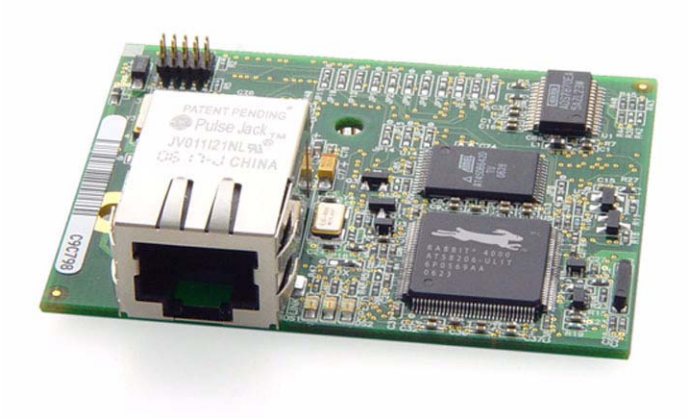
PRODUCT MANUAL



RabbitCore RCM4200

C-Programmable Analog Core Module
with Serial Flash and Ethernet

User's Manual

019–0159 • 090508–E

RabbitCore RCM4200 User's Manual

Part Number 019-0159 • 090508–E • Printed in U.S.A.

Trademarks

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

Rabbit 4000 is a trademark of Digi International Inc.

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

Digi International Inc.

www.rabbit.com

TABLE OF CONTENTS

1. INTRODUCTION

The RCM4200 series of RabbitCore modules is one of the next generation of core modules that take advantage of new Rabbit® 4000 features such as hardware DMA, clock speeds of up to 60 MHz, I/O lines shared with up to six serial ports and four levels of alternate pin functions that include variable-phase PWM, auxiliary I/O, quadrature decoder, and input capture. Coupled with more than 500 new opcode instructions that help to reduce code size and improve processing speed, this equates to a core module that is fast, efficient, and the ideal solution for a wide range of embedded applications. The RCM4200 also features an integrated 10/100Base-T Ethernet port, an A/D converter, and a serial flash memory for mass storage.

Each production model has a Development Kit with the essentials that you need to design your own microprocessor-based system, and includes a complete Dynamic C software development system. The Development Kits also contains a Prototyping Board that will allow you to evaluate the specific RCM4200 module and to prototype circuits that interface to the module. You will also be able to write and test software for the RCM4200 modules.

Throughout this manual, the term RCM4200 refers to the complete series of RCM4200 RabbitCore modules unless other production models are referred to specifically.

The RCM4200 has a Rabbit 4000 microprocessor operating at up to 58.98 MHz, static RAM, flash memory, serial flash mass-storage option, an 8-channel A/D converter, two clocks (main oscillator and timekeeping), and the circuitry necessary for reset and management of battery backup of the Rabbit 4000's internal real-time clock and 512K of static RAM. One 50-pin header brings out the Rabbit 4000 I/O bus lines, parallel ports, A/D converter channels, and serial ports.

The RCM4200 receives its +3.3 V power from the customer-supplied motherboard on which it is mounted. The RCM4200 can interface with all kinds of CMOS-compatible digital devices through the motherboard.

1.1 RCM4200 Features

- Small size: 1.84" × 2.42" × 0.84" (47 mm × 61 mm × 21 mm)
- Microprocessor: Rabbit 4000 running at up to 58.98 MHz
- Up to 33 general-purpose I/O lines configurable with up to four alternate functions
- 3.3 V I/O lines with low-power modes down to 2 kHz
- Up to six CMOS-compatible serial ports — four ports are configurable as a clocked serial ports (SPI), and two ports are configurable as SDLC/HDLC serial ports.
- Combinations of up to eight single-ended or four differential 12-bit analog inputs (RCM4200 only)
- Alternate I/O bus can be configured for 8 data lines and 6 address lines (shared with parallel I/O lines), I/O read/write
- 512K flash memory, 512K SRAM, and a fixed mass-storage flash-memory option that may be used with the standardized directory structure supported by the Dynamic C FAT File System module
- Real-time clock
- Watchdog supervisor

There are two RCM4200 production models. Table 1 summarizes their main features.

