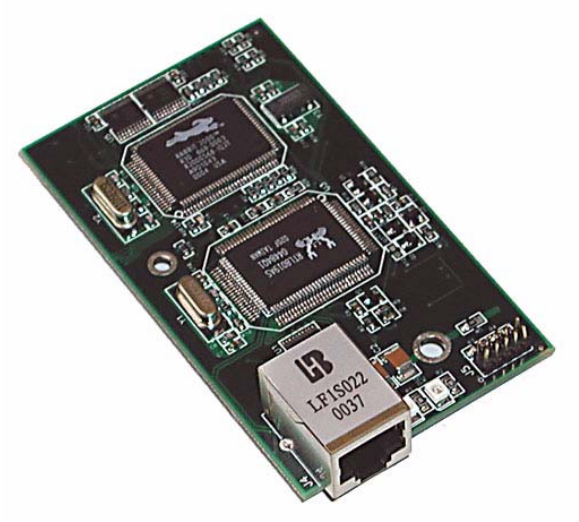
PRODUCT MANUAL



RabbitCore RCM2100

C-Programmable Module with Ethernet

User's Manual

019–0091 • 070831–K

RabbitCore RCM2100 User's Manual

Part Number 019-0091 • 070831–K • Printed in U.S.A.

Trademarks

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

Rabbit 2000 and RabbitCore are trademarks 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

1. INTRODUCTION

The RabbitCore RCM2100 series is a family of microprocessor modules designed to be the heart of embedded control systems. In addition to the array of I/O and addressing available on other Rabbit Semiconductor products, the RCM2100 series offers an optional integrated Ethernet port. These modules permit LAN and Internet-enabled systems to be built as easily as serial communications-only systems.

Throughout this manual, the term RCM2100 refers to the complete series of RCM2100 RabbitCore modules unless other production models are referred to specifically.

The RCM2100 is a microprocessor core module designed to be the heart of your own controller built around the plug-in module. Data processing is done by a Rabbit 2000 microprocessor operating at 22.1 MHz.

The RCM2100 has a Rabbit 2000 microprocessor, a static RAM, up to two flash memory chips, two quartz crystals (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 40-pin headers bring out the Rabbit 2000 I/O bus, address lines, data lines, parallel ports, and serial ports.

The RCM2100 receives its +5 V power from the user board on which it is mounted. The RCM2100 can interface with all kinds of CMOS-compatible digital devices through the user board.

1.1 RCM2100 Features

- Small size: 2.0" × 3.5" × 0.80"
  (51 mm × 89 mm × 20 mm)
- Microprocessor: Rabbit 2000 running at 22.1 MHz
- 34 CMOS-compatible parallel I/O lines grouped in five 8-bit ports (shared with serial ports)
- 8 data lines (BD0–BD7)
- 13 address lines (BA0–BA12)
- I/0 read, write, buffer enable
- Status, watchdog and clock outputs
- Two startup mode inputs for booting and master/slave configuration
- External reset input
- Reset output
- Five 8-bit timers, two 10-bit timers; five timers are cascadable in pairs
- 2 × 256K flash memory, 512K SRAM
- Real-time clock
- Watchdog supervisor
- Provision for customer-supplied backup battery via connections on header J2
- Four CMOS-compatible serial ports: maximum asynchronous baud rate of 690,625 bps, maximum synchronous baud rate of 5.52 Mbps. Two ports are configurable as clocked ports.

Appendix A, “RabbitCore RCM2100 Specifications,” provides detailed specifications for the RabbitCore RCM2100 modules.

Four versions of the RabbitCore RCM2100 are available. Their standard features are summarized in Table 1.

Table 1. RCM2100 Production Models

Model	Features
RCM2100	Full-featured module including 10/100-compatible Ethernet port with 10Base-T interface
RCM2110	RCM2100 with 128K SRAM, 256K flash memory
RCM2120	RCM2100 without Ethernet
RCM2130	RCM2110 without Ethernet

1.2 Advantages of the RCM2100

- 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 (on selected models) for network connectivity, royalty-free TCP/IP software.
- Models with and without Ethernet for flexible production options.
- Small size and identical footprint and pinout for all models.

1.3 Development and Evaluation Tools

A complete Development Kit, including a Prototyping Board, accessory components and Dynamic C development software, is available to accompany the RCM2100 module. 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 RCM2100 modules use 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 RCM2100 modules require Dynamic C v7.04 or later for development. A compatible version is included on the Development Kit CD-ROM.

1.3.2 Development Kit Contents

The RCM2100 Development Kit contains the following items:

- RCM2100 module with 10Base-T Ethernet port, 512K flash memory and 512K SRAM.
- RCM2100 Prototyping Board with accessory hardware and components.
- 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 their 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.
- Registration card.

1.4 How to Use This Manual

This user's manual is intended to give users detailed information on the RCM2100 modules. 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 RCM2100 module to effective use.

1.4.1 Additional Product Information

In addition to the product-specific information contained in the ***RabbitCore RCM2100 User's 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 RCM2100 modules:

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

1.4.2 Using Online Documentation

We provide the bulk of our user and reference documentation in two electronic formats, HTML and Adobe PDF. We do this for several reasons.

We believe that providing all users with our complete library of product and reference manuals is a useful convenience. However, printed manuals are expensive to print, stock, and ship. Rather than include and charge for manuals that every user may not want, or provide only product-specific manuals, we choose to provide our complete documentation and reference library in electronic form with every Development Kit and with our Dynamic C development environment.

> **NOTE:** The most current version of Adobe Acrobat Reader can always be downloaded from Adobe's web site at **http://www.adobe.com**. We recommend that you use version 5.0 or later.

Providing this documentation in electronic form saves an enormous amount of paper by not printing copies of manuals that users don't need. It reduces the number of outdated manuals we have to discard from stock as well, and it makes providing a complete library of manuals an almost cost-free option. For one-time or infrequent reference, electronic documents are more convenient than printed ones—after all, they aren't taking up shelf or desk space!

Finding Online Documents

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.

Printing Electronic Manuals

We recognize that many users prefer printed manuals for some uses. Users can easily print all or parts of those manuals provided in electronic form. The following guidelines may be helpful:

- Print from the Adobe PDF versions of the files, not the HTML versions.
- Print only the sections you will need to refer to more than once.
- Print manuals overnight, when appropriate, to keep from tying up shared resources during the work day.
- If your printer supports duplex printing, print pages double-sided to save paper and increase convenience.
- If you do not have a suitable printer or do not want to print the manual yourself, most retail copy shops (e.g., Kinkos, AlphaGraphics, CopyMax) will print the manual from the PDF file and bind it for a reasonable charge—about what we would have to charge for a printed and bound manual.

2. GETTING STARTED

This chapter describes the RCM2100 hardware in more detail, and explains how to set up and use the accompanying prototyping and development board.

NOTE: This chapter (and this manual) assume that you have the RabbitCore RCM2100 Development Kit. If you purchased an RCM2100 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 three steps to connecting the Prototyping Board for use with Dynamic C and the sample programs:

1. Attach the RCM2100 module to the Prototyping Board.
2. Connect the programming cable between the RCM2100 module and the workstation PC.
3. Connect the power supply to the Prototyping Board.

2.1.1 Attach Module to Prototyping Board

Turn the RCM2100 module so that the Ethernet connector is on the left, as shown in Figure 1 below. Align the module headers J1 and J2 on the bottom side of the RCM2100 into header sockets J1 and J3 on the Prototyping Board.

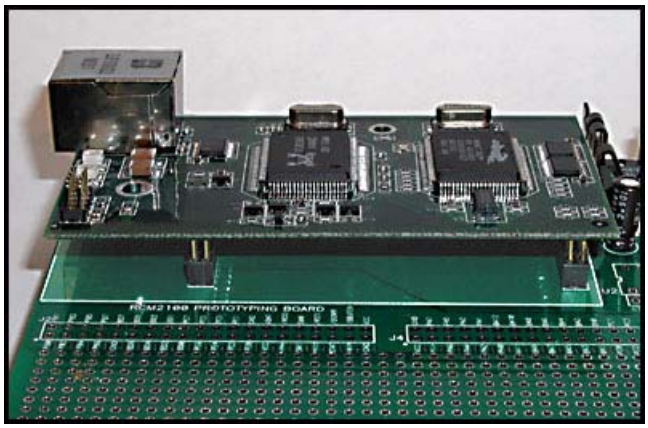


Figure 1. Installing the RCM2100 Module on the Prototyping Board. Note the orientation of the module.

NOTE: It is important that you line up the RCM2100 pins on headers J1 and J2 exactly with the corresponding pins of header sockets J1 and J3 on the Prototyping Board. The header pins may become bent or damaged if the pin alignment is offset, and the module will not work.

Press the module's pins firmly into the Prototyping Board header sockets. The installed module is shown in Figure 2 below.

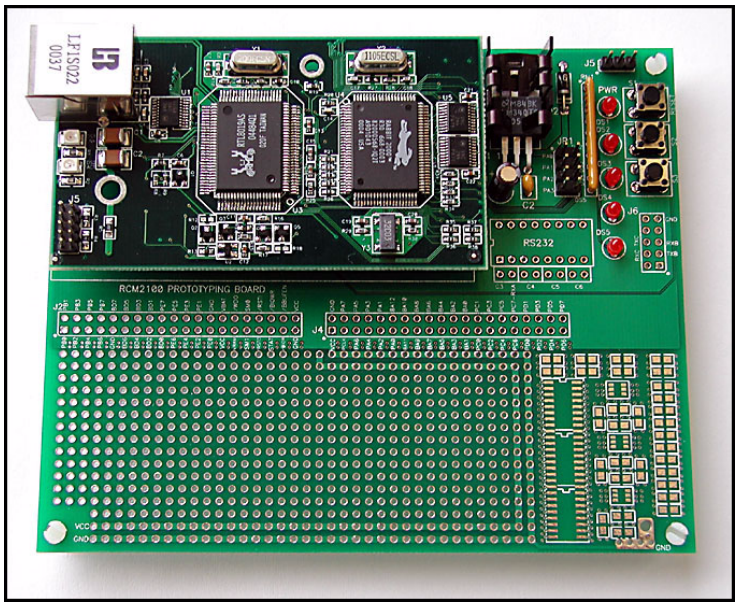


Figure 2. RCM2100 Installed and Seated on the Prototyping Board

2.1.2 Connect Programming Cable

The programming cable connects the RCM2100 module to the PC running Dynamic C, to download programs and to monitor the RCM2100 for debugging.

Connect the 10-pin connector of the programming cable labeled **PROG** to header J5 on the RCM2100 module as shown in Figure 3 below. Be sure to orient the 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.)

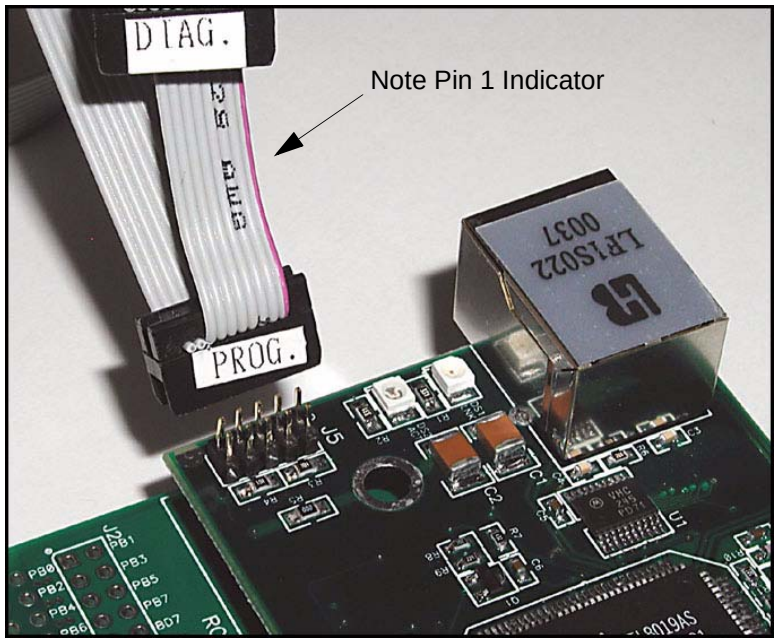


Figure 3. Attaching Programming Cable to the RCM2100

NOTE: The stripe on the cable is towards pin 1 of the header J5.

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.

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. 540-0070) with the programming cable supplied with the RCM2100 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 RCM2100 Prototyping Board.

Hook the connector from the wall transformer to header J5 on the Prototyping Board as shown in Figure 4 below. The connector may be attached either way as long as it is not offset to one side.

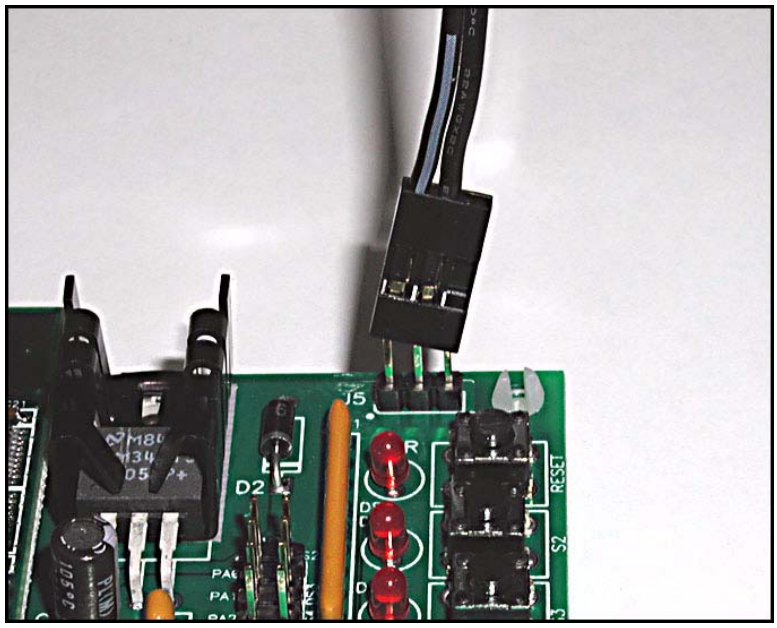


Figure 4. Power Supply Connections to Prototyping Board

Plug in the wall transformer. The power LED on the Prototyping Board should light up. The RCM2100 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 RCM2100 module from the Prototyping Board.

2.2 Run a Sample Program

Once the RCM2100 is connected as described in the preceding pages, start Dynamic C by double-clicking on the Dynamic C icon or by double-clicking on **dcrab_XXXX.exe** in the Dynamic C root directory, where **XXXX** are version-specific characters. Dynamic C uses the serial port specified during installation.

If you are using a USB port to connect your computer to the RCM2100 module, choose **Options > Project Options** and select "Use USB to Serial Converter" under the **Communications** tab.

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 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.

If there are any other problems:

- Check to make sure you are using the **PROG** connector, not the **DIAG** connector, on the programming cable.
- Check both ends of the programming cable to ensure that they are firmly plugged into the PC and the programming port on the RCM2100.
- Ensure that the RCM2100 module is firmly and correctly installed in its connectors on the Prototyping Board.
- 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 active COM port.

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 4 to get a basic familiarity with Dynamic C and the RabbitCore module's capabilities.
2. For further development, refer to the ***RabbitCore RCM2100 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 RCM2100 through a distributor or through a Rabbit Semiconductor 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 Semiconductor Technical Bulletin Board at www.rabbit.com/support/bb/.
- Use the Technical Support e-mail form at www.rabbit.com/support/.

3. RUNNING SAMPLE PROGRAMS

To develop and debug programs for the RCM2100 (and for all other Rabbit Semiconductor hardware), you must install and use 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 single-board computers based on the Rabbit microprocessor. This chapter takes you through the installation of Dynamic C, and then provides a tour of the sample programs for the RCM2100.

3.1 Sample Programs

To help familiarize you with the RCM2100 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 RCM2100'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 programs, make sure that your RCM2100 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 RabbitCore RCM2100.

- **RCM2100**—Demonstrates the basic operation and the Ethernet functionality of the RabbitCore RCM2100.
- **TCPIP**—Demonstrates more advanced TCP/IP programming for Rabbit Semiconductor's Ethernet-enabled Rabbit-based boards.

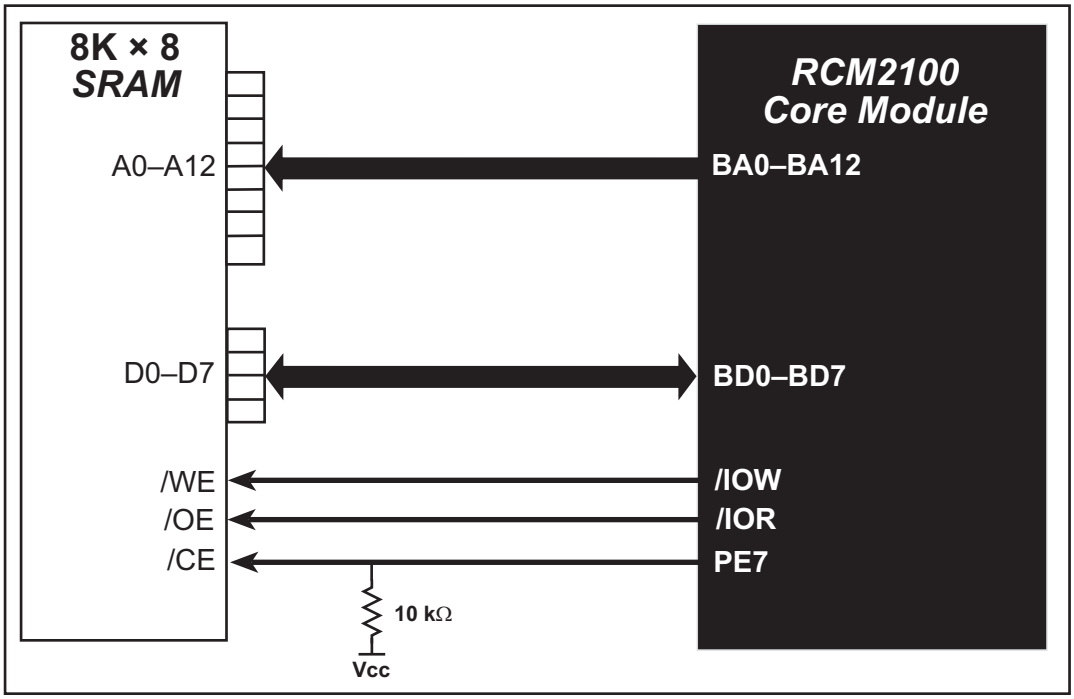
Complete information on Dynamic C is provided in the ***Dynamic C User's Manual***.

3.1.1 Getting to Know the RCM2100

The following sample programs can be found in the **SAMPLES\RCM2100** 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 A bit 1 (PA1).
- **FLASHLED2.C**—repeatedly flashes LED DS3 on the Prototyping Board on and off. LED DS3 is controlled by Parallel Port A bit 1 (PA1).

