PRODUCT MANUAL

RabbitCore RCM2200

C-Programmable Module with Ethernet

User's Manual

019–0097 • 090417–G

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

TABLE OF CONTENTS

1. INTRODUCTION

The RCM2200 RabbitCore module is designed to be the heart of embedded control systems. The RCM2200 features an integrated Ethernet port and provides for LAN and Internet-enabled systems to be built as easily as serial-communication systems.

Throughout this manual, the term RCM2200 refers to the complete series of RCM2200 RabbitCore modules unless other production models are referred to specifically.

The RCM2200 has a Rabbit 2000 microprocessor operating at 22.1 MHz, static RAM, flash memory, two clocks (main oscillator and timekeeping), and the circuitry necessary for reset and management of battery backup of the Rabbit 2000's internal real-time clock and the static RAM. Two 26-pin headers bring out the Rabbit 2000 I/O bus lines, address lines, data lines, parallel ports, and serial ports.

The RCM2200 receives its +5 V power from the user board on which it is mounted. The RabbitCore RCM2200 can interface with all kinds of CMOS-compatible digital devices through the user board.

1.1 RCM2200 Features

- Small size: 1.60" × 2.30" × 0.86"
  (41 mm × 58 mm × 22 mm)
- Microprocessor: Rabbit 2000 running at 22.1 MHz
- 26 parallel I/O lines: 16 configurable for input or output, 7 fixed inputs, 3 fixed outputs
- 8 data lines (D0–D7)
- 4 address lines (A0–A3)
- Memory I/0 read, write
- External reset input
- Five 8-bit timers (cascadable in pairs) and two 10-bit timers
- 256K–512K flash memory, 128K–512K SRAM
- Real-time clock
- Watchdog supervisor

- Provision for customer-supplied backup battery via connections on header J5
- 10/100-compatible RJ-45 Ethernet port with 10Base-T interface (Ethernet jack not installed on all models)
- Raw Ethernet and two associated LED control signals available on 26-pin header
- Three CMOS-compatible serial ports: maximum asynchronous baud rate of 691,200 bps, maximum synchronous baud rate of 5,529,600 bps. One port is configurable as a clocked port.
- Six additional I/O lines are located on the programming port, can be used as I/O lines when the programming port is not being used for programming or in-circuit debugging—one synchronous serial port can also be used as two general CMOS inputs and one general CMOS output, and there are two additional inputs and one additional output.

Appendix A, "RabbitCore RCM2200 Specifications," provides detailed specifications for the RCM2200.

In addition, three different RCM2200 models are available. A variant of the RCM2200, the RCM2300, omits the Ethernet connectivity but offers a much smaller footprint, one-half the size of the RCM2200.

1.2 Advantages of the RCM2200

- Fast time to market using a fully engineered, "ready to run" microprocessor core.
- Competitive pricing when compared with the alternative of purchasing and assembling individual components.
- Easy C-language program development and debugging, including rapid production loading of programs.
- Generous memory size allows large programs with tens of thousands of lines of code, and substantial data storage.
- Integrated Ethernet port for network connectivity, royalty-free TCP/IP software.

1.3 Development and Evaluation Tools

A complete Development Kit, including a Prototyping Board and Dynamic C development software, is available for the RCM2200. The Development Kit puts together the essentials you need to design an embedded microprocessor-based system rapidly and efficiently.

1.3.1 Development Software

The RCM2200 module uses the Dynamic C development environment for rapid creation and debugging of runtime applications. Dynamic C provides a complete development environment with integrated editor, compiler and source-level debugger. It interfaces directly with the target system, eliminating the need for complex and unreliable in-circuit emulators.

NOTE: The RCM2200 module requires Dynamic C v7.04 or later for development. A compatible version is included on the Development Kit CD-ROM.

1.4 Development Kit Contents

The RCM2200 Development Kit contains the following items:

- RCM2200 module with 10Base-T Ethernet port, 256K flash memory, and 128K SRAM.
- RCM2200/RCM2300 Prototyping Board.
- Wall transformer power supply, 12 V DC, 1 A. (Included only with Development Kits sold for the North American market. Overseas users will have to substitute a power supply compatible with local mains power.)
- 10-pin header to DB9 programming cable with integrated level-matching circuitry.
- ***Dynamic C*** CD-ROM, with complete product documentation on disk.
- ***Getting Started*** instructions.
- ***Rabbit 2000 Processor Easy Reference*** poster.
- Registration card.