Table 1. RCM4200 Features

Feature	RCM4200	RCM4210
Microprocessor	Rabbit® 4000 at 58.98 MHz	Rabbit® 4000 at 29.49 MHz
Data SRAM	512K	
Fast Program-Execution SRAM	512K	—
Flash Memory (program)	512K	
Flash Memory (mass data storage)	8 Mbytes (serial flash)	4 Mbytes (serial flash)
A/D Converter	12 bits	—
Serial Ports	4 high-speed, CMOS-compatible ports: • all 4 configurable as asynchronous (with IrDA), 4 as clocked serial (SPI) • 1 asynchronous clocked serial port shared with programming port • 1 clocked serial port shared with serial flash • 1 clocked serial port shared with A/D converter	5 high-speed, CMOS-compatible ports: • all 5 configurable as asynchronous (with IrDA), 4 as clocked serial (SPI), and 1 as SDLC/HDLC • 1 clocked serial port shared with serial flash • 1 asynchronous clocked serial port dedicated for programming

The RCM4200 is programmed over a standard PC USB port through a programming cable supplied with the Development Kit.

NOTE: The RabbitLink cannot be used to program RabbitCore modules based on the Rabbit 4000 microprocessor.

Appendix A provides detailed specifications for the RCM4200.

1.2 Advantages of the RCM4200

- Fast time to market using a fully engineered, “ready-to-run/ready-to-program” micro-processor core.
- Competitive pricing when compared with the alternative of purchasing and assembling individual components.
- Easy C-language program development and debugging
- Rabbit Field Utility to download compiled Dynamic C .bin files, and cloning board options for rapid production loading of programs.
- Generous memory size allows large programs with tens of thousands of lines of code, and substantial data storage.

1.3 Development and Evaluation Tools

1.3.1 RCM4200 Development Kit

The RCM4200 Development Kit contains the hardware essentials you will need to use the RCM4200 module. The items in the Development Kit and their use are as follows.

- RCM4200 module.
- Prototyping Board.
- Universal AC adapter, 12 V DC, 1 A (includes Canada/Japan/U.S., Australia/N.Z., U.K., and European style plugs). Development Kits sold in North America may contain an AC adapter with only a North American style plug.
- USB programming cable with 10-pin header.
- 10-pin header to DB9 serial cable.
- ***Dynamic C***® CD-ROM, with complete product documentation on disk.
- ***Getting Started*** instructions.
- A bag of accessory parts for use on the Prototyping Board.
- ***Rabbit 4000 Processor Easy Reference*** poster.
- Registration card.