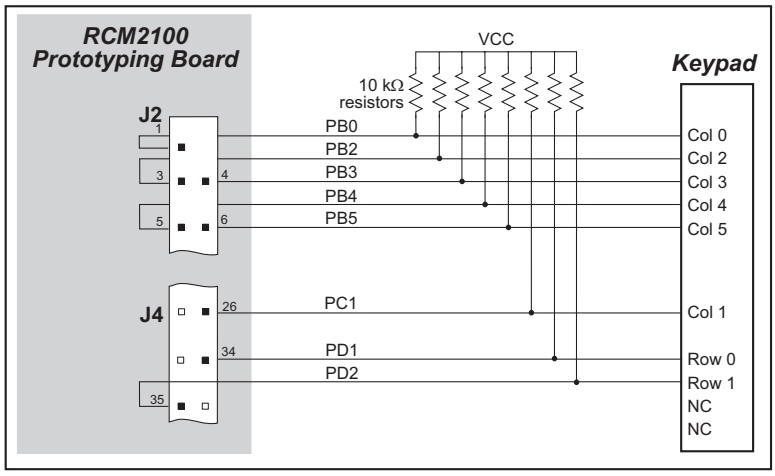
  This sample program also shows the use of the **runwatch()** function to allow Dynamic C to update watch expressions while running. The following steps explain how to do this.

  1. Add a watch expression for "k" in the **Inspect > Add Watch** dialog box.
  2. Click "Add" or "Add to top" so that it will be in the watch list permanently.
  3. Click **OK** to close the dialog box.
  4. Press **<Ctrl+U>** while the program is running. This will update the watch window.

- **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 A bit 0 (PA0) and Parallel Port A bit 1 (PA1).Once you have compile this program and it is running, LEDs DS2 and DS3 will flash on/off at different rates.
- **FLASHLEDS2.C**—demonstrates the use of 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 A bit 0 (PA0) and Parallel Port A bit 1 (PA1).Once you have compile this program and it is running, LEDs DS2 and DS3 will flash on/off at different rates.
- **KEYLCD2.C**—demonstrates a simple setup for a 2 × 6 keypad and a 2 × 20 LCD.

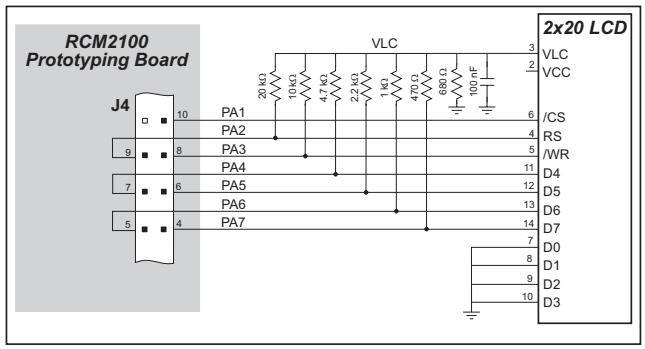
  Connect the keypad to Parallel Ports B, C, and D.

PB0—Keypad Col 0
PC1—Keypad Col 1
PB2—Keypad Col 2
PB3—Keypad Col 3
PB4—Keypad Col 4
PB5—Keypad Col 5
PD1—Keypad Row 0
PD2—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

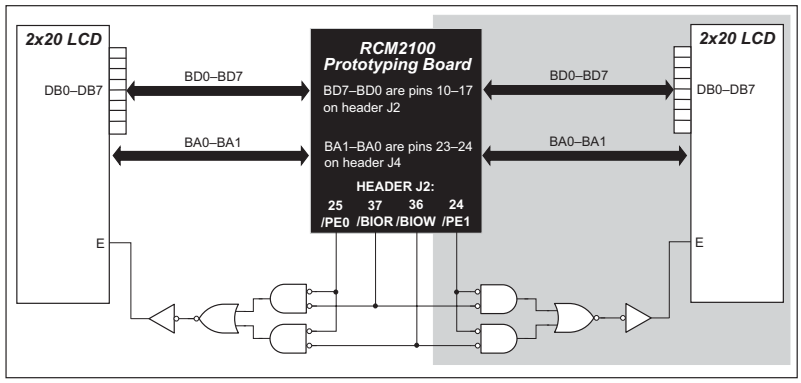


  Once the connections have been made and the sample program isrunning, the LCD will display two rows of 6 dots, each dot representing the corresponding key. When a key is pressed, the corresponding dot will become an asterisk.

- **LCD_DEMO.C**—demonstrates a simple setup for an LCD that uses the HD44780 controller or an equivalent.

  Connect the LCD to the RCM2100 address and data lines on the Prototyping Board.

  BD0—DB0
  BD1—DB1
  BD2—DB2
  BD3—DB3
  BD4—DB4
  BD5—DB5
  BD6—DB6
  BD7—DB7



  BA0—RS (Register Select: 0 = command, 1 = data)
  BA1—R/W (0=write, 1=read)
  *—E (normally low: latches on high-to-low transition)

- **SWTEST.C**—demonstrates the use of pushbutton switches S2 and S3 to toggle LEDs DS2 and DS3 on the Prototyping Board on and off.

  Parallel Port A bit 0 = LED DS2
  Parallel Port A bit 1 = LED DS3

  Parallel Port B bit 2 = switch S2
  Parallel Port B bit 3 = switch S3

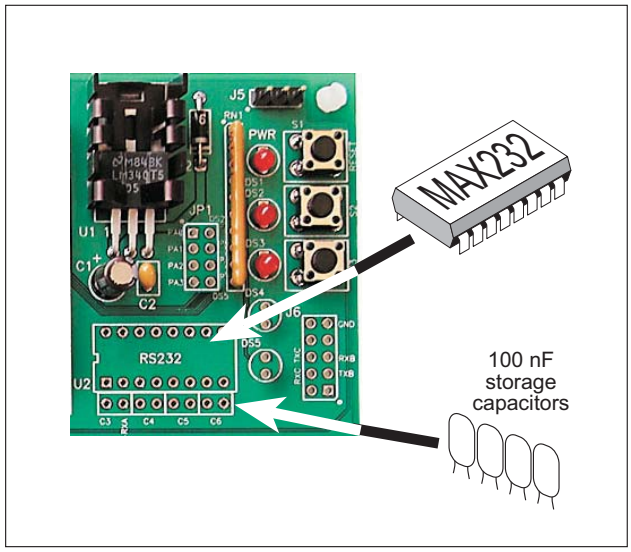
- **TOGGLELED.C**—demonstrates the use of costatements to detect switch presses using the press-and-release method of debouncing. As soon as the sample program starts running, LED DS3 on the Prototyping Board (which is controlled by PA1) starts flashing once per second. Press switch S2 on the Prototyping Board (which is connected to PB2) to toggle LED DS2 on the Prototyping Board (which is controlled by PA0). The pushbutton switch is debounced by the software.

3.1.2 Serial Communication

The following sample programs can be found in the **SAMPLES\RCM2100** folder.

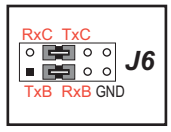
Two sample programs, **CORE_FLOW-CONTROL.C** and **CORE_PARITY.C**, are available to illustrate RS-232 communication. To run these sample programs, you will have to add an RS-232 transceiver such as the MAX232 at location U2 and four 100 nF capacitors at C3–C6 on the Prototyping Board. Also install the 2 × 5 IDC header included with the Prototyping Board accessory parts at J6 to interface the RS-232 signals.

The diagram shows the connections.



- **CORE_FLOWCONTROL.C**—This program demonstrates hardware flow control by configuring Serial Port C (PC3/PC2) for CTS/RTS with serial data coming from TxB at 115,200 bps. One character at a time is received and is displayed in the **STDIO** window.

  To set up the Prototyping Board, you will need to tie PC4 and PC5 (TxB and RxB) together at header J4, and you will also tie PC2 and PC3 (TxC and RxC) together using the jumpers supplied in the Development Kit as shown in the diagram.

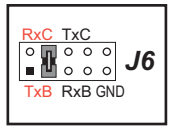
  

  A repeating triangular pattern should print out in the **STDIO** window. The program will periodically switch flow control on or off to demonstrate the effect of no flow control.

  Refer to the **serBflowcontrolOn()** function call in the ***Dynamic C Function Reference Manual*** for a general description on how to set up flow control lines.

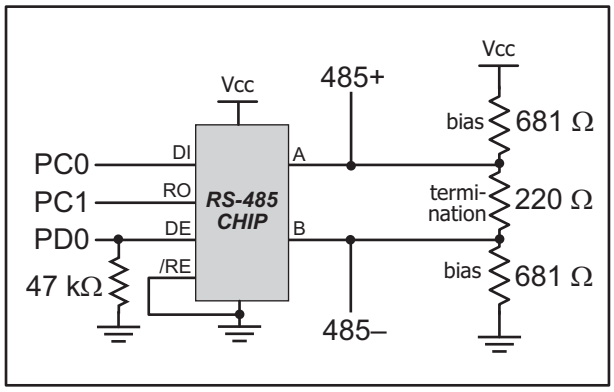
- **CORE_PARITY.C**—This program demonstrates the use of parity modes by repeatedly sending byte values 0–127 from Serial Port B to Serial Port C. The program will switch between generating parity or not on Serial Port B. Serial Port C will always be checking parity, so parity errors should occur during every other sequence.

  To set up the Prototyping Board, you will need to tie PC4 and PC3 (TxB and RxC) together at header J4 using the jumpers supplied in the Development Kit as shown in the diagram.

  The Dynamic C **STDIO** window will display the error sequence.

Two sample programs, **MASTER2.C** and **SLAVE2.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 PD0 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

- **MASTER2.C**—This program demonstrates a simple RS-485 transmission of lower case letters to a slave RCM2100. The slave will send back converted upper case letters back to the master RCM2100 and display them in the **STDIO** window. Use **SLAVE2.C** to program the slave RCM2100.
- **SLAVE2.C**—This program demonstrates a simple RS-485 transmission of lower case letters to a master RCM2100. The slave will send back converted upper case letters back to the master RCM2100 and display them in the **STDIO** window. Use **MASTER2.C** to program the master RCM2100.

3.1.3 Other Sample Programs

Section 6.7 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 **TOGGLELEDS.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 explana-

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 interation. 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 3 describes the hardware components and principal hardware subsystems of the RabbitCore RCM2100. Appendix A, "RabbitCore RCM2100 Specifications," provides complete physical and electrical specifications.

4.1 RCM2100 Digital Inputs and Outputs

Figure 5 shows the subsystems designed into the RCM2100 modules.

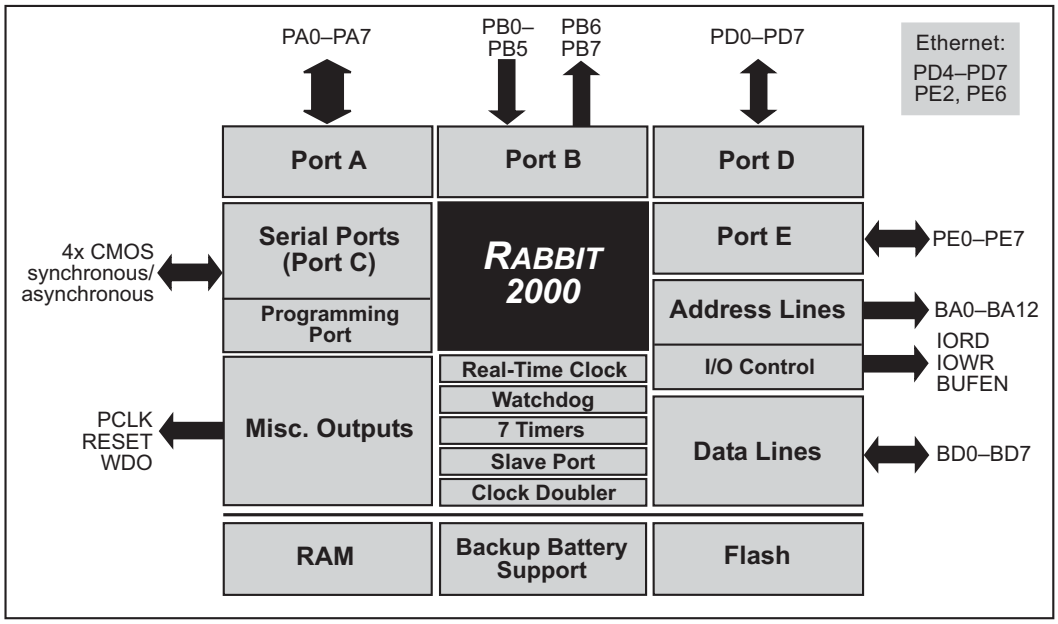


Figure 5. Rabbit Subsystems

The RCM2100 has 40 parallel I/O lines grouped in five 8-bit ports available on headers J1 and J2. The 24 bidirectional I/O lines are located on pins PA0–PA7, PD0–PD7, and PE0–PE7. The pinouts for headers J1 and J2 are shown in Figure 6.

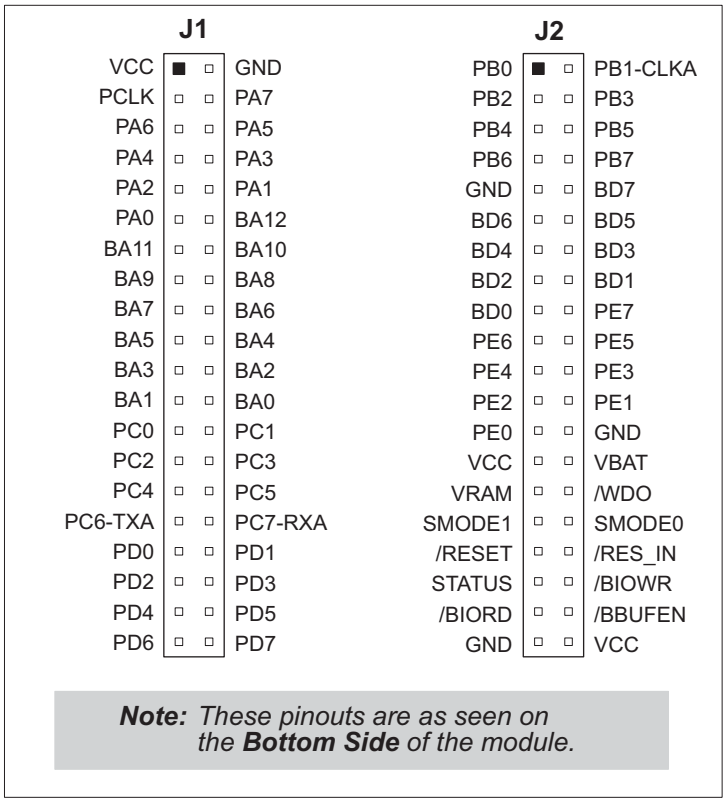


Figure 6. RCM2100 I/O Pinouts

The ports on the Rabbit 2000 microprocessor used in the RCM2100 are configurable, and so the factory defaults can be reconfigured. Table 2 lists the Rabbit 2000 factory defaults and the alternate configurations.

As shown in Table 2, pins PA0–PA7 can be used to allow the Rabbit 2000 to be a slave to another processor. PE0, PE1, PE4, and PE5 can be used as external interrupts INT0A, INT1A, INT0B, and INT1B. Pins PB0 and PB1 can be used to access the clock on Serial Port B and Serial Port A of the Rabbit microprocessor. Pins PD4 and PD6 can be programmed to be optional serial outputs for Serial Ports B and A. PD5 and PD7 can be used as alternate serial inputs by Serial Ports B and A.

The Ethernet-enabled versions of the RCM2100 do not have 0 Ω resistors (jumpers) installed at R21, R24, and R35–R38, which allows PE6, PE2, and PD4–PD7 to connect to the RealTek Ethernet chip that is stuffed on those versions.

Table 2. RCM2100 Pinout Configurations

Pin		Pin Name	Default Use	Alternate Use	Notes
Header J1	1	VCC			
	2	GND			
	3	PCLK	Output (Internal Clock)	Output	Turned off in software
	4–11	PA[7:0]	Parallel I/O	Slave port data bus SD0–SD7	
	12–24	BA[12:0]	Output		Buffered Rabbit 2000 address bus
	25	PC0	Output	TXD	
	26	PC1	Input	RXD	
	27	PC2	Output	TXC	
	28	PC3	Input	RXC	
	29	PC4	Output	TXB	
	30	PC5	Input	RXB	
	31	PC6	Output	TXA	Connected to programming port
	32	PC7	Input	RXA	
	33–36	PD[0:3]	Bitwise or parallel programmable I/O, can be driven or open-drain output		16 mA sourcing and sinking current at full AC switching speed
	37	PD4		ATXB output	Ethernet chip RSTDRV
	38	PD5		ARXB input	Ethernet chip BD5
	39	PD6		ATXA output	Ethernet chip BD6
	40	PD7		ARXA input	Ethernet chip BD7

Table 2. RCM2100 Pinout Configurations (continued)

Pin		Pin Name	Default Use	Alternate Use	Notes
Header J2	1	PB0	Input	Serial port clock CLKB	
	2	PB1	Input	Serial port clock CLKA	CLKA is connected to programming port (header J5, pin 3)
	3	PB2	Input	Slave port write /SWR	
	4	PB3	Input	Slave port read /SRD	
	5	PB4	Input	SA0	Slave port address lines
	6	PB5	Input	SA1	
	7	PB6	Output		
	8	PB7	Output	Slave port attention line /SLAVEATTN	
	9, 26, 39	GND			
	10–17	BD[7:0]	Input/Output		Buffered Rabbit 2000 data bus
	18	PE7	Bitwise or parallel programmable I/O	I7 output or slave port chip select /SCS	
	19	PE6		I6 output	Ethernet chip IOWB
	20	PE5		I5 output or INT1B input	
	21	PE4		I4 output or INT0B input	
	22	PE3		I3 output	
	23	PE2		I2 output	Ethernet chip IORB
	24	PE1		I1 output or INT1A input	
	25	PE0		I0 output or INT0A input	
	27, 40	VCC			
	28	VBAT	3 V battery input		
	29	VRAM	2.1 V output		100 µA maximum current draw
	30	/WDO	Output (Watchdog output)	May also be used to output a 30 µs pulse	Outputs a pulse when the internal watchdog times out

Table 2. RCM2100 Pinout Configurations (continued)

Pin		Pin Name	Default Use	Alternate Use	Notes
Header J2	31–32	SMODE1, SMODE0	(0,0)—start executing at address zero		No programming cable attached
			SMODE0 =1, SMODE1 = 1 Cold boot from asynchronous serial port A at 2400 bps (programming cable connected)	(0,1)—cold boot from slave port (1,0)—cold boot from clocked serial port A	With programming cable attached
	33	/RESET	Reset output		
	34	/RES_IN	Reset input		
	35	STATUS	Output (Status)	Output	
	36	/BIOWR	Output (I/O buffer write strobe)		
	37	/BIORD	Output (I/O buffered strobe)		
	38	/BUFEN	Output (I/O buffer enable)		

4.1.1 Dedicated Inputs

PB0 and PB1 are designated as inputs because the Rabbit 2000 is operating in an asynchronous mode. Four of the input-only pins are located on PB2–PB5. These pins are used for the slave port. PB2 and PB3 are slave write and slave read strobes, while PB4 and PB5 serve as slave address lines SA0 and SA1, and are used to access the slave registers (SD0–SD7), which is the alternate assignment for parallel port A. When Port C is used as a parallel port, PC1, PC3, PC5, and PC7 are inputs only. These pins can alternately be selectively enabled to serve as the serial data inputs for Serial Ports D, C, B, and A.