1.5 How to Use This Manual

This user's manual is intended to give users detailed information on the RCM2200 module. It does not contain detailed information on the Dynamic C development environment or the TCP/IP software support for the integrated Ethernet port. Most users will want more detailed information on some or all of these topics in order to put the RCM2200 module to effective use.

1.5.1 Additional Product Information

In addition to the product-specific information contained in the ***RabbitCore RCM2200 User's Manual*** (this manual), several higher level reference manuals are provided in HTML and PDF form on the accompanying CD-ROM. Advanced users will find these references valuable in developing systems based on the RCM2200 modules:

- ***Dynamic C User's Manual***
- ***An Introduction to TCP/IP***
- ***Dynamic C TCP/IP User's Manual***
- ***Rabbit 2000 Microprocessor User's Manual***

1.5.2 Online Documentation

The online documentation is installed along with Dynamic C, and an icon for the documentation menu is placed on the workstation's desktop. Double-click this icon to reach the menu. If the icon is missing, use your browser to find and load **default.htm** in the **docs** folder, found in the Dynamic C installation folder.

The latest versions of all documents are always available for free, unregistered download from our Web sites as well.

2. GETTING STARTED

This chapter describes the RCM2200 hardware in more detail, and explains how to set up and use the accompanying Prototyping Board.

NOTE: This chapter (and this manual) assume that you have the RCM2200 Development Kit. If you purchased an RCM2200 module by itself, you will have to adapt the information in this chapter and elsewhere to your test and development setup.

2.1 Connections

There are four steps to connecting the Prototyping Board for use with Dynamic C and the sample programs:

1. Attach the RCM2200 module to the Prototyping Board.
2. Connect the programming cable between the RCM2200 module and the workstation PC.
3. Connect the power supply to the Prototyping Board.

2.1.1 Attach Module to Prototyping Board

Turn the RCM2200 module so that the Ethernet connector end of the module extends off the Prototyping Board, as shown in Figure 1 below. Align the pins from headers J4 and J5 on the bottom side of the RCM2200 with header sockets J1 and J2 on the Prototyping Board.