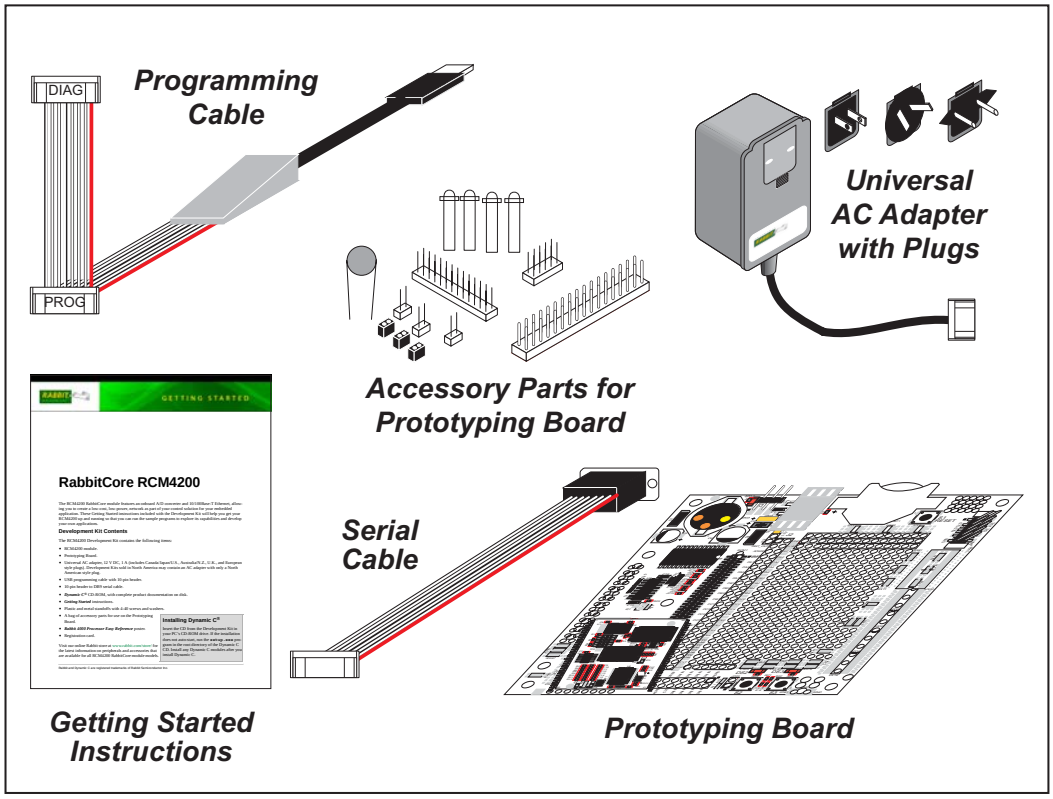


Figure 1. RCM4200 Development Kit

1.3.2 Software

The RCM4200 is programmed using version 10.09 or later of Dynamic C. A compatible version is included on the Development Kit CD-ROM.

Rabbit Semiconductor also offers add-on Dynamic C modules containing the popular µC/OS-II real-time operating system, the FAT file 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. Visit our Web site at www.rabbit.com or contact your Rabbit Semiconductor sales representative or authorized distributor for further information.

1.3.3 Online Documentation

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

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

2. GETTING STARTED

This chapter describes the RCM4200 hardware in more detail, and explains how to set up and use the accompanying Prototyping Board.

NOTE: This chapter (and this manual) assume that you have the RCM4200 Development Kit. If you purchased an RCM4200 module by itself, you will have to adapt the information in this chapter and elsewhere to your test and development setup.

2.1 Install Dynamic C

To develop and debug programs for the RCM4200 series of modules (and for all other Rabbit Semiconductor hardware), you must install and use Dynamic C.

If you have not yet installed Dynamic C version 10.09 (or a later version), do so now by inserting the Dynamic C CD from the Development Kit in your PC's CD-ROM drive. If autorun is enabled, the CD installation will begin automatically.

If autorun is disabled or the installation does not start, use the Windows **Start | Run** menu or Windows Disk Explorer to launch **setup.exe** from the root folder of the CD-ROM.

The installation program will guide you through the installation process. Most steps of the process are self-explanatory.

Dynamic C uses a COM (serial) port to communicate with the target development system. The installation allows you to choose the COM port that will be used. The default selection is COM1. You may select any available port for Dynamic C's use. If you are not certain which port is available, select COM1. This selection can be changed later within Dynamic C.

NOTE: The installation utility does not check the selected COM port in any way. Specifying a port in use by another device (mouse, modem, etc.) may lead to a message such as **"could not open serial port"** when Dynamic C is started.

Once your installation is complete, you will have up to three new icons on your PC desktop. One icon is for Dynamic C, another opens the documentation menu, and the third is for the Rabbit Field Utility, a tool used to download precompiled software to a target system.

If you have purchased any of the optional Dynamic C modules, install them after installing Dynamic C. The modules may be installed in any order. You must install the modules in the same directory where Dynamic C was installed.