4.1.2 Dedicated Outputs

Two of the output-only pins are located on PB6–PB7. PB7 can also be used with the slave port as the /SLAVEATTN output. This configuration signifies that the slave is requesting attention from the master. When Port C is used as a parallel port, PC0, PC2, PC4 and PC6 are outputs only. These pins can alternately serve as the serial data outputs for Serial Ports D, C, B, and A.

4.1.3 Memory I/O Interface

Thirteen of the Rabbit 2000 buffered address lines (A0–A12) and all the buffered data lines (D0–D7) are available as outputs. I/0 write (/IOWR), I/0 read (/IORD), buffer enable (/BUFEN), and Watchdog Output (/WDO) are also available for interfacing to external devices.

The STATUS output has three different programmable functions:

1. It can be driven low on the first op code fetch cycle.
2. It can be driven low during an interrupt acknowledge cycle.
3. It can also serve as a general-purpose output.

The output clock is available on the PCLK pin. The primary function of PCLK is as a peripheral clock or a peripheral clock ÷ 2, but PCLK can instead be used as a digital output. PCLK can also be disabled by removing R20 if there is a need to reduce radiated emissions. Removing R20 will disable the PCLK output on pin 3 of header J1. Alternatively, PCLK can be disabled in software using Dynamic C version 7.03 or later.

4.1.4 Additional I/0

Two status mode pins, SMODE0 and SMODE1, are available as inputs. The logic state of these two pins determines the startup procedure after a reset.

/RES_IN is an external input used to reset the Rabbit 2000 microprocessor and the RabbitCore RCM2100 memory. /RES_OUT is an output from the reset circuitry that can be used to reset other peripheral devices.

4.2 Serial Communication

The RCM2100 board does not have an RS-232 or an RS-485 transceiver directly on the board. However, an RS-232 or RS-485 interface may be incorporated on the board the RCM2100 is mounted on. For example, the Prototyping Board supports a standard RS-232 transceiver chip.

4.2.1 Serial Ports

There are four serial ports designated as Serial Ports A, B, C, and D. All four serial ports can operate in an asynchronous mode up to the baud rate of the system clock divided by 32. An asynchronous port can handle 7 or 8 data bits. A 9th bit address scheme, where an additional bit is sent to mark the first byte of a message, is also supported. Serial Ports A and B can be operated alternately in the clocked serial mode. In this mode, a clock line synchronously clocks the data in or out. Either of the two communicating devices can supply the clock. When the Rabbit 2000 provides the clock, the baud rate can be up to 1/4 of the system clock frequency, or 5.52 Mbps for a 22.1 MHz clock speed.

4.2.2 Ethernet Port

Figure 7 shows the pinout for the RJ-45 Ethernet port (J4). Note that there are two standards for numbering the pins on this connector—the convention used here, and numbering in reverse to that used here.

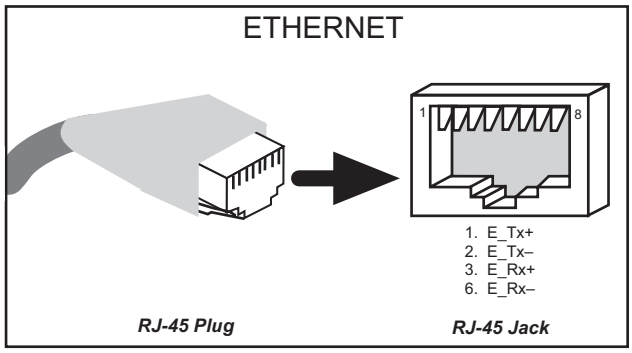


Figure 7. RJ-45 Ethernet Port Pinout

The transformer/connector assembly ground is connected to the RCM2100 printed circuit board digital ground via a 0 Ω resistor "jumper," R5, as shown in Figure 8.

The factory default is for the 0 Ω resistor "jumper" at R5 to be installed. In high-noise environments, it may be useful to ground the transformer/connector assembly directly through the chassis ground. This will be especially helpful to minimize EMI problems.

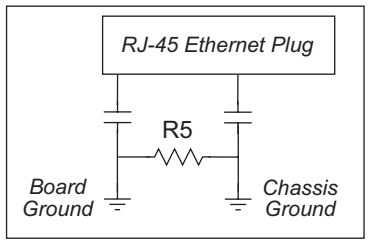


Figure 8. Isolation Resistor R5

Once you have removed the 0 Ω resistor "jumper," R5, use a screw in the position indicated in Figure 9 to attach the RCM2100 board to the chassis ground, thereby grounding the transformer/connector assembly.

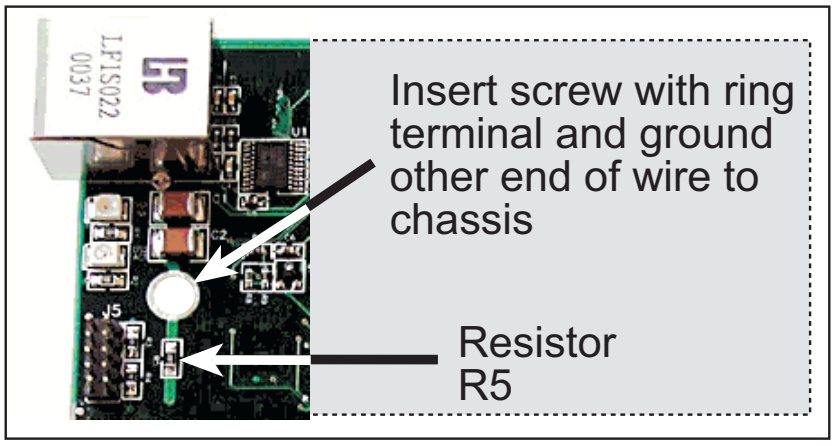


Figure 9. R5 and Chassis Ground Locations

The RCM2100 is available in quantity without the transformer/connector assembly and the **ACT** and **LNK** LEDs (shown to the right of the transformer/connector assembly in Figure 9 above) installed. The Ethernet signals and the LED control signals are then available on header J3 installed on the bottom side of the board for this option, and J3 may then be plugged in to the rest of the system. An Ethernet transformer and LEDs should be included on the board that the modified RCM2100 is plugged into.

NOTE: Contact your Rabbit Semiconductor Sales Representative for quantity and pricing information related to this option.

4.2.3 Programming Port

The RCM2100 has a 10-pin program header labeled J5. The programming port uses the Rabbit 2000's Serial Port A for communication. Dynamic C uses the programming port to download and debug programs.

The programming port is also used for the following operations.

- Cold-boot the Rabbit 2000 after a reset.
- Remotely download and debug a program over an Ethernet connection using the RabbitLink EG2110.
- Fast copy designated portions of flash memory from one Rabbit-based board (the master) to another (the slave) using the Rabbit Cloning Board.

Alternate Uses of the Serial Programming Port

All three clocked Serial Port A signals are available as

- a synchronous serial port
- an asynchronous serial port, with the clock line usable as a general CMOS input

The serial programming port may also be used as a serial port via the **DIAG** connector on the serial programming cable.

In addition to Serial Port A, the Rabbit 2000 startup-mode (SMODE0, SMODE1), status, and reset pins are available on the serial programming port.

The two startup mode pins determine what happens after a reset—the Rabbit 2000 is either cold-booted or the program begins executing at address 0x0000. These two SMODE pins can be used as general inputs once the cold boot is complete.

The status pin is used by Dynamic C to determine whether a Rabbit microprocessor is present. The status output has three different programmable functions:

1. It can be driven low on the first op code fetch cycle.
2. It can be driven low during an interrupt acknowledge cycle.
3. It can also serve as a general-purpose output.

The /RESET_IN pin is an external input that is used to reset the Rabbit 2000 and the onboard peripheral circuits on the RabbitCore module. The serial programming port can be used to force a hard reset on the RabbitCore module by asserting the /RESET_IN signal.

Refer to the ***Rabbit 2000 Microprocessor User's Manual*** for more information..

4.3 Serial Programming Cable

The programming cable is used to connect the RCM2100's programming port to a PC serial COM port. The programming cable converts the RS-232 voltage levels used by the PC serial port to the TTL voltage levels used by the Rabbit 2000.

When the **PROG** connector on the programming cable is connected to the RCM2100's programming header, programs can be downloaded and debugged over the serial interface.

The **DIAG** connector of the programming cable may be used on the RCM2100's programming header with the RCM2100 operating in the Run Mode. This allows the programming port to be used as a regular serial port.

4.3.1 Changing Between Program Mode and Run Mode

The RCM2100 is automatically in Program Mode when the **PROG** connector on the programming cable is attached to the RCM2100, and is automatically in Run Mode when no programming cable is attached. When the Rabbit 2000 is reset, the operating mode is determined by the status of the SMODE pins. When the programming cable's **PROG** connector is attached, the SMODE pins are pulled high, placing the Rabbit 2000 in the Program Mode. When the programming cable's **PROG** connector is not attached, the SMODE pins are pulled low, causing the Rabbit 2000 to operate in the Run Mode.

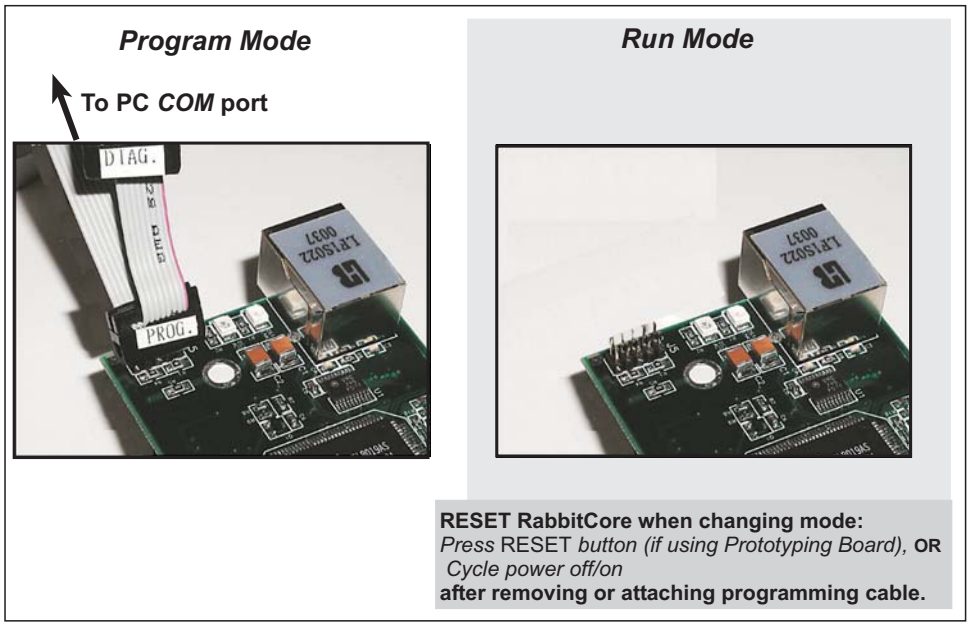


Figure 10. Switching Between Program Mode and Run Mode

A program "runs" in either mode, but can only be downloaded and debugged when the RCM2100 module is in the Program Mode.

Refer to the ***Rabbit 2000 Microprocessor User's Manual*** for more information on the programming port and the programming cable.

4.3.2 Standalone Operation of the RCM2100

The RCM2100 must be programmed via the RCM2100 Prototyping Board or via a similar arrangement on a customer-supplied board. Once the RCM2100 has been programmed successfully, remove the programming cable from the programming connector and reset the RCM2100. The RCM2100 may be reset by cycling the power off/on or by pressing the **RESET** button on the Prototyping Board. The RCM2100 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 RCM2100 module to protect against inadvertent shorts across the pins or damage to the RCM2100 if the pins are not plugged in correctly. Do not reapply power until you have verified that the RCM2100 module is plugged in correctly.

4.4 Memory

4.4.1 SRAM

The RCM2100 is designed to accept 32K to 512K of SRAM packaged in an SOIC case.

4.4.2 Flash Memory

The RCM2100 is also designed to accept 128K to 512K of flash memory packaged in a TSOP case.

> **NOTE:** Rabbit Semiconductor recommends that any customer applications should not be constrained by the sector size of the flash EPROM since it may be necessary to change the sector size in the future.

Writing to arbitrary flash memory addresses at run time is also discouraged. Instead, define a "user block" area to store persistent data. The functions **`writeUserBlock`** and **`readUserBlock`** are provided for this.

A Flash Memory Bank Select jumper configuration option based on 0 Ω surface-mounted resistors exists at header JP4. This option, used in conjunction with some configuration macros, allows Dynamic C to compile two different co-resident programs for the upper and lower halves of the 512K flash in such a way that both programs start at logical address 0000. This is useful for applications that require a resident download manager and a separate downloaded program. See Technical Note 218, ***Implementing a Serial Download Manager for a 256K Flash***, for details.

> **NOTE:** Only the Normal Mode (pins 1–2 connected at JP4), which corresponds to using the full code space, is supported at the present time.

4.4.3 Dynamic C BIOS Source Files

The Dynamic C BIOS source files handle different SRAM and flash EPROM sizes automatically.

4.5 Other Hardware

4.5.1 Clock Doubler

The RCM2100 takes advantage of the Rabbit 2000 microprocessor's internal clock doubler. A built-in clock doubler allows half-frequency crystals to be used to reduce radiated emissions. The 22.1 MHz frequency is generated using an 11.05 MHz crystal. The clock doubler is disabled automatically in the BIOS for crystals with a frequency above 12.9 MHz.

The clock doubler may be disabled if 22.1 MHz clock speeds are not required. Disabling the Rabbit 2000 microprocessor's internal clock will reduce power consumption and further reduce radiated emissions. The clock doubler is disabled with a simple configuration macro as shown below.

1. Select the "Defines" tab from the Dynamic C **Options > Project Options** menu.
2. Add the line **CLOCK_DOUBLED=0** to always disable the clock doubler.

   The clock doubler is enabled by default, and usually no entry is needed. If you need to specify that the clock doubler is always enabled, add the line **CLOCK_DOUBLED=1** to always enable the clock doubler. The clock speed will be doubled as long as the crystal frequency is less than or equal to 26.7264 MHz.
3. Click **OK** to save the macro. The clock doubler will now remain off whenever you are in the project file where you defined the macro.

4.5.2 Spectrum Spreader

RCM2100 modules that have a Rabbit 2000 microprocessor labeled ***IQ4T*** (or higher) are equipped with a Rabbit 2000 microprocessor that has a spectrum spreader, which helps to mitigate EMI problems. By default, the spectrum spreader is on automatically for RCM2100 modules that carry the ***IQ4T*** (or higher) marking when used with Dynamic C 7.30 or later versions, but the spectrum spreader may also be turned off or set to a stronger setting. The means for doing so is through a simple configuration macro as shown below.

1. Select the "Defines" tab from the Dynamic C **Options > Project Options** menu.
2. Normal spreading is the default, and usually no entry is needed. If you need to specify normal spreading, add the line

   ```
   ENABLE_SPREADER=1
   ```

   For strong spreading, add the line

   ```
   ENABLE_SPREADER=2
   ```

   To disable the spectrum spreader, add the line

   ```
   ENABLE_SPREADER=0
   ```

   NOTE: The strong spectrum-spreading setting is usually not necessary for the RCM2100.
3. Click **OK** to save the macro. The spectrum spreader will now remain off whenever you are in the project file where you defined the macro.

There is no spectrum spreader functionality for RCM2100 modules that have a Rabbit 2000 microprocessor labeled ***IQ1T***, ***IQ2T***, or ***IQ3T***, or when using any RCM2100 with a version of Dynamic C prior to 7.30.

5. SOFTWARE REFERENCE

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 single-board computers based on the Rabbit microprocessor. Chapter 4 provides the libraries and function calls related to the RCM2100.

5.1 More About Dynamic C

Dynamic C has been in use worldwide since 1989. Dynamic C is specially designed for programming embedded systems, and features quick compile and interactive debugging in the real environment. A complete reference to Dynamic C is contained in the ***Dynamic C User's Manual***.

Dynamic C for Rabbit® processors uses the standard Rabbit programming interface. This is a 10-pin connector that connects to the Rabbit Serial Port A. It is possible to reset and cold-boot a Rabbit processor via the programming port. No software needs to be present in the target system. More details are available in the ***Rabbit 2000 Microprocessor User's Manual***.

Dynamic C cold-boots the target system and compiles the BIOS. The BIOS is a basic program of a few thousand bytes in length that provides the debugging and communication facilities that Dynamic C needs. Once the BIOS has been compiled, the user can compile his own program and test it. If the BIOS fails because the program stops running, a new cold boot and BIOS compile can be done at any time.

The BIOS can be customized by using **`#define`** options.

Dynamic C does not use **`include`** files, rather it has libraries that are used for the same purpose, that is, to supply function prototypes to programs before they are compiled. See Section 4.24, "Modules," in the ***Dynamic C User's Manual*** for more information.

Dynamic C supports assembly language, either as separate functions or as fragments embedded in C programs. Interrupt routines may be written in Dynamic C or in assembly language.

5.1.1 Using Dynamic C

You have a choice of doing your software development in the flash memory or in the SRAM included on the RCM2100. There are 512K or 256K bytes of flash memory and 512K or 128K bytes of SRAM. The flash memory and SRAM options are selected with the **Options > Project Options > Compiler** menu.

The advantage of working in RAM is to save wear on the flash memory, which is limited to about 100,000 write cycles. The disadvantage is that the code and data might not both fit in RAM.

> **NOTE:** An application can be developed in RAM, but cannot run standalone from RAM after the programming cable is disconnected. All standalone applications can only run from flash memory.
>
> **NOTE:** Do not depend on the flash memory sector size or type. Due to the volatility of the flash memory market, the RCM2100 and Dynamic C were designed to accommodate flash devices with various sector sizes.

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, 98, 2000, NT, Me, and XP. 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.

5.1.2 Early Versions of Dynamic C

If you are using Dynamic C version 7.04 or earlier, modify the BIOS source code as follows. Skip these three steps if your version of Dynamic C is 7.05 or later.

1. Open the BIOS source code file named **`RABBITBIOS.C`**, which can be found in the **`BIOS`** directory.
2. Change the line

   ```
   #define USE115KBAUD 1   // set to 0 to use 57600 baud
   ```

   to read as follows.