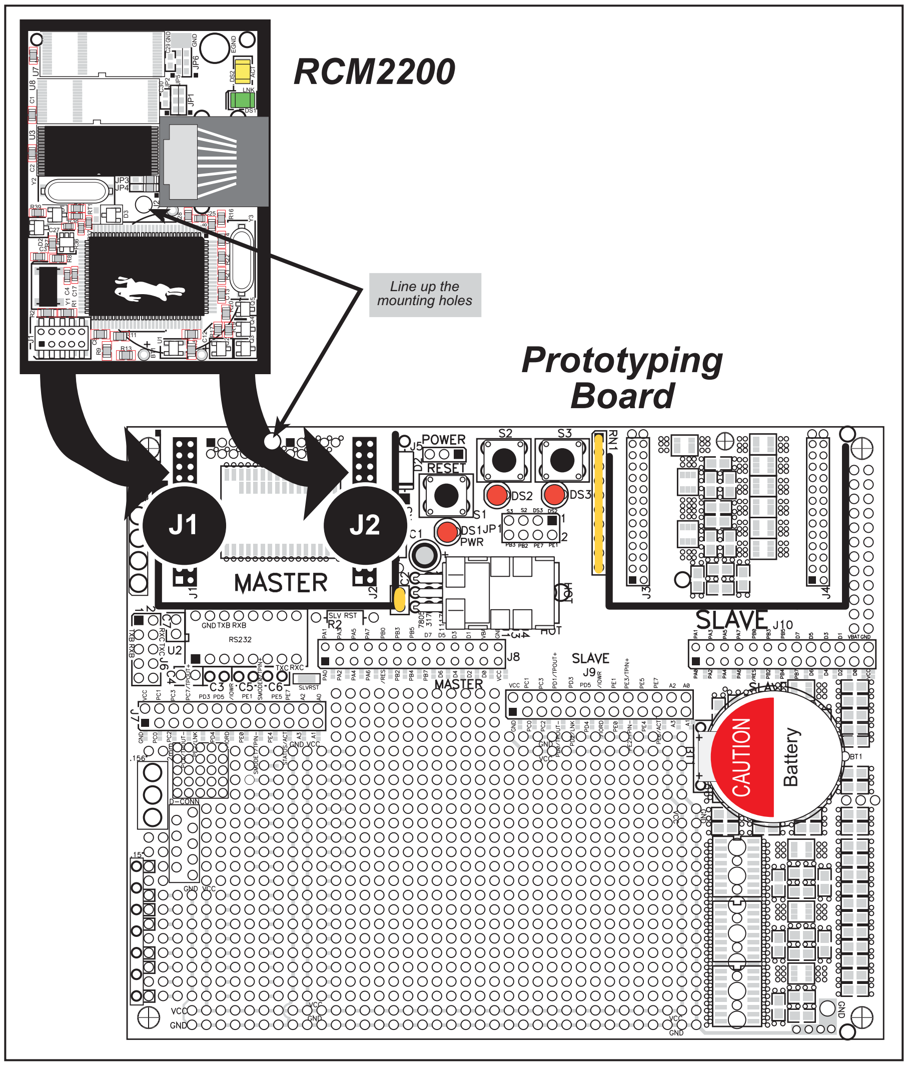


Figure 1. Installing the RCM2200 on the Prototyping Board

Although you can install a single module into either the **MASTER** or the **SLAVE** position on the Prototyping Board, all the Prototyping Board features (switches, LEDs, serial port drivers, etc.) are connected to the **MASTER** position. We recommend you install the module in the **MASTER** position.

> **NOTE:** It is important that you line up the pins on headers J4 and J5 of the RCM2200 exactly with the corresponding pins of header sockets J1 and J2 on the Prototyping Board. The header pins may become bent or damaged if the pin alignment is offset, and the module will not work. Permanent electrical damage to the module may also result if a misaligned module is powered up.

Press the module's pins firmly into the Prototyping Board header sockets.

2.1.2 Connect Programming Cable

The programming cable connects the RCM2200 module to the PC running Dynamic C to download programs and to monitor the RCM2200 for debugging.

Connect the 10-pin connector of the programming cable labeled **PROG** to header J1 on the RCM2200 module as shown in Figure 2. Be sure to orient the marked (usually red) edge of the cable towards pin 1 of the connector. (Do not use the **DIAG** connector, which is used for a normal serial connection.)

Connect the other end of the programming cable to a COM port on your PC. Make a note of the port to which you connect the cable, as Dynamic C needs to have this parameter configured when it is installed.

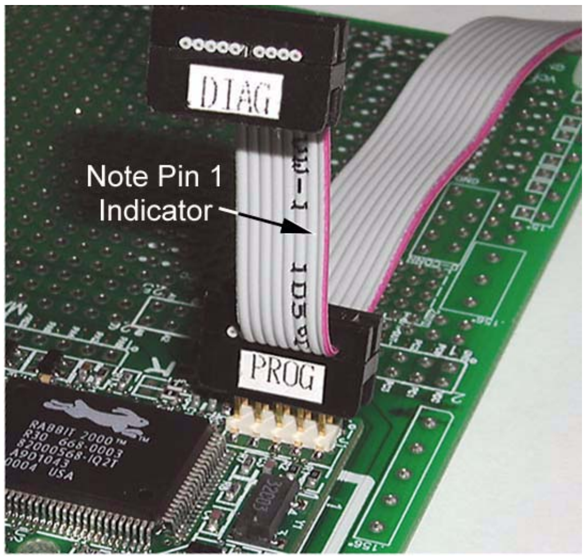


Figure 2. Connect Programming Cable to RCM2200

> **NOTE:** COM 1 is the default port used by Dynamic C.

> **NOTE:** Some PCs now come equipped only with a USB port. It may be possible to use an RS-232/USB converter (Part No. 20-151-0178) with the programming cable supplied with the RCM2200 Development Kit. Note that not all RS-232/USB converters work with Dynamic C.

2.1.3 Connect Power

When all other connections have been made, you can connect power to the RCM2200 Prototyping Board.