2.2 Hardware Connections

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

1. Prepare the Prototyping Board for Development.
2. Attach the RCM4200 module to the Prototyping Board.
3. Connect the programming cable between the RCM4200 and the PC.
4. Connect the power supply to the Prototyping Board.

2.2.1 Step 1 — Prepare the Prototyping Board for Development

Snap in four of the plastic standoffs supplied in the bag of accessory parts from the Development Kit in the holes at the corners as shown in Figure 2.

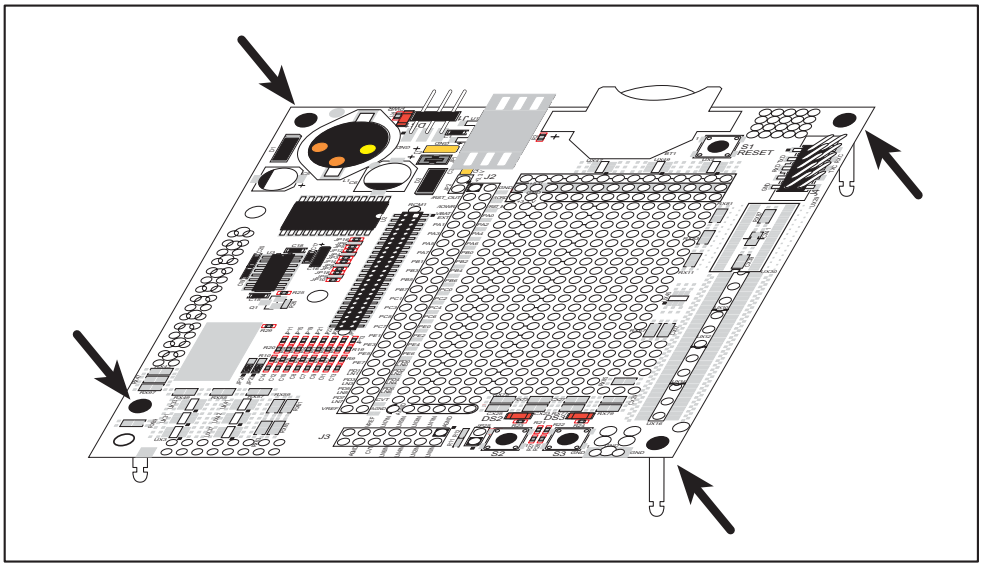


Figure 2. Insert Standoffs

2.2.2 Step 2 — Attach Module to Prototyping Board

Turn the RCM4200 module so that the mounting holes line up with the corresponding holes on the Prototyping Board. Insert the metal standoffs as shown in Figure 3, secure them from the bottom using two screws and washers, then insert the module's header J2 on the bottom side into socket RCM1 on the Prototyping Board.