   ```
   #define USE115KBAUD 0   // set to 0 to use 57600 baud
   ```

3. Save the changes using **File > Save**.

Now press **<Ctrl-Y>**. You should receive the **`"BIOS successfully compiled …"`** message indicating that the target is now ready to compile a program.

5.2 I/O

The RCM2100 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.

The sample programs in the Dynamic C **SAMPLES/RCM2100** directory provide further examples.

5.2.1 PCLK Output

The PCLK output is controlled by bits 7 and 6 of the Global Output Register (GOCR) on the Rabbit 2000 microprocessor, and so can be enabled or disabled in software. Starting with Dynamic C v 7.02, the PCLK output is disabled by default at compile time to minimize radiated emissions; the PCLK output is enabled in earlier versions of Dynamic C.

Use the following code to set the PCLK output as needed.

PCLK output driven with peripheral clock:

```
WrPortI(GOCR, &GOCRShadow, (GOCRShadow&~0xc0));
```

PCLK output driven with peripheral clock ÷ 2:

```
WrPortI(GOCR, &GOCRShadow, ((GOCRShadow&~0xc0)| 0x40));
```

PCLK output off (low):

```
WrPortI(GOCR, &GOCRShadow, ((GOCRShadow&~0xc0)| 0x80));
```

PCLK output on (high):

```
WrPortI(GOCR, &GOCRShadow, (GOCRShadow | 0xc0));
```

5.3 Serial Communication Drivers

Library files included with Dynamic C provide a full range of serial communications support. The **RS232.LIB** library provides a set of circular-buffer-based serial functions. The **PACKET.LIB** library provides packet-based serial functions where packets can be delimited by the 9th bit, by transmission gaps, or with user-defined special characters. Both libraries provide blocking functions, which do not return until they are finished transmitting or receiving, and nonblocking functions, which must be called repeatedly until they are finished. For more information, see the ***Dynamic C User's Manual*** and Technical Note 213, ***Rabbit 2000 Serial Port Software***.

5.4 TCP/IP Drivers

The TCP/IP drivers are located in the **TCPIP** directory. Complete information on these libraries and the TCP/IP functions is provided in the ***Dynamic C TCP/IP User's Manual***.

5.5 Upgrading Dynamic C

Dynamic C patches that focus on bug fixes are available from time to time. Check the Web site www.rabbit.com/support/ for the latest patches, workarounds, and bug fixes.

The default installation of a patch or bug fix is to install the file in a directory (folder) different from that of the original Dynamic C installation. Rabbit Semiconductor recommends using a different directory so that you can verify the operation of the patch without overwriting the existing Dynamic C installation. If you have made any changes to the BIOS or to libraries, or if you have programs in the old directory (folder), make these same changes to the BIOS or libraries in the new directory containing the patch. Do ***not*** simply copy over an entire file since you may overwrite a bug fix; of course, you may copy over any programs you have written. Once you are sure the new patch works entirely to your satisfaction, you may retire the existing installation, but keep it available to handle legacy applications.

5.5.1 Upgrades

Dynamic C installations are designed for use with the board they are included with, and are included at no charge as part of our low-cost kits. Dynamic C is a complete software development system, but does not include all the Dynamic C features. Rabbit Semiconductor also offers add-on Dynamic C modules containing the popular µC/OS-II real-time operating system, as well as PPP, Advanced Encryption Standard (AES), and other select libraries. In addition to the Web-based technical support included at no extra charge, a one-year telephone-based technical support module is also available for purchase.

6. USING THE TCP/IP FEATURES

6.1 TCP/IP Connections

Programming and development can be done with the RCM2100 RabbitCore 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 RCM2100 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.

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

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

   There are four options for connecting the RCM2100 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 RCM2100'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 RCM2100'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 RCM2100'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 RCM2100 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 through the serial port on the RCM2100 does not need to be the PC with the Ethernet card.

3. Apply Power

   Plug in the AC adapter. The RCM2100 module is now ready to be used.

6.2 TCP/IP Primer on IP Addresses

Obtaining IP addresses to interact over an existing, operating, network can involve a number of complications, and must usually be done with cooperation from your ISP and/or network systems administrator. For this reason, it is suggested that the user begin instead by using a direct connection between a PC and the RCM2100 board using an Ethernet crossover cable or a simple arrangement with a hub. (A crossover cable should not be confused with regular straight through cables.)

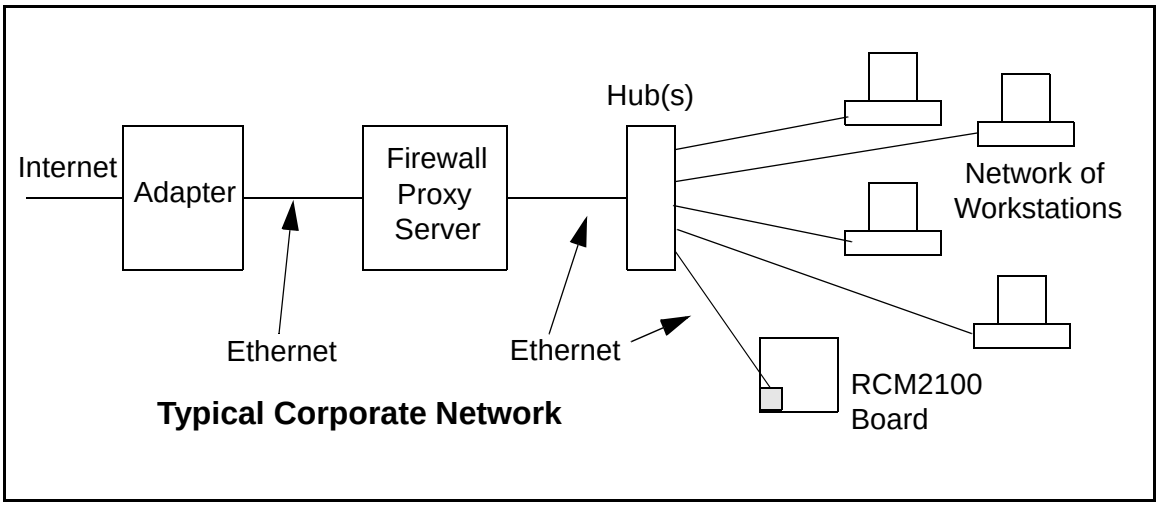
In order to set up this direct connection, the user will have to use a PC without networking, or disconnect a PC from the corporate network, or install a second Ethernet adapter and set up a separate private network attached to the second Ethernet adapter. Disconnecting your PC from the corporate network may be easy or nearly impossible, depending on how it is set up. If your PC boots from the network or is dependent on the network for some or all of its disks, then it probably should not be disconnected. If a second Ethernet adapter is used, be aware that Windows TCP/IP will send messages to one adapter or the other, depending on the IP address and the binding order in Microsoft products. Thus you should have different ranges of IP addresses on your private network from those used on the corporate network. If both networks service the same IP address, then Windows may send a packet intended for your private network to the corporate network. A similar situation will take place if you use a dial-up line to send a packet to the Internet. Windows may try to send it via the local Ethernet network if it is also valid for that network.

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 RCM2100 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 RCM2100 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 RCM2100 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.



Typical Corporate Network

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 set up a direct connection between your computer and the RCM2100 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.

6.3 IP Addresses Explained

IP (Internet Protocol) addresses are expressed as 4 decimal numbers separated by periods, for example:

216.103.126.155

10.1.1.6

Each decimal number must be between 0 and 255. The total IP address is a 32-bit number consisting of the 4 bytes expressed as shown above. A local network uses a group of adjacent IP addresses. There are always 2^N IP addresses in a local network. The netmask (also called subnet mask) determines how many IP addresses belong to the local network. The netmask is also a 32-bit address expressed in the same form as the IP address. An example netmask is:

255.255.255.0

This netmask has 8 zero bits in the least significant portion, and this means that 2^8 addresses are a part of the local network. Applied to the IP address above (216.103.126.155), this netmask would indicate that the following IP addresses belong to the local network:

216.103.126.0

216.103.126.1

216.103.126.2

etc.

216.103.126.254

216.103.126.255

The lowest and highest address are reserved for special purposes. The lowest address (216.103.126.0) is used to identify the local network. The highest address (216.103.126.255) is used as a broadcast address. Usually one other address is used for the address of the gateway out of the network. This leaves 256 - 3 = 253 available IP addresses for the example given.

6.4 How IP Addresses are Used

The actual hardware connection via an Ethernet uses Ethernet adapter addresses (also called MAC addresses). These are 48-bit addresses and are unique for every Ethernet adapter manufactured. In order to send a packet to another computer, given the IP address of the other computer, it is first determined if the packet needs to be sent directly to the other computer or to the gateway. In either case, there is an IP address on the local network to which the packet must be sent. A table is maintained to allow the protocol driver to determine the MAC address corresponding to a particular IP address. If the table is empty, the MAC address is determined by sending an Ethernet broadcast packet to all devices on the local network asking the device with the desired IP address to answer with its MAC address. In this way, the table entry can be filled in. If no device answers, then the device is nonexistent or inoperative, and the packet cannot be sent.

IP addresses are arbitrary and can be allocated as desired provided that they don't conflict with other IP addresses. However, if they are to be used with the Internet, then they must be numbers that are assigned to your connection by proper authorities, generally by delegation via your service provider.

Each RCM2100 RabbitCore module has its own unique MAC address, which consists of the prefix 0090C2 followed by the code that appears on the label affixed to the RCM2100 module. For example, a MAC address might be 0090C2C002C0.

> **TIP:** You can always verify the MAC address on your board by running the sample program **DISPLAY_MAC.C** from the **SAMPLES\TCPIP** folder.

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 RCM2100 RabbitCore modules 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 addresses 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 what this dynamically assigned IP address is, 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.

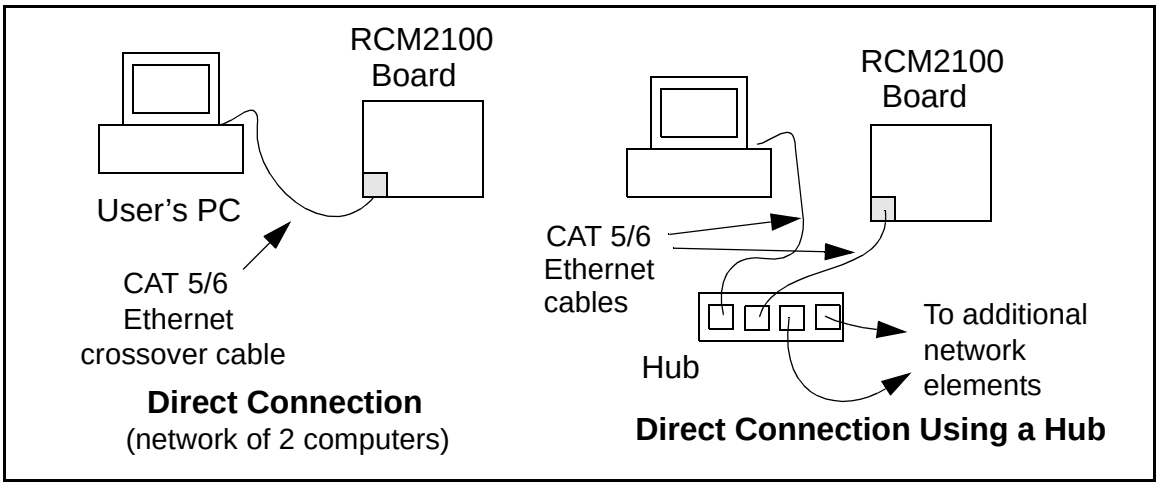
If you are not concerned about accessing the RCM2100 from the Internet, you can place the RCM2100 on the internal network using a private address assigned either statically or through DHCP.

6.6 Placing Your Device on the Network

In many corporate settings, users are isolated from the Internet by a firewall and/or a proxy server. These devices attempt to secure the company from unauthorized network traffic, and usually work by disallowing traffic that did not originate from inside the network. If you want users on the Internet to communicate with your RCM2100, you have several options. You can either place the RCM2100 directly on the Internet with a real Internet address or place it behind the firewall. If you place the RCM2100 behind the firewall, you need to configure the firewall to translate and forward packets from the Internet to the RCM2100.

6.7 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 RCM2100 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.



6.8 How to Set IP Addresses in the Sample Programs

With the introduction of Dynamic C 7.30 we have taken steps to make it easier to run many of our sample programs. Instead of the **MY_IP_ADDRESS** and other macros, 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 RCM2100 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** file.

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. 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 such as shown in the example below use macros to define the IP address assigned to the board and the IP address of the gateway, if there is a gateway.

```
#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 RCM2100:

```
#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**.

6.8.1 How to Set Up your Computer for Direct Connect

Follow these instructions to set up your PC or notebook. Check with your administrator if you are unable to change the settings as described here since you may need administrator privileges. The instructions are specifically for Windows 2000, but the interface is similar for other versions of Windows.

> **TIP:** If you are using a PC that is already on a network, you will disconnect the PC from that network to run these sample programs. Write down the existing settings before changing them to facilitate restoring them when you are finished with the sample programs and reconnect your PC to the network.

1. Go to the control panel (**Start > Settings > Control Panel**), and then double-click the Network icon.
2. Select the network interface card used for the Ethernet interface you intend to use (e.g., **TCP/IP Xircom Credit Card Network Adapter**) and click on the "Properties" button. Depending on which version of Windows your PC is running, you may have to select the "Local Area Connection" first, and then click on the "Properties" button to bring up the Ethernet interface dialog. Then "Configure" your interface card for a "10Base-T Half-Duplex" or an "Auto-Negotiation" connection on the "Advanced" tab.

   > **NOTE:** Your network interface card will likely have a different name.

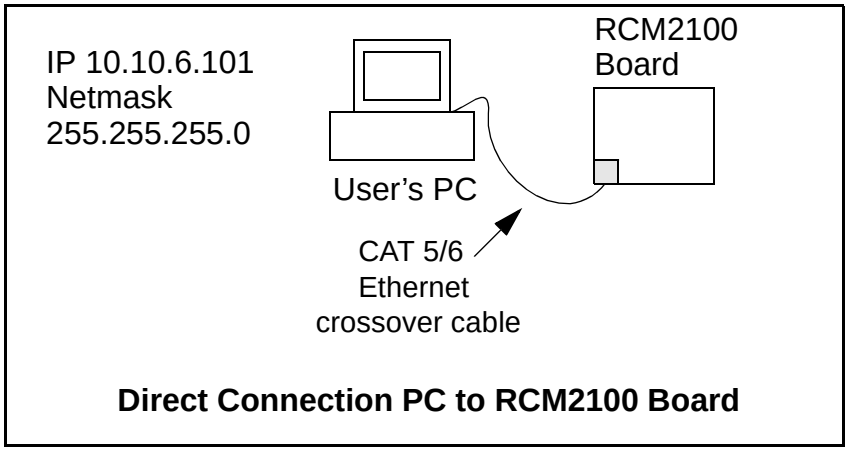
3. Now select the **IP Address** tab, and check **Specify an IP Address**, or select TCP/IP and click on "Properties" to assign an IP address to your computer (this will disable "obtain an IP address automatically"):

   IP Address : 10.10.6.101

   Netmask : 255.255.255.0

   Default gateway : 10.10.6.1

4. Click **<OK>** or **<Close>** to exit the various dialog boxes.



Direct Connection PC to RCM2100 Board

6.9 Run the PINGME.C Sample Program

Connect the crossover cable from your computer's Ethernet port to the RCM2100 board's RJ-45 Ethernet connector. Open this sample program from the **SAMPLES\TCPIP\ICMP** folder, compile the program, and start it running under Dynamic C. When the program starts running, the green **LNK** light on the RCM2100 board should be on to indicate an Ethernet connection is made. (Note: If the **LNK** light does not light, you may not have a crossover cable, or if you are using a hub perhaps the power is off on the hub.)

The next step is to ping the board from your PC. This can be done by bringing up the MS-DOS window and running the pingme program:

```
ping 10.10.6.100
```

or by **Start > Run**

and typing the entry

```
ping 10.10.6.100
```

Notice that the red **ACT** light flashes on the RCM2100 board while the ping is taking place, and indicates the transfer of data. The ping routine will ping the board four times and write a summary message on the screen describing the operation.

6.10 Running More Sample Programs With Direct Connect

The sample programs discussed here are in the Dynamic C **SAMPLES\RCM2100** folder.

6.10.1 Sample Program: PINGLED.C

One of the RCM2100's most important features is the availability of the built-in Ethernet port. This program makes the simplest possible use of the network port by "pinging" a remote system and using LEDs to report the status of the ping attempt and its return.

Compile & Run Program

Open the **PINGLED.C** sample program. Press **F9** to compile and run the program.

Each time the program sends a ping to the remote address, LED DS2 on the Prototyping Board will flash. Each time a successful return from a ping attempt is received, LED DS3 will flash.

If the ping return is unsuccessful (i.e., the remote system does not exist or does not acknowledge the ping within the timeout period), DS3 will not flash.

With short ping times, as will be encountered in most micro-LAN and LAN settings, the two LEDs should flash almost in parallel as pings are sent and returned.

You can modify the **#define PING_DELAY** statement to change the amount of time between the outgoing pings.

Program Description

For operation, network addresses must be correctly defined at the start of this program. The **TCPCONFIG 1** macro in the sample program provides default settings for **MY_IP_ADDRESS**, which is the address of the RCM2100 module, **MY_NETMASK**, and **MY_GATEWAY** (which needs to be defined if you wish to ping systems outside the local network). If you wish to ping systems using domain names instead of IP addresses, a valid DNS server address must be defined for **MY_NAMESERVER**. These TCP/IP settings can be changed as needed in the **TCP_CONFIG.LIB** library.

The IP address to be pinged is defined by **PING_WHO**. You will have to change this address and recompile the program to ping different addresses. (In most real-world applications, there should be some mechanism by which to dynamically define or select addresses.) This address may be defined as a numeric IP address. If a gateway to the Internet and a valid DNS server are specified, this definition may also be a fully-qualified domain name (such as "www.rabbit.com").

The program first defines three functions to control the LEDs—one to initialize them, and then one each to drive the "ping out" and "ping in" LEDs.

The program begins by calling the LED initialization function **pingleds_setup( )**. More importantly, it then calls **sock_init( )**, which initializes the packet driver and the TCP manager using the compiler defaults. This function must always be called before any other TCP/IP functions.

The program then resolves the address to be pinged into a numeric value. using the library function **resolve()**. If the defined address is numeric, it converts the define string into truly numeric form. If the address is a domain name, the function queries the indicated DNS server to obtain the numeric address. (If the function is unable to resolve the address—if, for example, the numeric address is incomplete or badly formed, or the DNS server is unable to identify the domain name—the program will print a message to the screen and terminate.)

The program then begins an endless loop using **for(;;)**. Within this loop, the program executes the following steps:

1. Calls **tcp_tick()** to perform the basic housekeeping functions for the socket;
2. As a costatement, waits for the duration of **PING_DELAY** (defined by default as 500 ms or one-half second), issues a ping to the resolved address using the **_ping()** function, and flashes LED DS2;
3. As a second costatement, checks for a ping return using the **_chk_ping()** function. If the ping is successful, the costatement flashes LED DS3.