First, prepare the AC adapter for the country where it will be used by selecting the plug. The RCM2200 Development Kit presently includes Canada/Japan/U.S., Australia/N.Z., U.K., and European style plugs. Snap in the top of the plug assembly into the slot at the top of the AC adapter as shown in Figure 3, then press down on the spring-loaded clip below the plug assembly to allow the plug assembly to click into place.

Connect the AC adapter to 3-pin header J5 on the Prototyping Board as shown in Figure 3 below. The connector may be attached either way as long as it is not offset to one side.

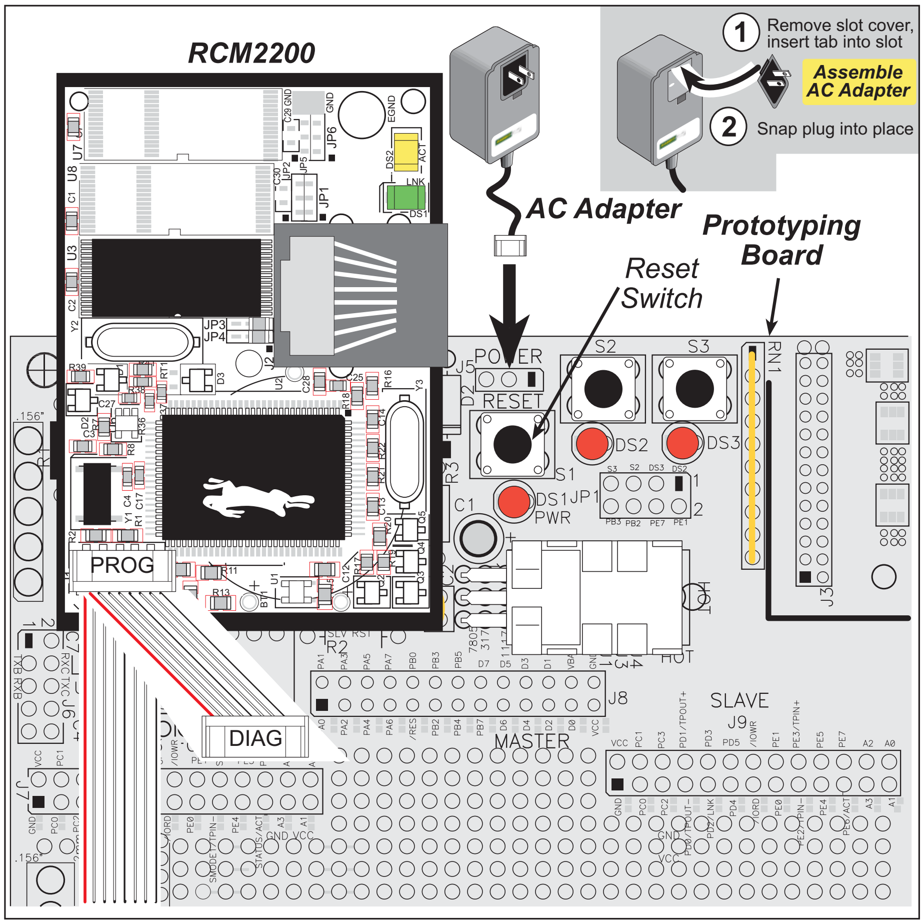


Figure 3. Power Supply Connections

Plug in the AC adapter. The power LED on the Prototyping Board should light up. The RCM2200 and the Prototyping Board are now ready to be used.

> **NOTE:** A **RESET** button is provided on the Prototyping Board to allow hardware reset without disconnecting power.

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.

If Dynamic C appears to compile the BIOS successfully, but you then receive a communication error message when you compile and load the sample program, it is possible that your PC cannot handle the higher program-loading baud rate. Try changing the maximum download rate to a slower baud rate as follows.

- Locate the **Serial Options** dialog in the Dynamic C **Options > Project Options > Communications** menu. Select a slower Max download baud rate.

If a program compiles and loads, but then loses target communication before you can begin debugging, it is possible that your PC cannot handle the default debugging baud rate. Try lowering the debugging baud rate as follows.

- Locate the **Serial Options** dialog in the Dynamic C **Options > Project Options > Communications** menu. Choose a lower debug baud rate.

