setup time relative to the clock. Tsetup is specified from 30%/70% of the V_{DD} voltage level.

A.6 Conformal Coating

The areas around the 32 kHz real-time clock crystal oscillator has had the Dow Corning silicone-based 1-2620 conformal coating applied. The conformally coated area is shown in Figure A-6. The conformal coating protects these high-impedance circuits from the effects of moisture and contaminants over time.

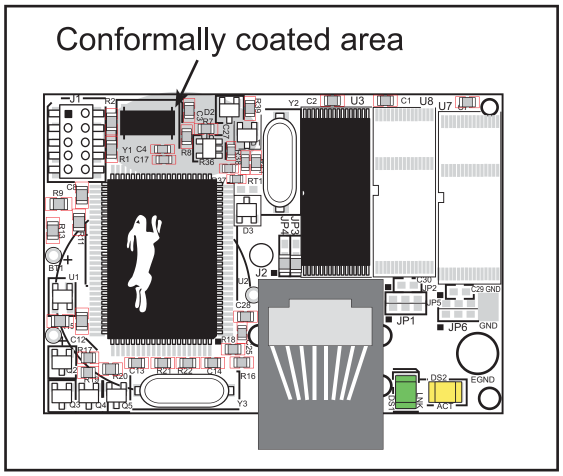


Figure A-6. RCM2200 Areas Receiving Conformal Coating

Any components in the conformally coated area may be replaced using standard soldering procedures for surface-mounted components. A new conformal coating should then be applied to offer continuing protection against the effects of moisture and contaminants.

NOTE: For more information on conformal coatings, refer to Technical Note TN303, ***Conformal Coatings***.

D.3 External Memory

The sample circuit can be used with an external 64K memory device. Larger SRAMs can be written to using this scheme by using other available Rabbit 2000 ports (parallel ports A to E) as address lines.

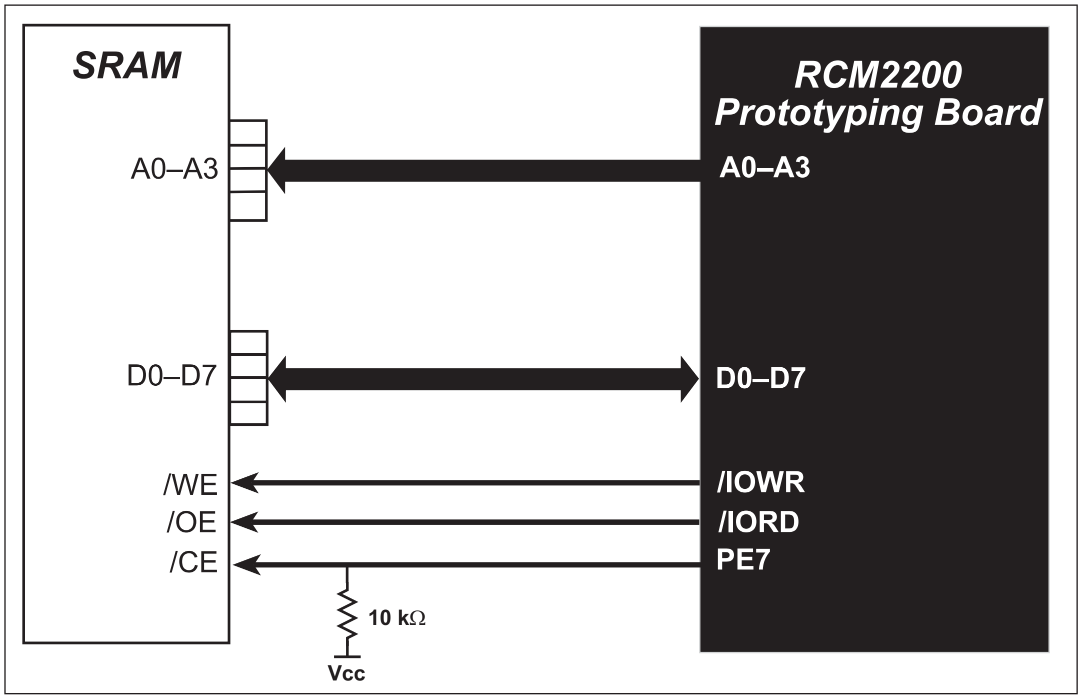


Figure D-4. Sample External Memory Connections

Sample Program: `EXTSRAM.C` in `SAMPLES\RCM2200`.