If you uncomment the **#VERBOSE** define near the beginning of the program, the ping return costatement will also print a message to the screen indicating each successful ping.

6.10.2 Sample Program: ETHCORE1.C

The RCM2100 modules with Ethernet ports can act as micro Web page servers, with dynamic interaction between the controller and the web pages. This sample program demonstrates how a Web page can be used to both monitor and control an RCM2100 module.

Compile & Run Program

Open the sample program **ETHCORE1.C**. Press **F9** to compile and run the program.

> **TIP:** This program will be more interesting to observe if LEDs DS4 and DS5 are installed on the Prototyping Board.

When the program starts, LEDs DS2, DS3 and DS5 will be lit, and DS4 will be dark. Open a web browser and enter the IP address you defined for the RCM2100 module in the program in the address window. A page like that shown in Figure 11 should appear.

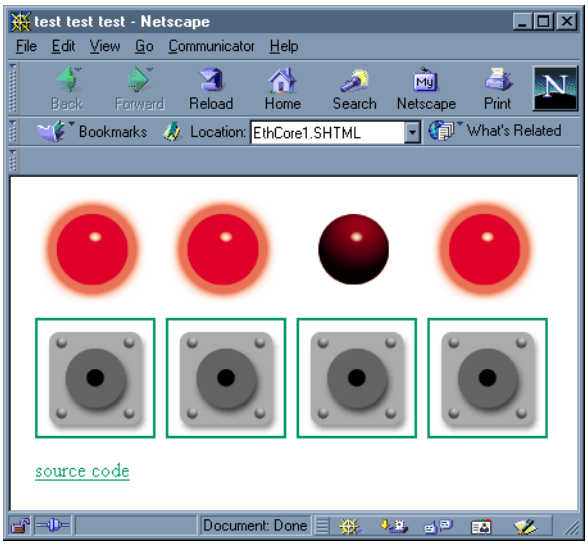


Figure 11. Browser screen for Sample Program ETHCORE1.C

Clicking on each of the button images in the browser window will toggle the state of the associated LED image, and will toggle the state of the corresponding LED on the Prototyping Board. Since the web page is generated by the RabbitCore module (using Dynamic HTML), the LED image and the corresponding LED's real state will always be in step.

Program Description

This program begins to show the range of applications for an Ethernet-enabled embedded system controller, so let's look closely at its operation.

As with **PINGLED.C**, several network addresses must be defined before this application can work. The **TCPCONFIG 1** macro in the sample program provides default settings for **MY_IP_ADDRESS**, which is the address of the RCM2100 module, **MY_NETMASK**, and **MY_GATEWAY** (which needs to be defined if you wish to reach the system from outside the local network). These TCP/IP settings can be changed as needed in the **TCP_CONFIG.LIB** library.

Generally, the other defined values may be left at their default settings. If you are operating the system behind a firewall or proxy and need to specify a host port for redirection, you should comment out the line reading:

```
#define REDIRECTHOST MY_IP_ADDRESS
```

Then uncomment the next line, which defines a specific redirection host and port:

```
#define REDIRECTHOST "my host.com:8080"
```

Be sure to enter the host port where indicated by **"my host.com:8080".**

This application creates dynamic HTML web pages on the fly. For simplicity, all of the Web page components—shell HTML, image GIFs, etc.—are imported into flash memory using the **#ximport** statements. It is also possible to read these files from other locations, including the onboard flash file system, but this application keeps things simple by loading all the components into working memory.

The program then defines four instances of an LED toggling function, which are basic CGI functions that swap the values "ledon.gif" and "ledoff.gif" as the contents of the **ledn** strings, and then force a reload of the web page to change the associated LED image. The physical LEDs on the Prototyping Board are turned on or off to match the **ledn** strings displayed on the Web page.

6.10.3 Additional Sample Programs

- **ETHCORE2.C**—This program takes anything that comes in on a port and sends it out Serial Port C. It uses SW2 as a signal that the connection should be closed, and PA0 as an indication that there is an open connection. You may change SW2 and PA0 to suit your application needs.

  Follow the instructions included with the sample program. Run the Telnet program on your PC (**Start > Run** `telnet 10.10.6.100`). As long as you have not modified the **TCPCONFIG 1** macro in the sample program, the IP address is 10.10.6.100 as shown; otherwise use the TCP/IP settings you entered in the **TCP_CONFIG.LIB** library. Each character you type will be printed in Dynamic C's **STDIO** window, indicating that the board is receiving the characters typed via TCP/IP.

- **LEDCONSOLE.C**—Demonstrates the features of **ZCONSOLE.LIB** command-oriented console library to control two LEDs on the Prototyping Board.

6.10.4 More Information

Refer to the ***Dynamic C TCP/IP User's Manual*** for complete details on the Dynamic C implementation of TCP/IP protocols.

6.11 Where Do I Go From Here?

NOTE: If you purchased your RCM2100 through a distributor or through a Rabbit Semiconductor 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 Semiconductor Technical Bulletin Board at www.rabbit.com/support/bb/.
- Use the Technical Support e-mail form at www.rabbit.com/support/.

If the sample programs ran fine, you are now ready to go on.

Additional sample programs are described in the ***Dynamic C TCP/IP User's Manual***.

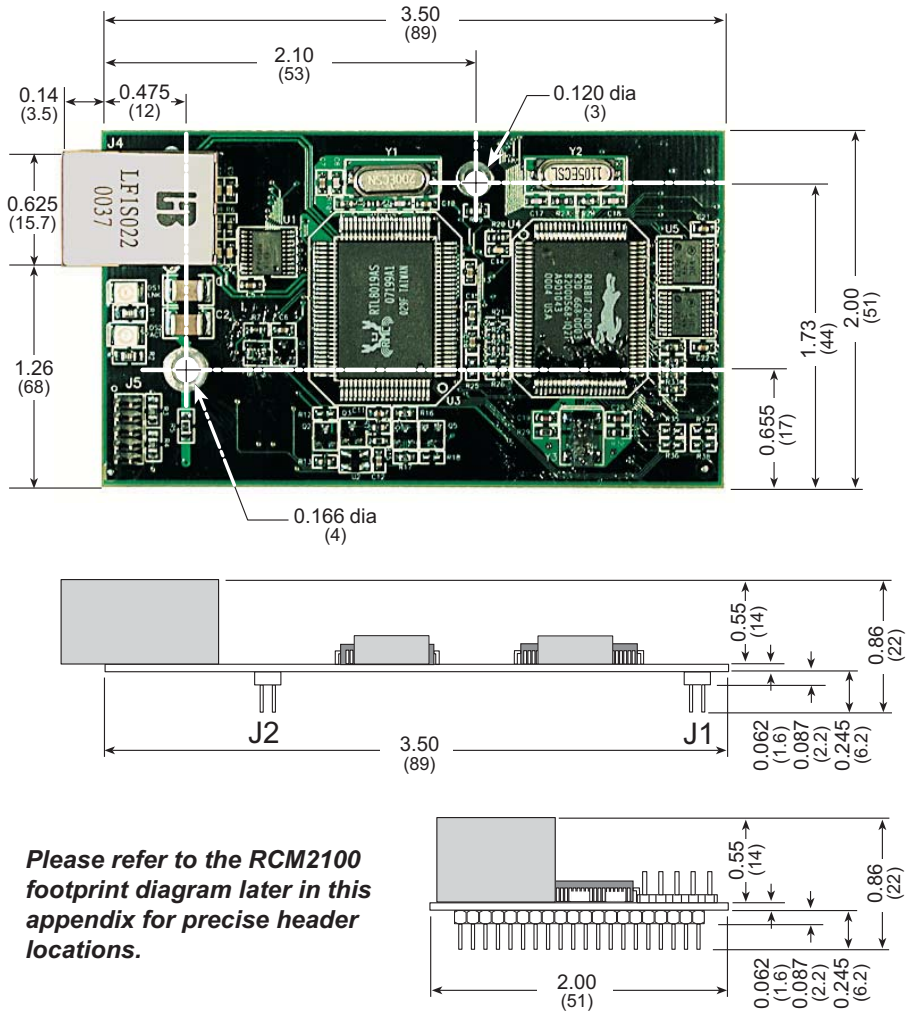
Please refer to the ***Dynamic C TCP/IP User's Manual*** to develop your own applications. ***An Introduction to TCP/IP*** provides background information on TCP/IP, and is available on the CD and on our Web site.

APPENDIX A. RABBITCORE RCM2100 SPECIFICATIONS

Appendix A provides the specifications for the RCM2100, and describes the conformal coating.

A.1 Electrical and Mechanical Characteristics

Figure A-1 shows the mechanical dimensions for the RCM2100.



Please refer to the RCM2100 footprint diagram later in this appendix for precise header locations.

Figure A-1. RCM2100 Dimensions

NOTE: All measurements are in inches followed by millimeters enclosed in parentheses. All dimensions have a manufacturing tolerance of ±0.01" (0.25 mm).

It is recommended that you allow for an “exclusion zone” of 0.04" (1 mm) around the RCM2100 in all directions when the RCM2100 is incorporated into an assembly that includes other components. An “exclusion zone” of 0.16" (4 mm) is recommended below the RCM2100 when the RCM2100 is plugged into another assembly using the shortest connectors for headers J1 and J2 on the RCM2100. Figure A-2 shows this “exclusion zone.”

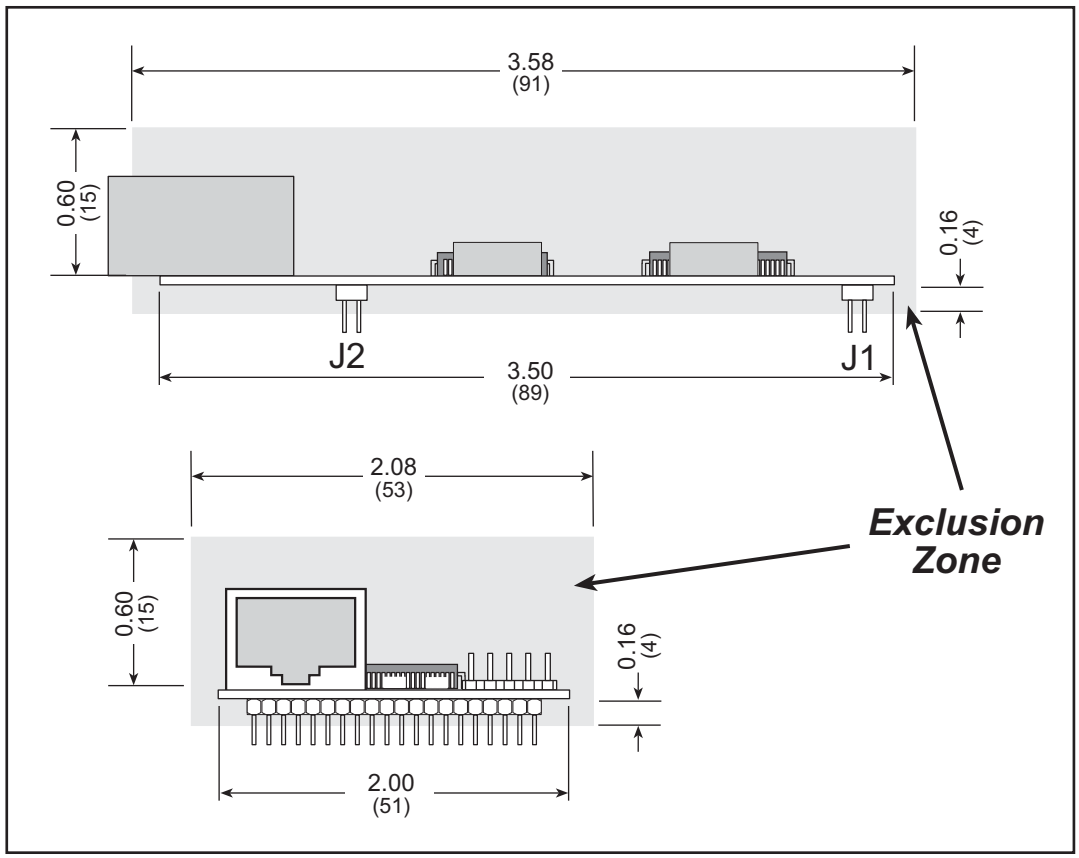


Figure A-2. RCM2100 “Exclusion Zone”

Table A-1 lists the electrical, mechanical, and environmental specifications for the RCM2100.

Table A-1. RCM2100 Specifications

Parameter	RCM2100	RCM2110	RCM2120	RCM2130
Microprocessor	Rabbit® 2000 at 22.1 MHz			
Ethernet Port	10/100-compatible with 10Base-T interface, RJ-45, 2 LEDs		None	
Flash Memory	Two 256K × 8	One 256K × 8	Two 256K × 8	One 256K × 8
SRAM	512K × 8	128K × 8	512K × 8	128K × 8
Backup Battery	Connection for user-supplied backup battery (to support RTC and SRAM)			
General-Purpose I/O	34 parallel I/0 lines grouped in five 8-bit ports (and shared with serial ports): • 20 configurable I/O • 8 fixed inputs • 6 fixed outputs		40 parallel I/0 lines grouped in five 8-bit ports (and shared with serial ports): • 26 configurable I/O • 8 fixed inputs • 6 fixed outputs	
Additional Inputs	2 startup mode (for master/slave), reset			
Additional Outputs	Status, clock, watchdog, reset			
Memory, I/O Interface	13 address lines, 8 data lines, I/O read/write, buffer enable			
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 RCM2100 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, 140 mA			
Operating Temperature	–40°C to +70°C		–40°C to +85°C	
Humidity	5% to 95%, noncondensing			
Connectors	Two IDC headers 2 × 20, 2 mm pitch			
Board Size	2.00" × 3.50" × 0.86" (51 mm × 89 mm × 22 mm)		2.00" × 3.50" × 0.5" (51 mm × 89 mm × 13 mm)	

A.1.1 Headers

The RCM2100 uses headers at J1, J2, and J3 for physical connection to other boards. J1 and J2 are 2 × 20 SMT headers with a 2 mm pin spacing. J3 is a 2 × 4 header with a 2 mm pin spacing.

Figure A-3 shows the layout of another board for the RCM2100 to be plugged in to. These reference design values are relative to the mounting hole or to the header connectors.

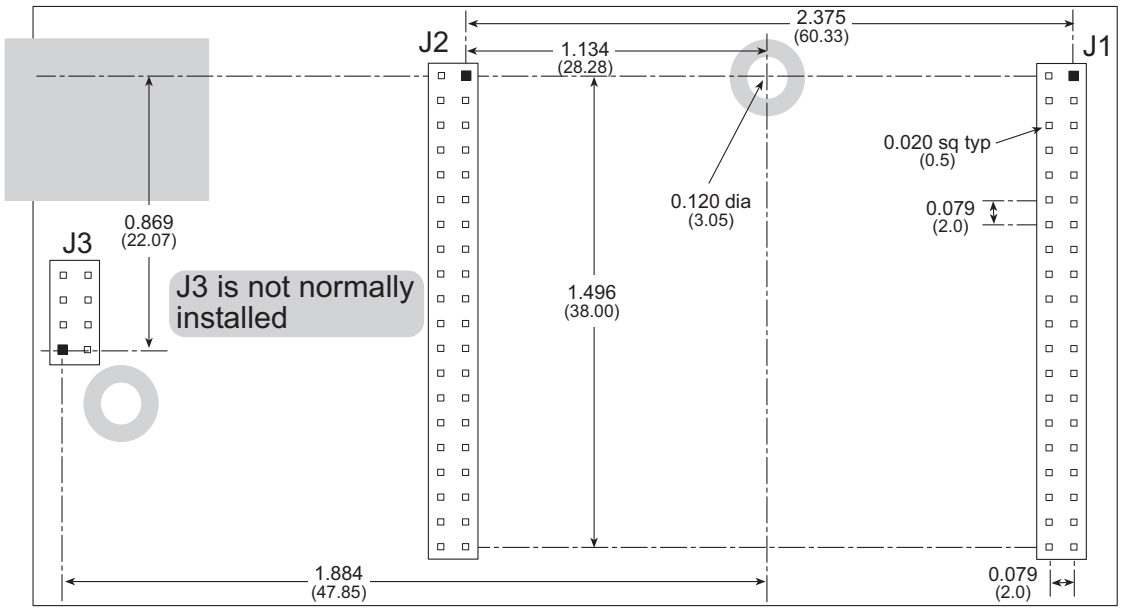


Figure A-3. User Board Footprint for the RCM2100

A.1.2 Physical Mounting

A 9/32” (7 mm) standoff with a 4-40 screw is recommended to attach the RCM2100 to a user board at the hole position shown in Figure A-3. A standoff with a screw may also be used at the hole position close to the RJ-45 Ethernet connector for a second anchor, or you may opt to have a nut and bolt with a wire at this hole position if you removed resistor R5 and elected to ground the RJ-45 Ethernet connector to the chassis.

A.2 Bus Loading

You must pay careful attention to bus loading when designing an interface to the Rabbit-Core RCM2100. This section provides bus loading information for external devices.

Table A-2 lists the capacitance for the various RCM2100 I/O ports.

Table A-2. Capacitance of RCM2100 I/O Ports

I/O Ports	Input Capacitance		Output Capacitance	
	Typ.	Max.	Typ.	Max.
Parallel Ports A to E	6 pF	12 pF	10 pF	14 pF
Data Lines BD0–BD7	12 pF	18 pF	18 pF	22 pF
Address Lines BA0–BA12	—	—	8 pF	12 pF

Table A-3 lists the external capacitive bus loading for the various RCM2100 output ports. Be sure to add the loads for the devices you are using in your custom system and verify that they do not exceed the values in Table A-3.

Table A-3. External Capacitive Bus Loading -40°C to +70°C

Output Port	Clock Speed (MHz)	Maximum External Capacitive Loading (pF)
A[12:0] D[7:0]	22.1	50
PD[3:0]	22.1	100
PA[7:0] PB[7,6] PC[6,4,2,0] PD[7:4]* PE[7:0]	22.1	90
All data, address, and I/O lines with clock doubler disabled	11.0592	100

* The Parallel Port D outputs (PD[7:4]) are available only on the RCM2120 and the RCM2130 models.