2.3 Where Do I Go From Here?

If everything appears to be working, we recommend the following sequence of action:

1. Run all of the sample programs described in Chapter 3 to get a basic familiarity with Dynamic C and the RCM2200 module's capabilities.
2. For further development, refer to the ***RabbitCore RCM2200 User's Manual*** for details of the module's hardware and software components.

   A documentation icon should have been installed on your workstation's desktop; click on it to reach the documentation menu. You can create a new desktop icon that points to **default.htm** in the **docs** folder in the Dynamic C installation folder.
3. For advanced development topics, refer to the ***Dynamic C User's Manual*** and the ***Dynamic C TCP/IP User's Manual***, also in the online documentation set.

2.3.1 Technical Support

NOTE: If you purchased your RCM2200 through a distributor or through a Rabbit partner, contact the distributor or partner first for technical support.

If there are any problems at this point:

- Use the Dynamic C **Help** menu to get further assistance with Dynamic C.
- Check the Rabbit Technical Bulletin Board and forums at www.rabbit.com/support/bb/ and at www.rabbit.com/forums/.
- Use the Technical Support e-mail form at www.rabbit.com/support/.

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.

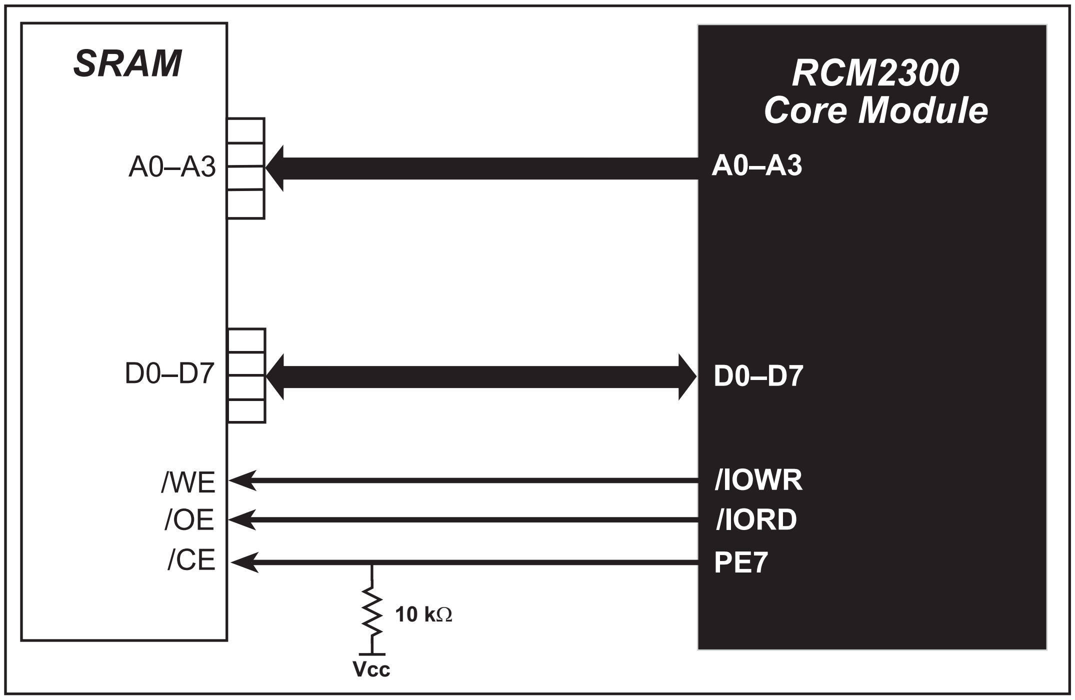
Complete information on Dynamic C is provided in the ***Dynamic C User's Manual***.

3.1.1 Getting to Know the RCM2200

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

- **`EXTSRAM.C`**—demonstrates the setup and simple addressing to an external SRAM. This program first maps the external SRAM to the I/O Bank 7 register with a maximum of 15 wait states, chip select strobe (PE7), and allows writes. The first 256 bytes of SRAM are cleared and read back. Values are then written to the same area and are read back. The Dynamic C **STDIO** window will indicate if writes and reads did not occur

  Connect an external SRAM as shown below before you run this sample program.