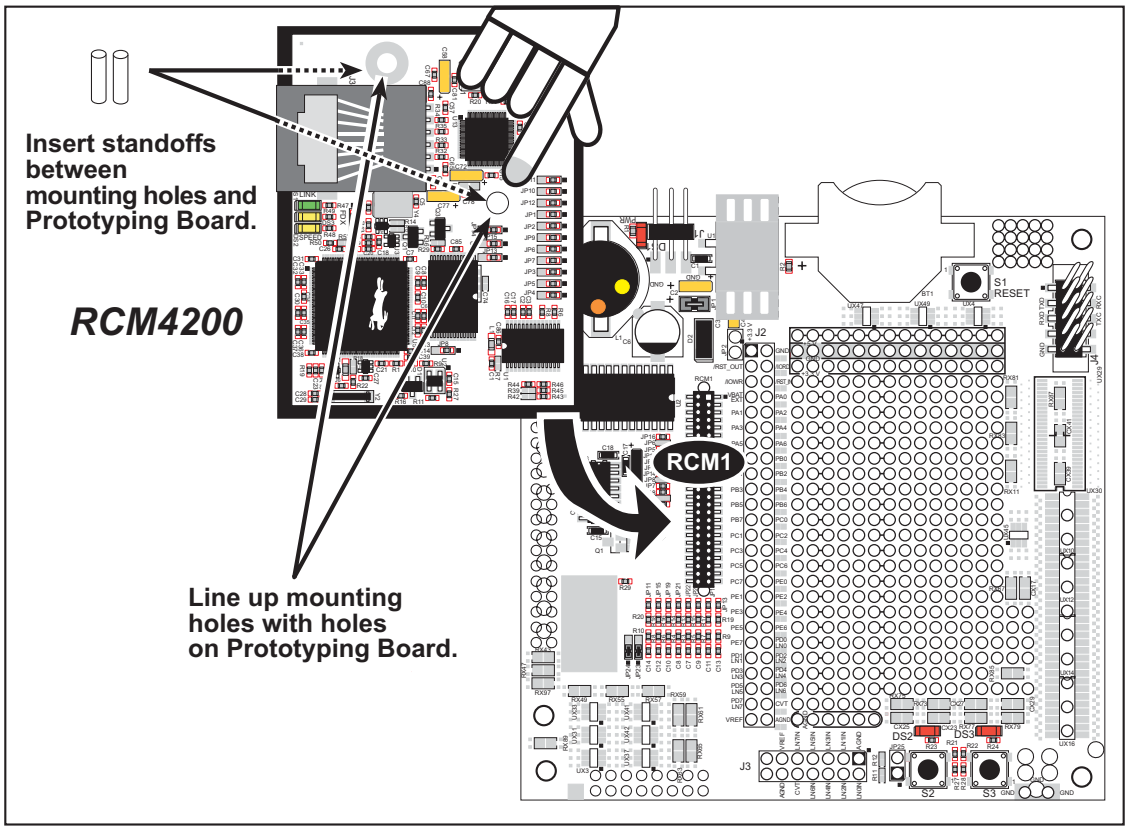


Figure 3. Install the Module on the Prototyping Board

NOTE: It is important that you line up the pins on header J2 of the module exactly with socket RCM1 on the Prototyping Board. The header pins may become bent or damaged if the pin alignment is offset, and the module will not work. Permanent electrical damage to the module may also result if a misaligned module is powered up.

Press the module's pins gently into the Prototyping Board socket—press down in the area above the header pins. For additional integrity, you may secure the RCM4200 to the standoffs from the top using the remaining two screws and washers.

2.2.3 Step 3 — Connect Programming Cable

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

Connect the 10-pin connector of the programming cable labeled **PROG** to header J1 on the RCM4200 as shown in Figure 4. Be sure to orient the marked (usually red) edge of the cable towards pin 1 of the connector. (Do not use the **DIAG** connector, which is used for a normal serial connection.)