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

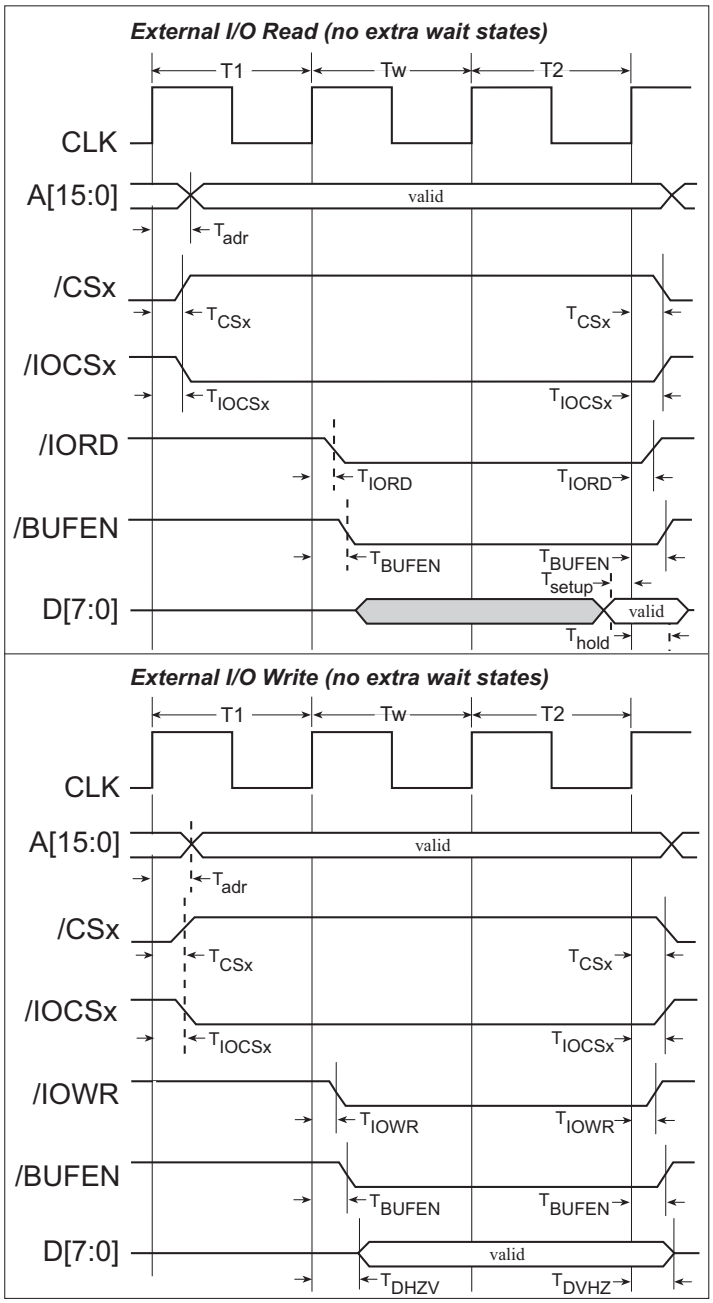


Figure A-4. External I/O Read and Write Cycles—No Extra Wait States

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.3 Rabbit 2000 DC Characteristics

Table A-4 outlines the DC characteristics for the Rabbit 2000 at 5.0 V over the recommended operating temperature range from T_a = –40°C to +85°C, V_{DD} = 4.5 V to 5.5 V.

Table A-4. 5.0 Volt DC Characteristics

Symbol	Parameter	Test Conditions	Min	Typ	Max	Units
I_{IH}	Input Leakage High	$V_{IN} = V_{DD}$, V_{DD} = 5.5 V			10	µA
I_{IL}	Input Leakage Low (no pull-up)	$V_{IN} = V_{SS}$, V_{DD} = 5.5 V	-10			µA
I_{OZ}	Output Leakage (no pull-up)	$V_{IN} = V_{DD}$ or V_{SS}, V_{DD} = 5.5 V	-10		10	µA
V_{IL}	CMOS Input Low Voltage				0.3 x V_{DD}	V
V_{IH}	CMOS Input High Voltage		0.7 x V_{DD}			V
V_T	CMOS Switching Threshold	V_{DD} = 5.0 V, 25°C		2.4		V
V_{OL}	CMOS Output Low Voltage	I_{OL} = See Table A-5 (sinking) V_{DD} = 4.5 V		0.2	0.4	V
V_{OH}	CMOS Output High Voltage	I_{OH} = See Table A-5 (sourcing) V_{DD} = 4.5 V	0.7 x V_{DD}	4.2		V

A.4 I/O Buffer Sourcing and Sinking Limit

Unless otherwise specified, the Rabbit I/O buffers are capable of sourcing and sinking 8 mA of current per pin at full AC switching speed. Full AC switching assumes a 22.1 MHz CPU clock and capacitive loading on address and data lines of less than 100 pF per pin. Pins A0–A12 and D0–D7 are each rated at 8 mA. The absolute maximum operating voltage on all I/O is V_{DD} + 0.5 V, or 5.5 V.

Table A-5 shows the AC and DC output drive limits of the parallel I/O buffers when the Rabbit 2000 is used in the RCM2100.

Table A-5. I/O Buffer Sourcing and Sinking Capability

Pin Name	Output Drive Sourcing*/Sinking† Limits (mA)	
Output Port Name	Full AC Switching SRC/SNK	Maximum‡ DC Output Drive SRC/SNK
PA [7:0]	8/8	12/12
PB [7, 1, 0]	8/8	12/12
PC [6, 4, 2, 0]	8/8	12/12
PD [7:4]	8/8	12/12
PD [3:0]**	16/16	25/25
PE [7:0]	8/8	12/12

* The maximum DC sourcing current for I/O buffers between V_{DD} pins is 112 mA.

† The maximum DC sinking current for I/O buffers between V_{SS} pins is 150 mA.

‡ The maximum DC output drive on I/O buffers must be adjusted to take into consideration the current demands made my AC switching outputs, capacitive loading on switching outputs, and switching voltage.

The current drawn by all switching and nonswitching I/O must not exceed the limits specified in the first two footnotes.

** The combined sourcing from Port D [7:0] may need to be adjusted so as not to exceed the 112 mA sourcing limit requirement specified in the first footnote.

A.5 Jumper Configurations

Figure A-5 shows the header locations used to configure the various RCM2100 options via jumpers.

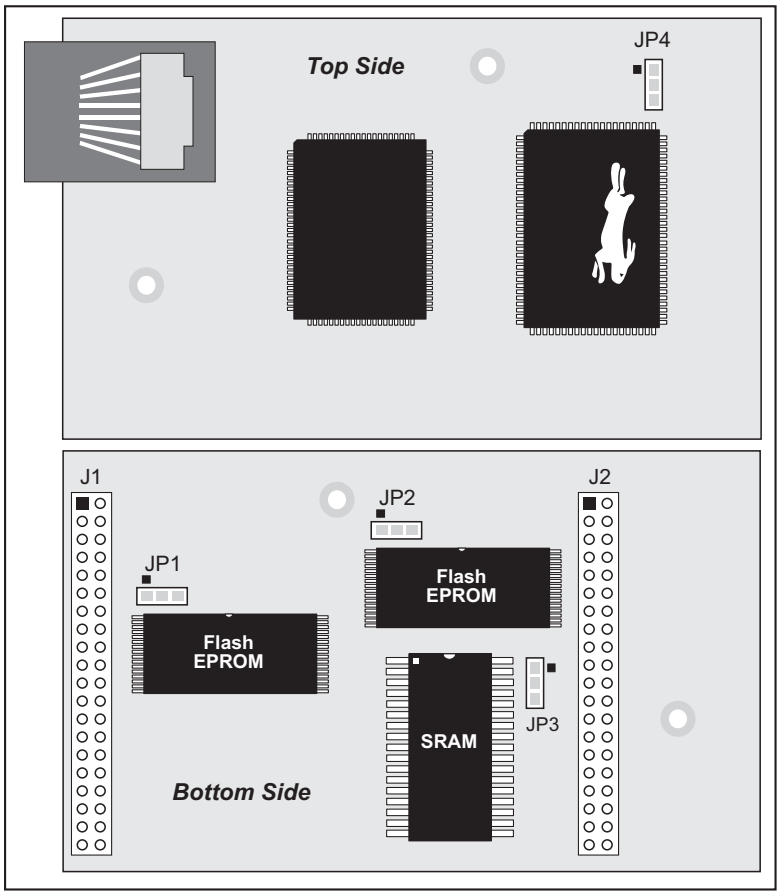


Figure A-5. Location of RCM2100 Configurable Positions

Table A-6 lists the configuration options.

Table A-6. RCM2100 Jumper Configurations

Header	Description	Pins Connected		Factory Default
JP1	Flash Memory Size (U10)	1–2	128K/256K	✕
		2–3	512K	
JP2	Flash Memory Size (U11) (not installed on RCM2110/RCM2130)	1–2	128K/256K	RCM2100 RCM2120
		2–3	512K	—
JP3	SRAM Size	n.c.	32K	—
		1–2	128K	RCM2110 RCM2130
		2–3	512K	RCM2100 RCM2120
JP4	Flash Memory Bank Select (U10)	1–2	Normal Mode	✕
		2–3	Bank Mode	

NOTE: The jumper connections are made using 0 Ω surface-mounted resistors.

A.6 Conformal Coating

The areas around the 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.

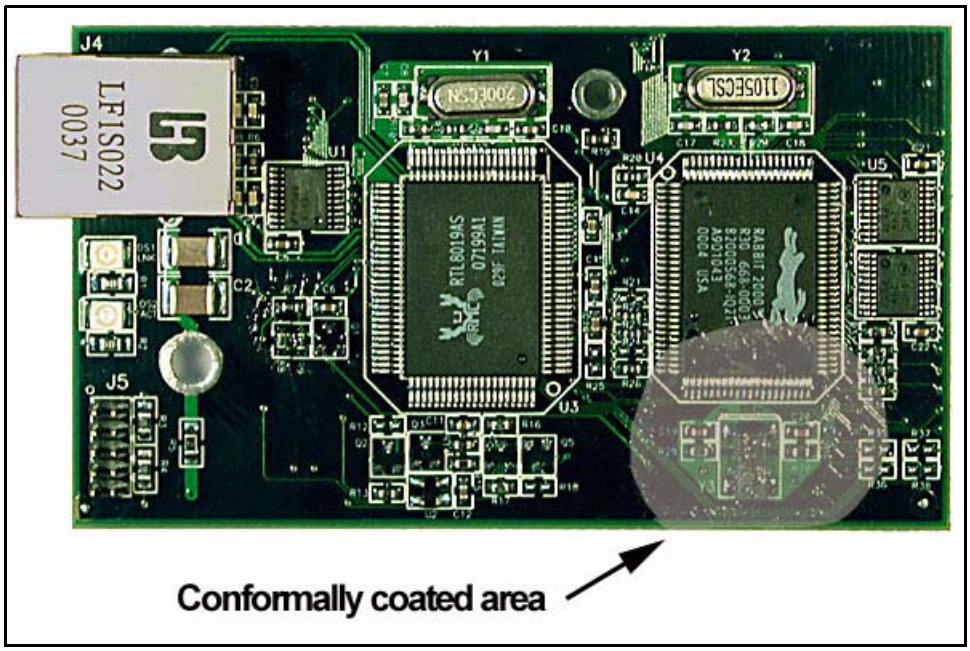


Figure A-6. RCM2100 Areas Receiving Conformal Coating

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 Rabbit Semiconductor Technical Note 303, ***Conformal Coatings***.

APPENDIX B. PROTOTYPING BOARD

Appendix B describes the features and accessories of the Prototyping Board, and explains the use of the Prototyping Board to demonstrate the RCM2100 and to build prototypes of your own circuits.

B.1 Overview of the Prototyping Board

The Prototyping Board included in the Development Kit makes it easy to connect an RCM2100 module to a power supply and a PC workstation for development. It also provides an array of basic I/O peripherals (switches and LEDs), as well as a prototyping area for more advanced hardware development.

For the most basic level of evaluation and development, the Prototyping Board can be used without modification.

As you progress to more sophisticated experimentation and hardware development, modifications and additions can be made to the board without modifying or damaging the RCM2100 module itself.

The Prototyping Board is shown in Figure B-1 below, with its main features identified

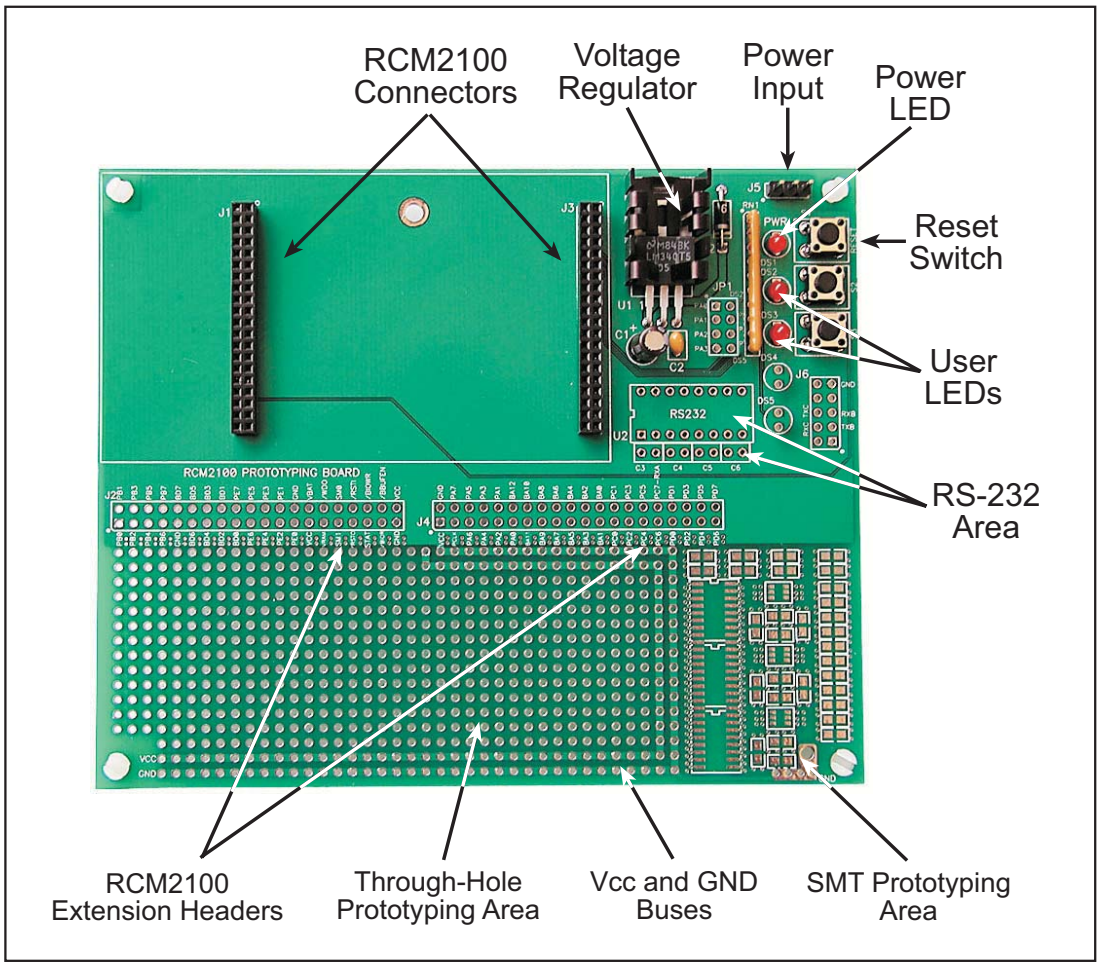


Figure B-1. RCM2100 Prototyping Board

B.1.1 Prototyping Board Features

Power Connection. A 3-pin header is provided for connection of a power supply. Note that it is symmetrical, with both outer pins connected to ground and the center pin connected to the raw V+ input. The cable of the wall transformer provided with the North American version of the Development Kit ends in a connector that is correctly connected in either orientation.

Users providing their own power supply should ensure that it delivers 9–24 V DC at not less than 500 mA. The voltage regulator will get warm in use, but lower supply voltages will reduce thermal dissipation from the device.

Regulated Power Supply. The raw DC voltage provided at the POWER IN jack is routed to a 5 V linear voltage regulator, which provides stable power to the RCM2100 module and the Prototyping Board. A Shottky diode protects the power supply against damage from reversed raw power connections.

Power LED. The power LED lights whenever power is connected to the Prototyping Board.

Reset Switch. A momentary-contact, normally open switch is connected directly to the RCM2100's **/RES_IN** pin. Pressing the switch forces a hardware reset of the system.

I/O Switches & LEDs. Two momentary-contact, normally open switches are connected to the PB2 and PB3 pins of the RCM2100 module, and may be read as inputs by sample applications.

Two LEDs are connected to the PA0 and PA1 pins of the module, and may be driven as output indicators by sample applications. (Two more LEDs, driven by PA2 and PA3, may be added to the Prototyping Board for additional outputs.)

All the LEDs are connected through JP1, which has traces shorting adjacent pads together. These traces may be cut to disconnect the LEDs, and an 8-pin header soldered into JP1 to permit their selective reconnection with jumpers. See Figure B-4 for details.

Expansion Areas. The Prototyping Board is provided with several unpopulated areas for expansion of I/O and interfacing capabilities. See the next section for details.

Prototyping Area. A generous prototyping area has been provided for the installation of through-hole components. Vcc (5 V DC) and Ground buses run around the edge of this area. An area for surface-mount devices is provided to the right of the through-hole area. (Note that there are SMT device pads on both top and bottom of the Prototyping Board.)

B.1.2 Prototyping Board Expansion

The Prototyping Board comes with several unpopulated areas, which may be filled with components to suit the user's development needs. After you have experimented with the sample programs in Chapter 4, you may wish to expand the board's capabilities for further experimentation and development. Refer to the Prototyping Board schematic (090–0116) for details as necessary.

Module Extension Headers The complete pin set of the RCM2100 module is duplicated at these two headers. Developers can solder wires directly into the appropriate holes, or, for more flexible development, two 40-pin header strips can be soldered into place. See Figure B-5 for the header pinouts.

RS-232 Port Two 2-wire or one 5-wire RS-232 serial port can be added to the Prototyping Board by installing a driver IC and four capacitors where indicated. The Maxim MAX232 driver chip or a similar device is recommended for U2. Refer to the Prototyping Board schematic for additional details.

A 10-pin 0.1" spacing header strip can be installed at J6 to permit connection of a ribbon cable leading to a standard DE-9 serial connector.

> **NOTE:** The RS-232 chip, capacitors and header strip are available from electronics distributors such as Digi-Key and Mouser Electronics.

Additional LEDs Two additional LEDs (supplied with the development kit) can be soldered into place at DS4 and DS5. The cathode lead (longer of the two, marked by a flat on the LED case) should go towards the module.

Prototyping Board Component Header Several I/O pins from the module are hardwired to the Prototyping Board LEDs and switches.

To disconnect these devices and permit the pins to be used for other purposes, cut the traces between the pin rows. Use an exacto knife or similar tool to cut or break the traces crossing JP1, in the area indicated in Figure B-4.

To permit selective reconnection of the devices, jumpers may be placed across the 8-pin header strip at JP1.

B.2 Mechanical Dimensions and Layout

Figure B-2 shows the mechanical dimensions and layout for the RCM2100 Prototyping Board.