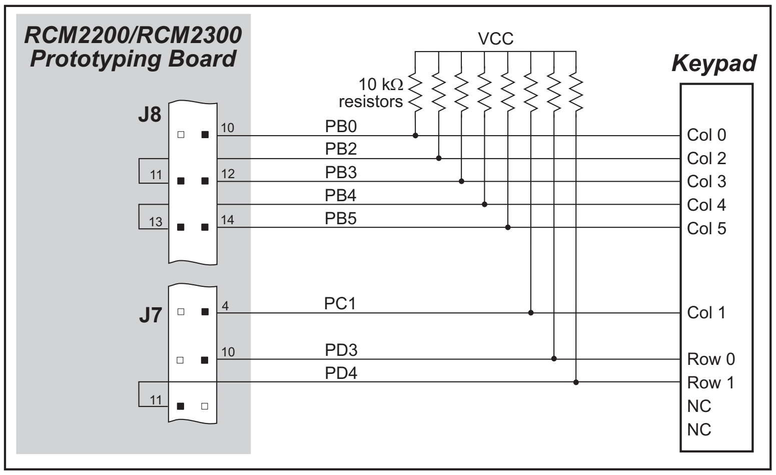


- **`FLASHLED.C`**—repeatedly flashes LED DS3 on the Prototyping Board on and off. LED DS3 is controlled by Parallel Port E bit 7 (PE7). LED DS2 will remain on continuously.
- **`FLASHLEDS.C`**—demonstrates the use of coding with assembly instructions, cofunctions, and costatements to flash LEDs DS2 and DS3 on the Prototyping Board on and off. LEDs DS2 and DS3 are controlled by Parallel Port E bit 1 (PE1) and Parallel Port E bit 7 (PE7).Once you have compiled this program and it is running, LEDs DS2 and DS3 will flash on/off at different rates.
- **`TOGGLELED.C`**—demonstrates the use of costatements to detect switch presses using the press-and-release method of debouncing. As soon as the sample program starts running, LED DS2 on the Prototyping Board (which is controlled by PE1) starts flashing once per second. Press switch S3 on the Prototyping Board (which is connected to PB3) to toggle LED DS3 on the Prototyping Board (which is controlled by PE7) on and off. The pushbutton switch is debounced by the software.

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

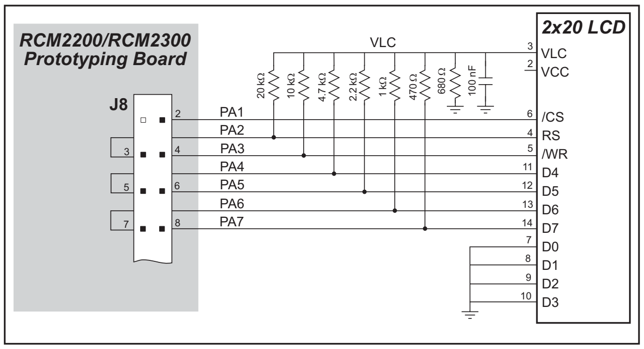
  Connect the keypad to Parallel Ports B, C, and D.

  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

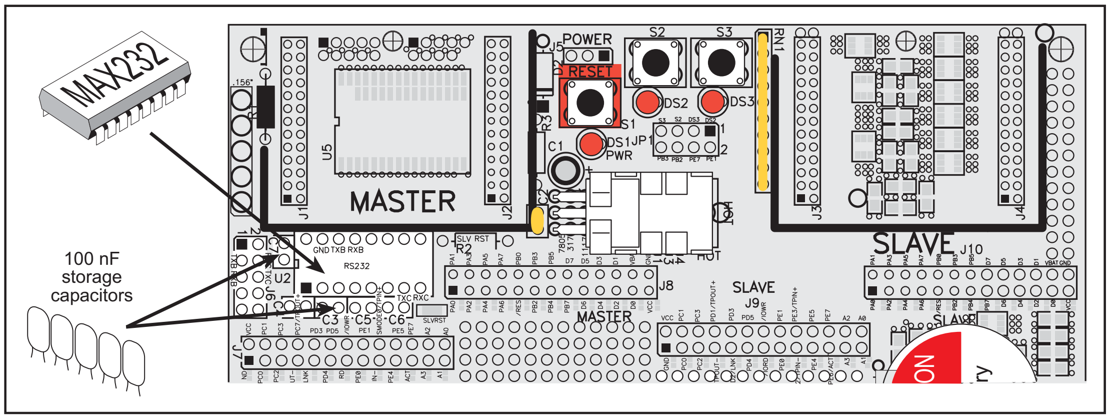


  Once the connections have been made and the sample program is running, the LCD will display two rows of 6 dots, each dot representing the corresponding key. When a key is pressed, the corresponding dot will become an asterisk.