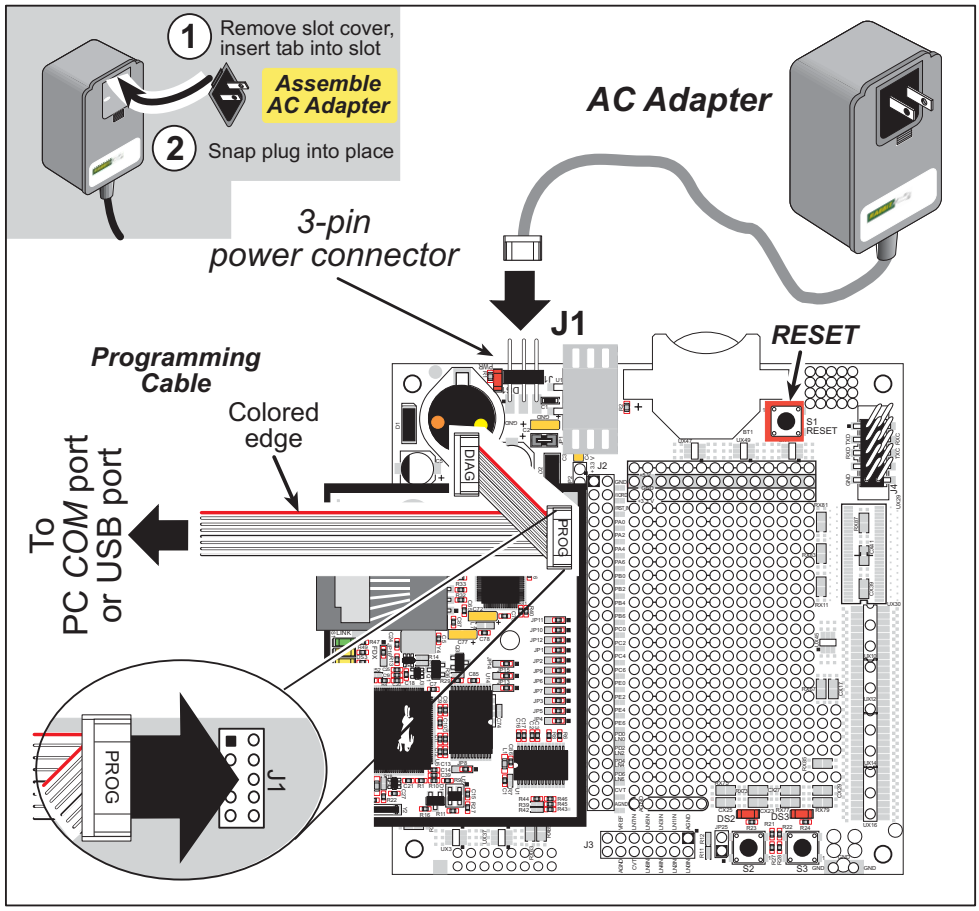


Figure 4. Connect Programming Cable and Power Supply

NOTE: Never disconnect the programming cable by pulling on the ribbon cable. Carefully pull on the connector to remove it from the header.

NOTE: Either a serial or a USB programming cable was supplied with the Development Kit. If you have a serial programming cable, an RS-232/USB converter (Rabbit Part No. 20-151-0178) is available to allow you to use the serial programming cable with a USB port.

Depending on the programming cable, connect the other end to a COM port or a USB port on your PC.

If you are using a USB programming cable, your PC should recognize the new USB hardware, and the LEDs in the shrink-wrapped area of the programming cable will flash — if you get an error message, you will have to install USB drivers. Drivers for Windows XP are available in the Dynamic C **Drivers\Rabbit USB Programming Cable\WinXP_2K** folder — double-click **DPInst.exe** to install the USB drivers. Drivers for other operating systems are available online at www.ftdichip.com/Drivers/VCP.htm.

2.2.4 Step 4 — Connect Power

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

If you have the universal AC adapter, prepare the AC adapter for the country where it will be used by selecting the appropriate plug. Snap in the top of the plug assembly into the slot at the top of the AC adapter as shown in Figure 4, then press down on the plug until it clicks into place.

Connect the AC adapter to 3-pin header J1 on the Prototyping Board as shown in Figure 4 above. The connector may be attached either way as long as it is not offset to one side—the center pin of J1 is always connected to the positive terminal, and either edge pin is ground.

Plug in the AC adapter. The **PWR** LED on the Prototyping Board next to the power connector at J1 should light up. The RCM4200 and the Prototyping Board are now ready to be used.

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

To power down the Prototyping Board, unplug the power connector from J1. You should disconnect power before making any circuit adjustments in the prototyping area, changing any connections to the board, or removing the RCM4200 from the Prototyping Board.

2.3 Run a Sample Program

Once the RCM4200 is connected as described in the preceding pages, start Dynamic C by double-clicking on the Dynamic C icon on your desktop or in your **Start** menu. For the RCM4200 model, select **Code and BIOS in Flash, Run in RAM** on the "Compiler" tab in the Dynamic C **Options > Project Options** menu. (Select **Code and BIOS in Flash** for the RCM4210.) Click **OK**.