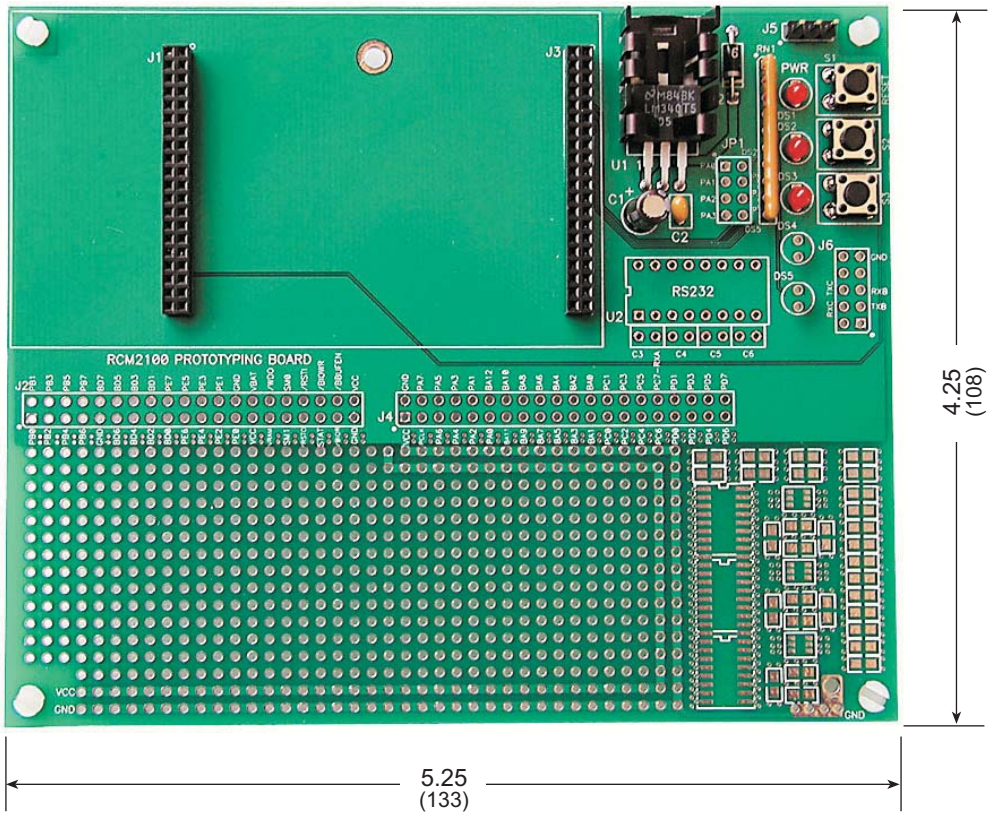


Figure B-2. RCM2100 Prototyping Board Dimensions

Table B-1 lists the electrical, mechanical, and environmental specifications for the Prototyping Board.

Table B-1. Prototyping Board Specifications

Parameter	Specification
Board Size	4.25" × 5.25" × 1.00" (108 mm × 133 mm × 25 mm)
Operating Temperature	–40°C to +70°C
Humidity	5% to 95%, noncondensing
Input Voltage	7.5 V to 25 V DC
Maximum Current Draw (including user-added circuits)	1 A at 12 V and 25°C, 0.7 A at 12 V and 70ºC
Prototyping Area	1.7" × 4" (43 mm × 102 mm) throughhole, 0.1" spacing
Standoffs/Spacers	4, accept 6-32 × 3/8 screws

B.3 Power Supply

The RCM2100 requires a regulated 5 V ± 0.25 V DC power source to operate. Depending on the amount of current required by the application, different regulators can be used to supply this voltage.

The Prototyping Board has an onboard LM340-T5 or equivalent. The LM340-T5 is an inexpensive linear regulator that is easy to use. Its major drawback is its inefficiency, which is directly proportional to the voltage drop across it. The voltage drop creates heat and wastes power.

A switching power supply may be used in applications where better efficiency is desirable. The LM2575 is an example of an easy-to-use switcher. This part greatly reduces the heat dissipation of the regulator. The drawback in using a switcher is the increased cost.

The Prototyping Board itself is protected against reverse polarity by a Shottky diode at D2 as shown in Figure B-3.

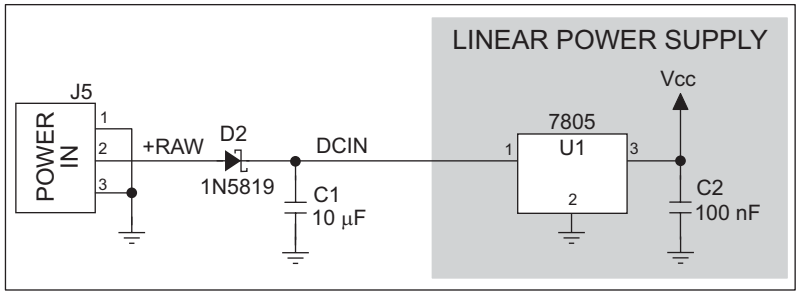


Figure B-3. Prototyping Board Power Supply

Capacitor C1 provides surge current protection for the voltage regulator, and allows the external power supply to be located some distance away.

B.4 Using the Prototyping Board

The Prototyping Board is actually both a demonstration board and a prototyping board. As a demonstration board, it can be used to demonstrate the functionality of the RCM2100 right out of the box without any modifications to either board. There are no jumpers or dip switches to configure or misconfigure on the Prototyping Board so that the initial setup is very straightforward.

The Prototyping Board comes with the basic components necessary to demonstrate the operation of the RCM2100. Two LEDs (DS2 and DS3) are connected to PA0 and PA1, and two switches (S2 and S3) are connected to PB2 and PB3 to demonstrate the interface to the Rabbit 2000 microprocessor. Reset switch S1 is the hardware reset for the RCM2100.

Two more LEDs, driven by PA2 and PA3, may be added to the Prototyping Board for additional outputs.

To maximize the availability of RCM2100 resources, the demonstration hardware (LEDs and switches) on the Prototyping Board may be disconnected. This is done by cutting the traces below the silk-screen outline of header JP1 on the bottom side of the Prototyping Board. Figure B-4 shows the four places where cuts should be made. An exacto knife would work nicely to cut the traces. Alternatively, a small standard screwdriver may be carefully and forcefully used to wipe through the PCB traces.

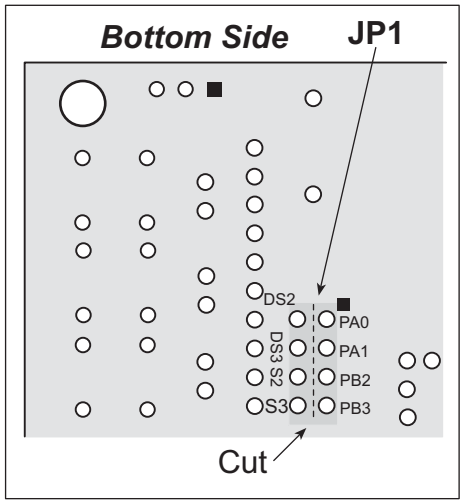


Figure B-4. Where to Cut Traces to Permanently Disable Demonstration Hardware on Prototyping Board

The power LED (PWR) and the RESET switch remain connected. Jumpers across the appropriate pins on header JP1 can be used to reconnect specific demonstration hardware later if needed.

Table B-2. Prototyping Board Jumper Settings

Header JP2	
Pins	**Description**
1–2	PA0 to LED DS2
3–4	PA1 to LED DS3
5–6	PB2 to Switch S2
7–8	PB3 to Switch S3

Note that the pinout at location JP1 on the bottom side of the Prototyping Board (shown in Figure B-4) is a mirror image of the top side pinout.

The Prototyping Board provides the user with RCM2100 connection points brought out conveniently to labeled points at headers J2 and J4 on the Prototyping Board. Small to medium circuits can be prototyped using point-to-point wiring with 20 to 30 AWG wire between the prototyping area and the holes at locations J2 and J4. The holes are spaced at 0.1" (2.5 mm),

and 40-pin headers or sockets may be installed at J2 and J4. The pinouts for locations J1 and J3, which correspond to J2 and J4, are shown in Figure B-5.

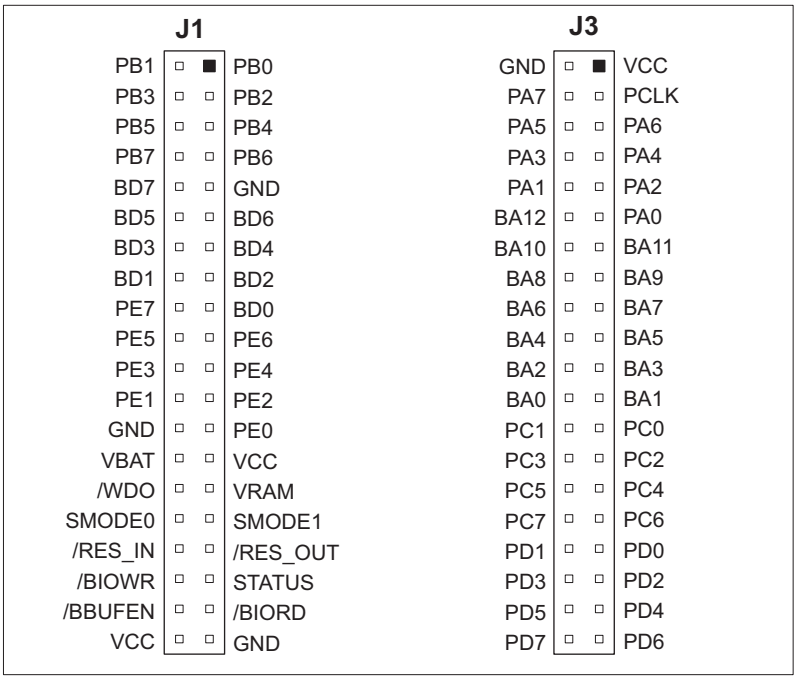


Figure B-5. RCM2100 Prototyping Board Pinout (Top View)

A pair of small holes capable of holding 30 AWG wire appears below each hole pair at locations J2 and J4 for convenience in point-to-point wiring when headers are installed. The signals are those of the adjacent pairs of holes at J2 and J4. These small holes are also provided for the components that may be installed to the right of the prototyping area.

There is an additional 2" × 3" of through-hole prototyping space available on the Prototyping Board. VCC and GND traces run along the edge of the Prototyping Board for easy access. A GND pad is also provided at the lower right for alligator clips or probes.

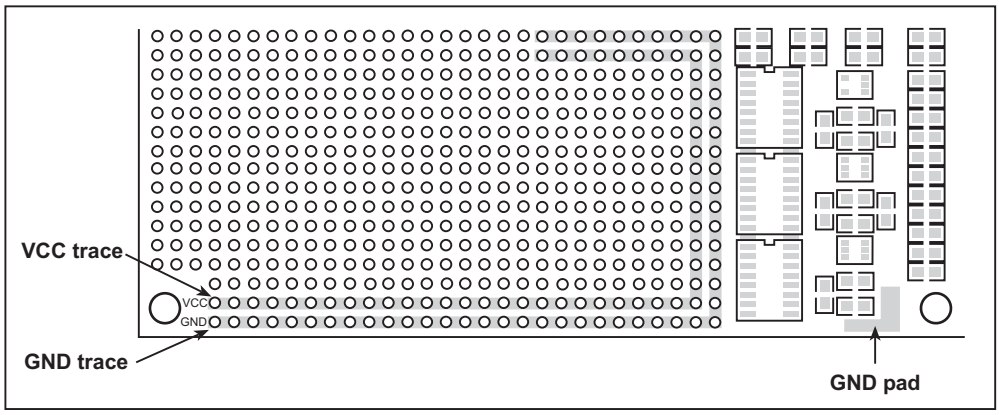


Figure B-6. VCC and GND Traces Along Edge of Prototyping Board

B.4.1 Adding Other Components

There is room on the Prototyping Board for a user-supplied RS-232 transceiver chip at location U2 and a 10-pin header for serial interfacing to external devices at location J6. A Maxim MAX232 transceiver is recommended. When adding the MAX232 transceiver at position U2, you must also add 100 nF charge storage capacitors at positions C3–C6 as shown in Figure B-7.

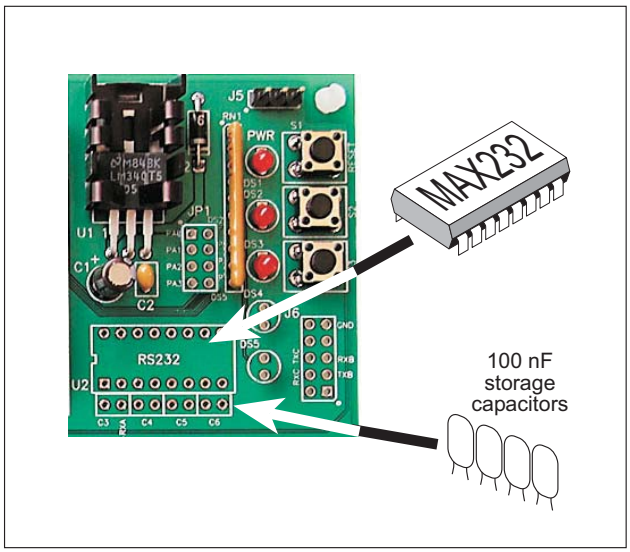


Figure B-7. Location for User-Supplied RS-232 Transceiver and Charge Storage Capacitors

There are two sets of pads that can be used for surface mount prototyping SOIC devices. The silk screen layout separates the rows into six 16-pin devices (three on each side). However, there are pads between the silk screen layouts giving the user two 52-pin (2×26) SOIC layouts with 50 mil pin spacing. There are six sets of pads that can be used for 3- to 6-pin SOT23 packages. There are also 60 sets of pads that can be used for SMT resistors and capacitors in an 0805 SMT package. Each component has every one of its pin pads connected to a hole in which a 30 AWG wire can be soldered (standard wire wrap wire can be soldered in for point-to-point wiring on the Prototyping Board). Because the traces are very thin, carefully determine which set of holes is connected to which surface-mount pad.

APPENDIX C. POWER SUPPLY

Appendix C provides information on the current and power supply requirements of the RCM2100, and some background on the chip select and battery-backup circuits used in power mangement.

C.1 Power Supplies

The RCM2100 requires a regulated 5 V ± 0.25 V DC power source. The RCM2100 design presumes that the voltage regulator is on the user board, and that the power is made available to the RCM2100 board through headers J1 and J2.

An RCM2100 with no loading at the outputs operating at 22.1 MHz typically draws 140 mA. The RCM2100 will consume 13 mA to 15 mA of additional current when the programming cable is used to connect J5 to a PC.

C.1.1 Batteries and External Battery Connections

The RCM2100 does not have a battery, but there is provision for a customer-supplied battery to back up SRAM and keep the internal Rabbit 2000 real-time clock running.

Header J2, shown in Figure C-1, allows access to the external battery. This header makes it possible to connect an external 3 V power supply. This allows the SRAM and the internal Rabbit 2000 real-time clock to retain data with the RCM2100 powered down.

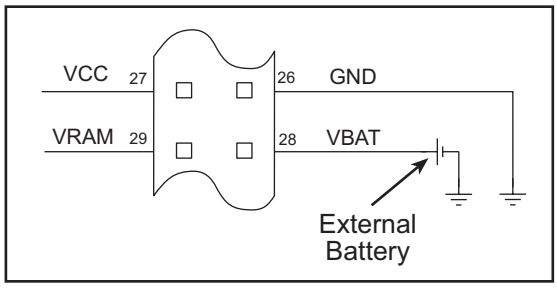


Figure C-1. External Battery Connections at Header J2

A lithium battery with a nominal voltage of 3 V and a minimum capacity of 165 mA·h is recommended. A lithium battery is strongly recommended because of its nearly constant nominal voltage over most of its life.

The drain on the battery by the RCM2100 is typically 16 µA when no other power is supplied. If a 950 mA·h battery is used, the battery can last more than 6 years:

$$\frac{950\ \text{mA}\cdot\text{h}}{16\ \mu\text{A}} = 6.8\ \text{years}.$$

The actual life in your application will depend on the current drawn by components not on the RCM2100 and the storage capacity of the battery. Note that the shelf life of a lithium battery is ultimately 10 years.

The battery-backup circuit serves three purposes:

- It reduces the battery voltage to the SRAM and to the real-time clock, thereby limiting the current consumed by the real-time clock and lengthening the battery life.
- It ensures that current can flow only ***out*** of the battery to prevent charging the battery.
- A voltage, VOSC, is supplied to U7, which keeps the 32.768 kHz oscillator working when the voltage begins to drop.

VRAM and Vcc are nearly equal (<100 mV, typically 10 mV) when power is supplied to the RCM2100.

Figure C-2 shows the RCM2100 battery-backup circuit.

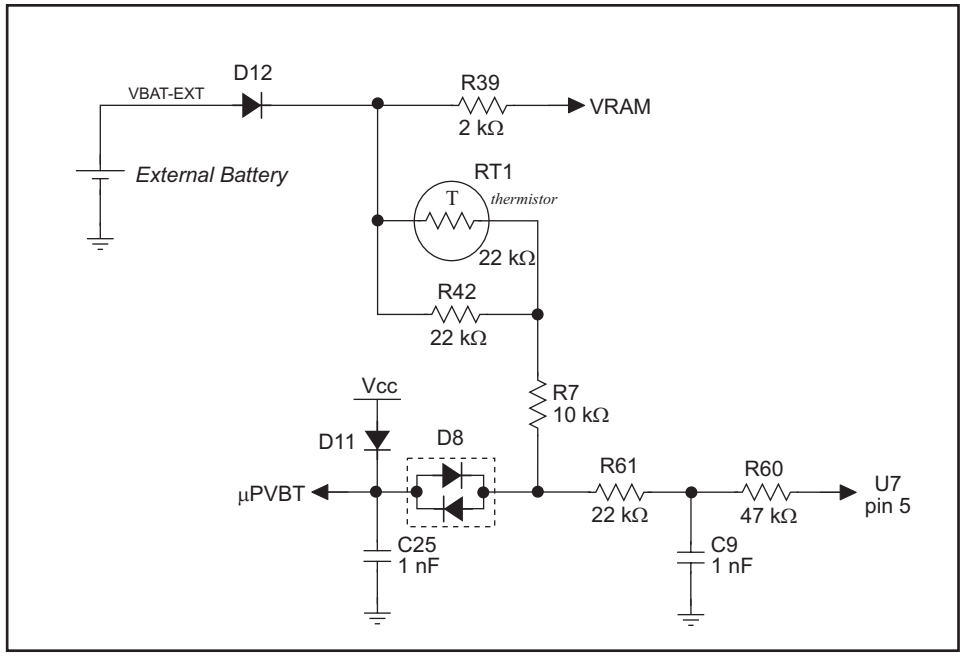


Figure C-2. RCM2100 Battery-Backup Circuit

VRAM is also available on pin 29 of header J2 to facilitate battery backup of the external circuit. Note that the recommended maximum external current draw from VRAM is 100 µA, and new battery-life calculations should be done to take external loading into account.