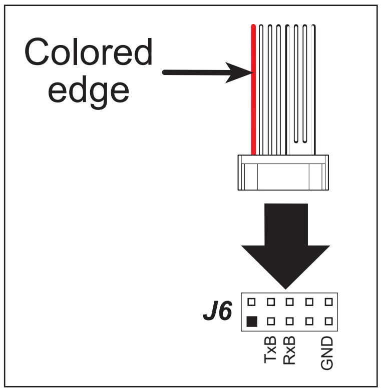
3.1.2 Serial Communication

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

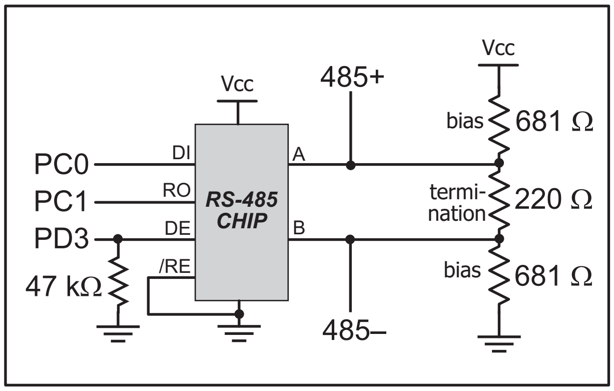
Parity: None

Stop bits: 1

Flow control: None

Two sample programs, **`MASTER.C`** and **`SLAVE.C`**, are available to illustrate RS-485 master/slave communication. To run these sample programs, you will need a second Rabbit-based system with RS-485, and you will also have to add an RS-485 transceiver such as the SP483E and bias resistors to the Prototyping Board.

The diagram shows the connections. You will have to connect PC0 and PC1 (Serial Port D) on the Prototyping Board to the RS-485 transceiver, and you will connect PD3 to the RS-485 transceiver to enable or disable the RS-485 transmitter.



The RS-485 connections between the slave and master devices are as follows.

- RS485+ to RS485+
- RS485– to RS485–
- GND to GND

- **`MASTER.C`**—This program demonstrates a simple RS-485 transmission of lower case letters to a slave RCM2200. The slave will send back converted upper case letters back to the master RCM2200 and display them in the **STDIO** window. Use **`SLAVE.C`** to program the slave RCM2200—reset the slave before you run **`MASTER.C`** on the master.
- **`SLAVE.C`**—This program demonstrates a simple RS-485 transmission of lower case letters to a master RCM2200. The slave will send back converted upper case letters back to the master RCM2200 and display them in the **STDIO** window. Compile and run this program on the slave before you use **`MASTER.C`** to program the master.

3.1.3 Other Sample Programs

Section 6.2 covers how to run the TCP/IP sample programs, which are then described in detail.

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***.

3.1.4.2 FLASHLEDS.C

In addition to Dynamic C's implementation of C-language programming for embedded systems, it supports assembly-language programming for very efficient processor-level control of the module hardware and program flow. This application is similar to `FLASHLED.C` and `TOGGLELED.C`, but uses assembly language for the low-level port control within *cofunctions*, another powerful multitasking tool.

Dynamic C permits the use of assembly language statements within C code. This program creates three functions using assembly language statements, then creates a C cofunction to call two of them. That cofunction is then called within `main()`.

Within each of the C-like functions, the `#asm` and `#endasm` directives are used to indicate the beginning and end of the assembly language statements.

In the function `initialize_ports( )`, port A is initialized to be all outputs while bit 0 of port E is initialized to be an output.