If you are using a USB port to connect your computer to the RCM4200, click on the "Communications" tab and verify that **Use USB to Serial Converter** is selected to support the USB programming cable. Click **OK**. You may have to determine which COM port was assigned to the RS-232/USB converter. Open **Control Panel > System > Hardware > Device Manager > Ports** and identify which COM port is used for the USB connection. In Dynamic C, select **Options > Project Options**, then select this COM port on the **Communications** tab, then click **OK**. You may type the COM port number followed by **Enter** on your computer keyboard if the COM port number is outside the range on the dropdown menu.

Now find the file **PONG.C**, which is in the Dynamic C **SAMPLES** folder. To run the program, open it with the **File** menu, compile it using the **Compile** menu, and then run it by selecting **Run** in the **Run** menu. The **STDIO** window will open on your PC and will display a small square bouncing around in a box.

2.3.1 Troubleshooting

If you receive the message **No Rabbit Processor Detected**, the programming cable may be connected to the wrong COM port, a connection may be faulty, or the target system may not be powered up. First, check to see that the power LED on the Prototyping Board is lit. If the LED is lit, check both ends of the programming cable to ensure that it is firmly plugged into the PC and the programming header on the RCM4200 with the marked (colored) edge of the programming cable towards pin 1 of the programming header. Ensure that the module is firmly and correctly installed in its connectors on the Prototyping Board.

If Dynamic C appears to compile the BIOS successfully, but you then receive a communication error message when you compile and load a 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 no faults with the hardware, check that you have selected the correct COM port within Dynamic C as explained for the USB port above. 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 for another available COM port. You should receive a **`Bios compiled successfully`** message once this step is completed successfully.

2.4 Where Do I Go From Here?

If the sample program ran fine, you are now ready to go on to the sample programs in Chapter 3 and to develop your own applications. The sample programs can be easily modified for your own use. The user's manual also provides complete hardware reference information and software function calls for the RCM4200 series of modules and the Prototyping Board.

For advanced development topics, refer to the ***Dynamic C User's Manual***, also in the online documentation set.

2.4.1 Technical Support

NOTE: If you purchased your RCM4200 through a distributor or through a Rabbit partner, contact the distributor or partner first for technical support.

If there are any problems at this point:

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

3. RUNNING SAMPLE PROGRAMS

To develop and debug programs for the RCM4200 (and for all other Rabbit Semiconductor hardware), you must install and use Dynamic C. This chapter provides a tour of its major features with respect to the RCM4200.

3.1 Introduction

To help familiarize you with the RCM4200 modules, Dynamic C includes several sample programs. Loading, executing and studying these programs will give you a solid hands-on overview of the RCM4200's capabilities, as well as a quick start with Dynamic C as an application development tool.

> **NOTE:** The sample programs assume 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.

In order to run the sample programs discussed in this chapter and elsewhere in this manual,

1. Your module must be plugged in to the Prototyping Board as described in Chapter 2, "Getting Started."
2. Dynamic C must be installed and running on your PC.
3. The programming cable must connect the programming header on the module to your PC.
4. Power must be applied to the module through the Prototyping Board.

Refer to Chapter 2, "Getting Started," if you need further information on these steps.

To run a sample program, open it with the **File** menu (if it is not still open), then compile and run it by pressing **F9**.

Each sample program has comments that describe the purpose and function of the program. Follow the instructions at the beginning of the sample program.