C.1.2 Power to VRAM Switch

The VRAM switch, shown in Figure C-3, allows a customer-supplied external battery to provide power when the external power goes off. The switch provides an isolation between Vcc and the battery when Vcc goes low. This prevents the Vcc line from draining the battery.

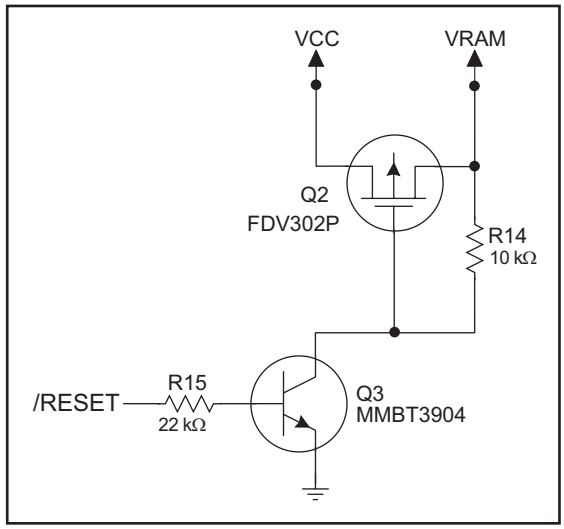


Figure C-3. VRAM Switch

Transistor Q2 is needed to provide a very small voltage drop between Vcc and VRAM (<100 mV, typically 10 mV) so that the processor lines powered by Vcc will not have a significantly different voltage than VRAM.

When the RCM2100 is not resetting (pin 2 on U3 is high), the /RESET line will be high. This turns on Q3, causing its collector to go low. This turns on Q2, allowing VRAM to nearly equal Vcc.

When the RCM2100 is resetting, the /RESET line will go low. This turns off Q2 and Q3, providing an isolation between Vcc and VRAM.

The battery-backup circuit keeps VRAM from dropping below 2 V.

C.1.3 Reset Generator

The RCM2100 uses a reset generator, U2, to reset the Rabbit 2000 microprocessor when the voltage drops below the voltage necessary for reliable operation. The reset occurs between 4.50 V and 4.75 V, typically 4.63 V. The RCM2100 has a reset output, pin 33 on header J2. The reset generator has a reset input, pin 34 on header J2, that can be used to force the RCM2100 to reset.

C.2 Chip Select Circuit

The RCM2100 has provision for battery backup, which kicks in to keep VRAM from dropping below 2 V.

The current drain on the battery in a battery-backed circuit must be kept to a minimum. When the RCM2100 is not powered, the battery keeps the SRAM memory contents and the real-time clock (RTC) going. The SRAM has a powerdown mode that greatly reduces power consumption. This powerdown mode is activated by raising the chip select (CS) signal line. Normally the SRAM requires Vcc to operate. However, only 2 V is required for data retention in powerdown mode. Thus, when power is removed from the circuit, the battery voltage needs to be provided to both the SRAM power pin and to the CS signal line. The CS control circuit accomplishes this task for the CS signal line.

Figure C-4 shows a schematic of the chip select circuit.

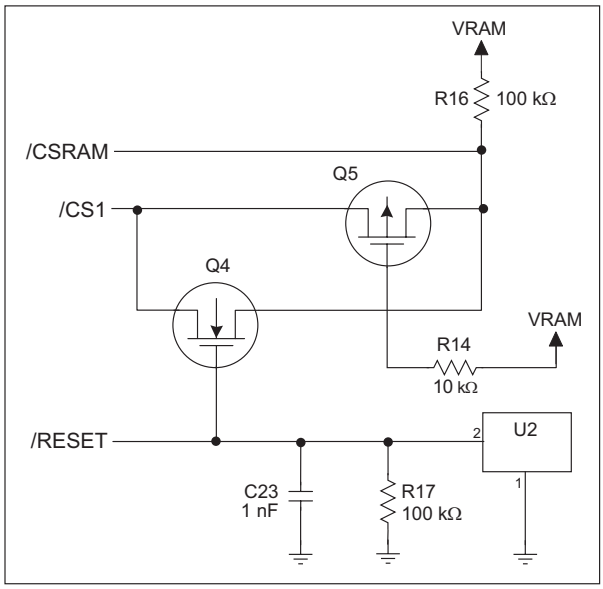


Figure C-4. Chip Select Circuit

In a powered-up condition, the CS control circuit must allow the processor's chip select signal /CS1 to control the SRAM's CS signal /CSRAM. So, with power applied, /CSRAM must be the same signal as /CS1, and with power removed, /CSRAM must be held high (but only needs to be as high as the battery voltage). Q4 and Q5 are MOSFET transistors with opposing polarity. They are both turned on when power is applied to the circuit. They allow the CS signal to pass from the processor to the SRAM so that the processor can periodically access the SRAM. When power is removed from the circuit, the transistors will turn off and isolate /CSRAM from the processor. The isolated /CSRAM line has a 100 kΩ pullup resistor to VRAM (R16). This pullup resistor keeps /CSRAM at the VRAM voltage level (which under no power condition is the backup battery's regulated voltage at a little more than 2 V).

Transistors Q4 and Q5 are of opposite polarity so that a rail-to-rail voltages can be passed. When the /CS1 voltage is low, Q4 will conduct. When the /CS1 voltage is high, Q5 will conduct. It takes time for the transistors to turn on, creating a propagation delay. This delay is typically very small, about 10 ns to 15 ns.

APPENDIX D. SAMPLE CIRCUITS

This appendix details several basic sample circuits that can be used with the RCM2100 modules.

- RS-232/RS-485 Serial Communication
- Keypad and LCD Connections
- External Memory
- D/A Converter

D.1 RS-232/RS-485 Serial Communication

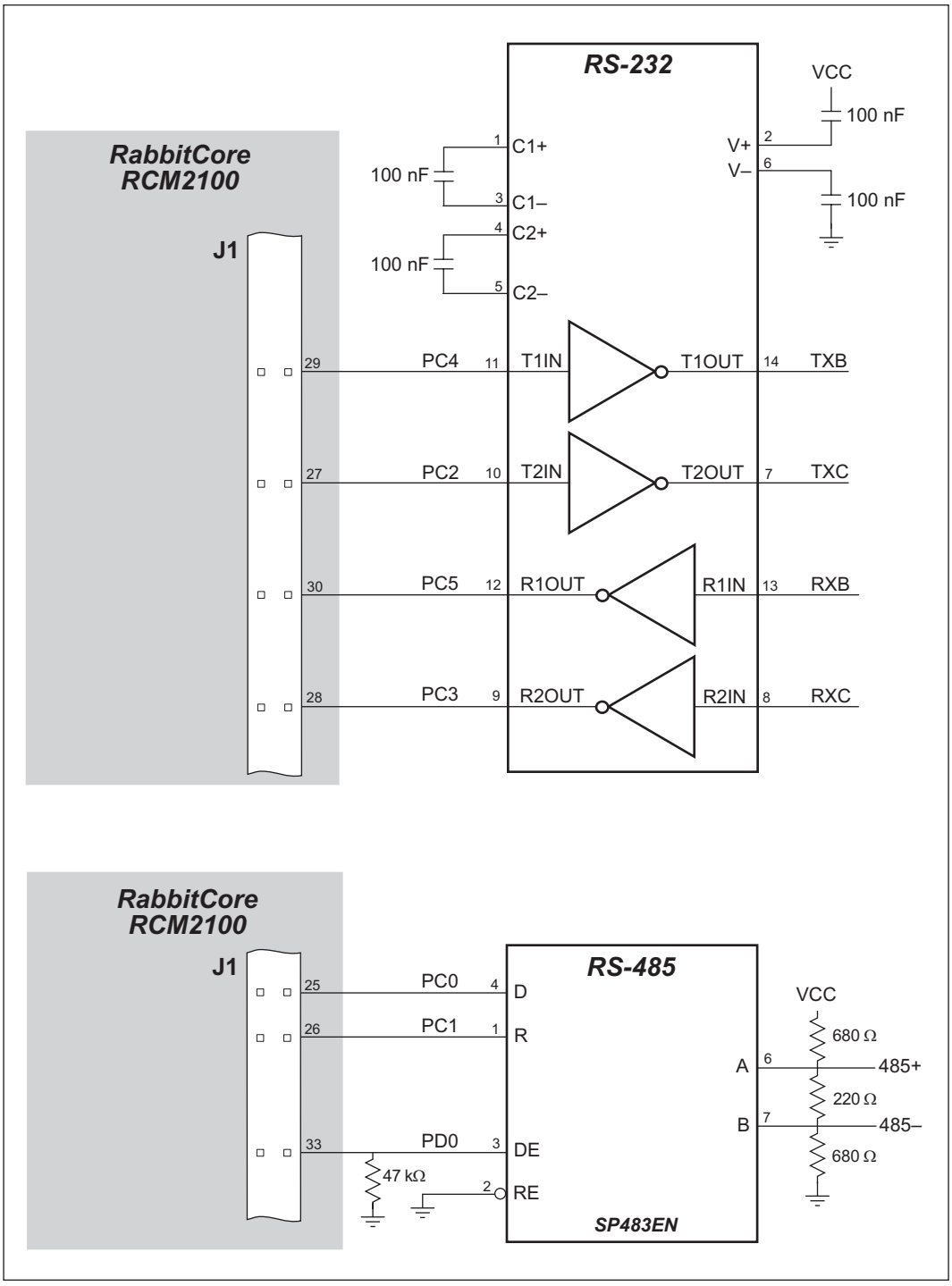


Figure D-1. Sample RS-232 and RS-485 Circuits

Sample Program: **PUTS.C** in **SAMPLES\SERIAL**.

D.2 Keypad and LCD Connections

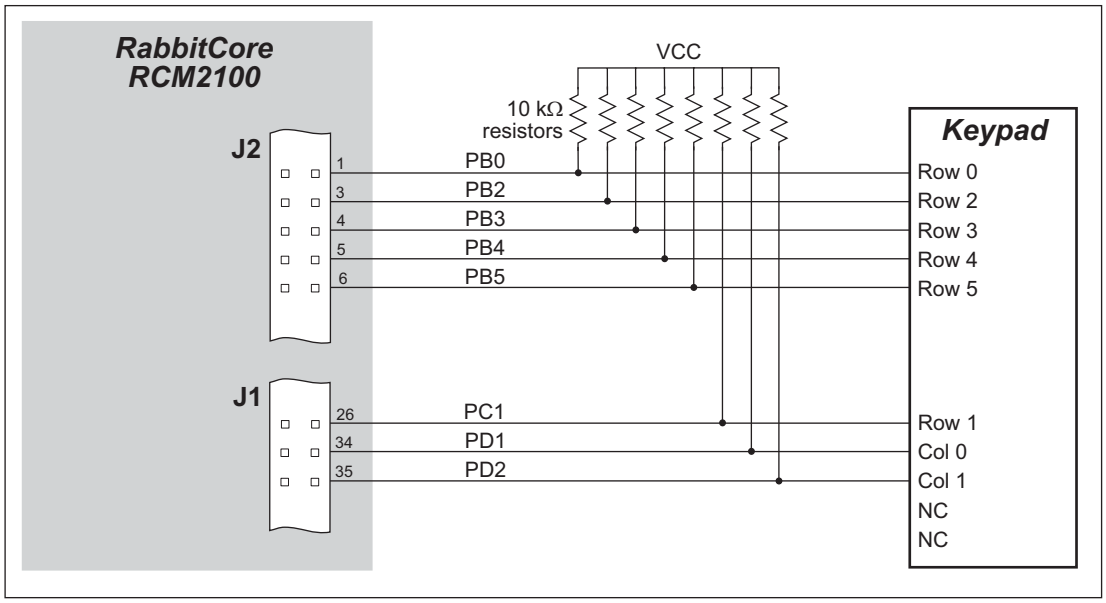


Figure D-2. Sample Keypad Connections

Sample Program: **KEYLCD2.C** in **SAMPLES\RCM2100**.

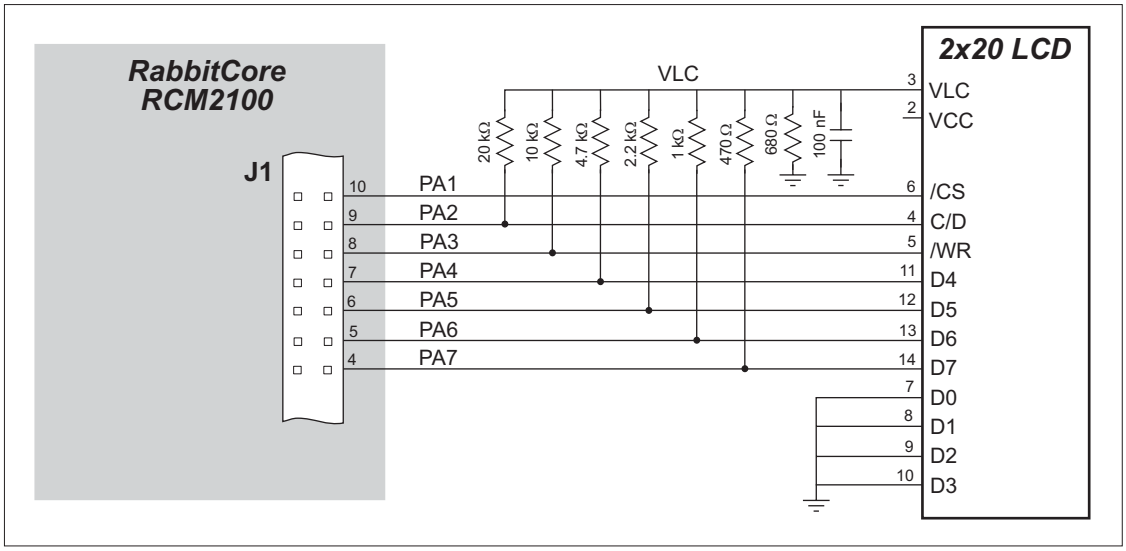


Figure D-3. Sample LCD Connections

Sample Program: **KEYLCD2.C** in **SAMPLES\RCM2100**.

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.

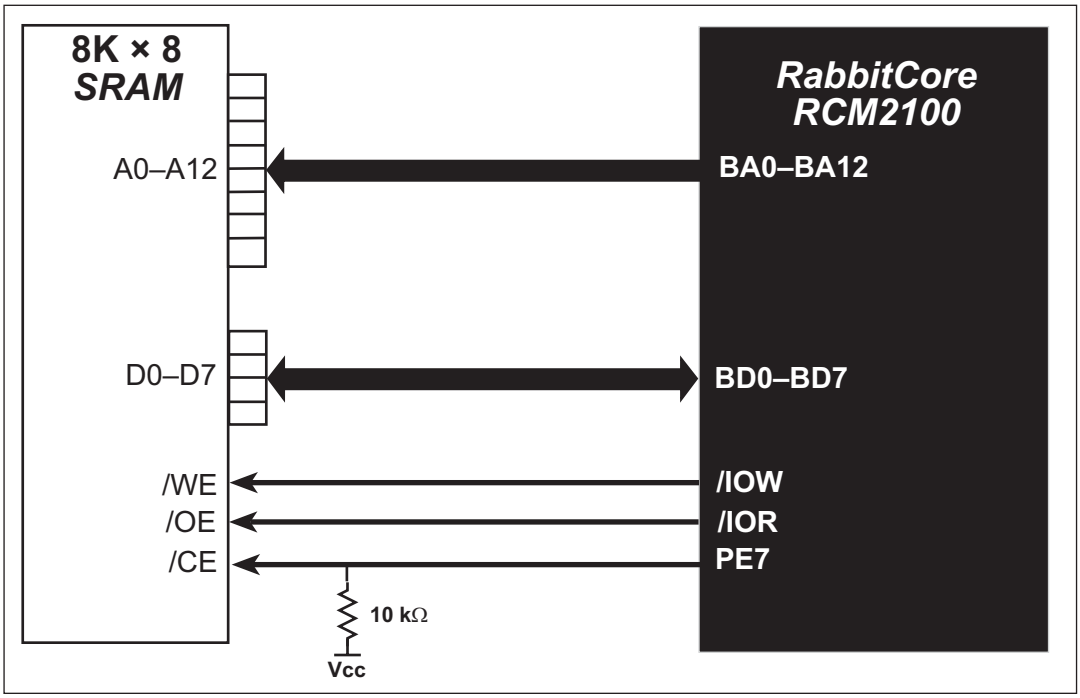


Figure D-4. Sample External Memory Connections

Sample Program: **EXTSRAM2.C** in **SAMPLES\RCM2100**.

D.4 D/A Converter

The output will initially be 0 V to -10.05 V after the first inverting op-amp, and 0 V to +10.05 V after the second inverting op-amp. All lows produce 0 V out, FF produces 10 V out. The output can be scaled by changing the feedback resistors on the op-amps. For example, changing 5.11 kΩ to 2.5 kΩ will produce an output from 0 V to -5 V. Op-amps with a very low input offset voltage are recommended.

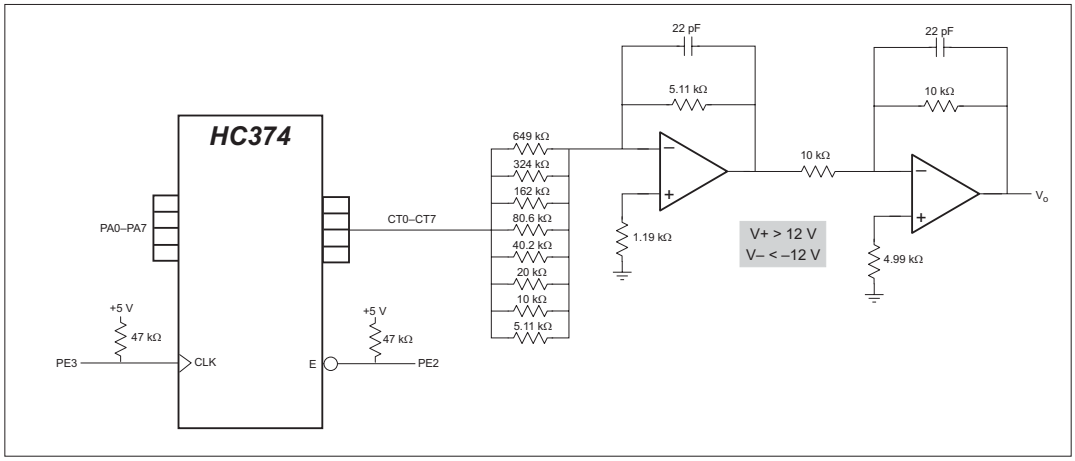


Figure D-5. Sample D/A Converter Connections

INDEX

R

S

T

U

SCHEMATICS

090-0114 RCM2100 Schematic

www.rabbit.com/documentation/schemat/090-0114.pdf

090-0116 RCM2100 Prototyping Board Schematic

www.rabbit.com/documentation/schemat/090-0116.pdf

090-0128 Programming Cable Schematic

www.rabbit.com/documentation/schemat/090-0128.pdf

You may use the URL information provided above to access the latest schematics directly.