In the function `ledon()`, a 0 is written to the port A bit corresponding to the desired LED (0, which equals DS3, or 1 which equals DS4), turning that LED on. The `ledoff( )` function works exactly the same way except that a 1 is written to the bit, turning the selected LED off.

Finally, in the cofunction `flashled()`, the LED to be flashed, the on time in milliseconds, and the off time in milliseconds are passed as arguments. This function uses an endless `for(;;)` loop to call the `ledon()` and `ledoff()` functions, separated by calls to the wait function `DelayMs()`. This sequence will make the indicated LED flash on and off.

As is proper in C program design, the contents of `main()` are almost trivial. The program first calls `initialize_ports()`, then begins an endless `for(;;)` loop. Within this loop, the program:

1. Calls the library function `hitwd()`, which resets the microprocessor's watchdog timer. (If the watchdog timer is not reset every so often, it will force a hard reset of the system. The purpose is to keep an intermittent program or hardware fault from locking up the system. Normally, this function is taken care of by the virtual driver, but it is called explicitly here).
2. Sets up a costatement which calls two instances of the `flashled()` function, one for each LED. Note that one LED is flashed one second on, one-half second (500 ms) off, while the other is flashed in the reverse pattern.

Note also the `wfd` keyword in the costatement. This keyword (an abbreviation for `waitfordone`, which can also be used) must be used when calling cofunctions. For a complete explanation, see Section 5 and 6 in the ***Dynamic C User's Manual***.

More Information

See the entries for the `hitwd()` and `DelayMs()` functions in the ***Dynamic C User's Manual***, as well as those for the directives `#asm` and `#endasm`. For a complete explanation 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. Hardware Reference

Chapter 2 describes the hardware components and principal hardware subsystems of the RCM2200. Appendix A, "RabbitCore RCM2200 Specifications," provides complete physical and electrical specifications.

4.1 RCM2200 Digital Inputs and Outputs

Figure 4 shows the subsystems designed into the RCM2200.

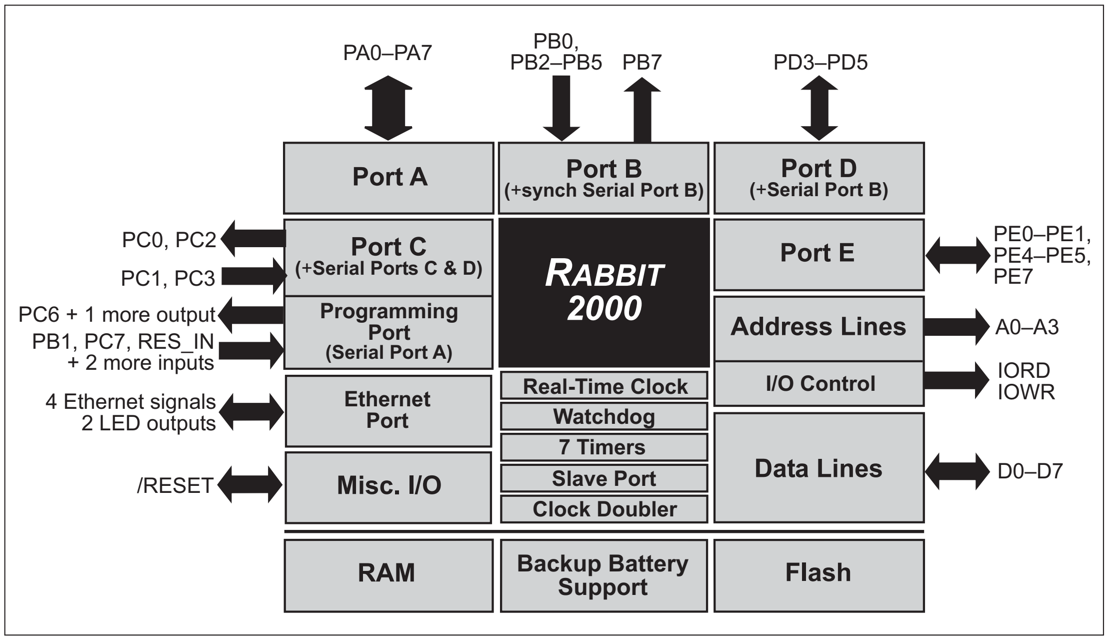


Figure 4. Rabbit Subsystems