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

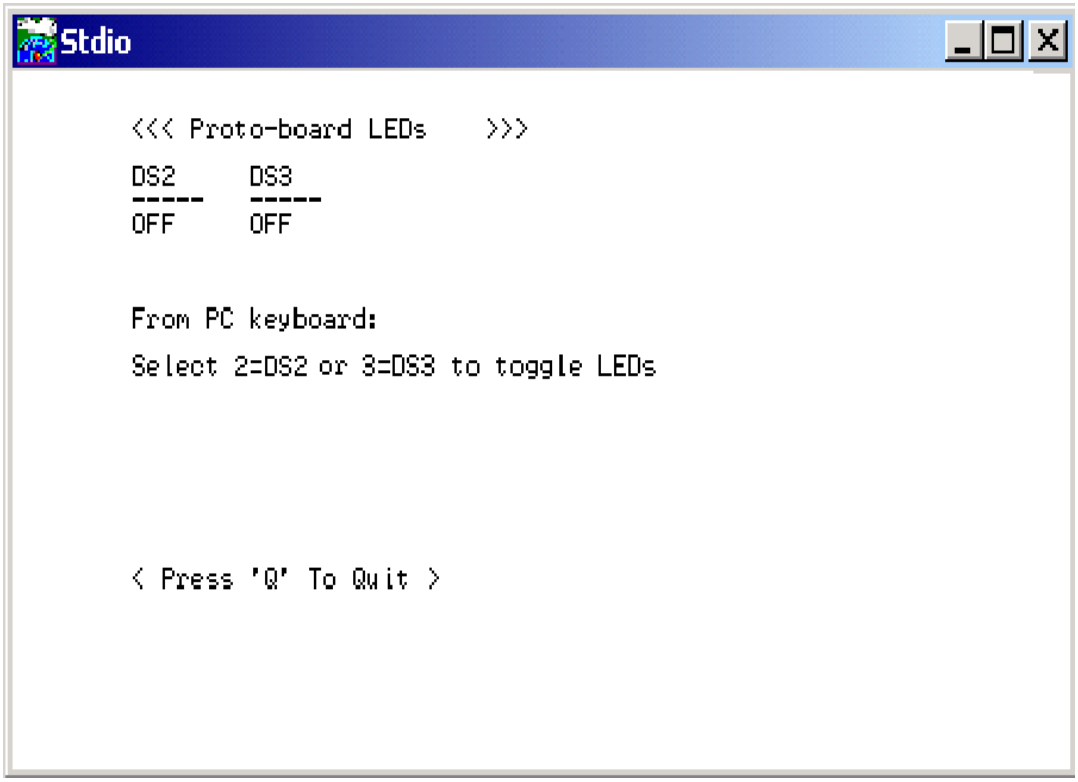
3.2 Sample Programs

Of the many sample programs included with Dynamic C, several are specific to the RCM4200 modules. These programs will be found in the **SAMPLES\RCM4200** folder.

- **CONTROLLED.C**—Demonstrates use of the digital outputs by having you turn LEDs DS2 and DS3 on the Prototyping Board on or off from the **STDIO** window on your PC.

  Parallel Port B bit 2 = LED DS2
  Parallel Port B bit 3 = LED DS3

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



  Press "2" or "3" on your keyboard to select LED DS2 or DS3 on the Prototyping Board. Then follow the prompt in the Dynamic C **STDIO** window to turn the LED ON or OFF. A logic low will light up the LED you selected.

- **FLASHLED1.C**—demonstrates the use of assembly language to flash LEDs DS2 and DS3 on the Prototyping Board at different rates. Once you have compiled and run this program, LEDs DS2 and DS3 will flash on/off at different rates.

- **FLASHLED2.C**—demonstrates the use of cofunctions and costatements to flash LEDs DS2 and DS3 on the Prototyping Board at different rates. Once you have compiled and run this program, LEDs DS2 and DS3 will flash on/off at different rates.

- **TAMPERDETECTION.C**—demonstrates how to detect an attempt to enter the bootstrap mode. When an attempt is detected, the battery-backed onchip-encryption RAM on the Rabbit 4000 is erased. This battery-backed onchip-encryption RAM can be useful to store data such as an AES encryption key from a remote location.

  This sample program shows how to load and read the battery-backed onchip-encryption RAM and how to enable a visual indicator.

  Once this sample is compiled and running (you pressed the **F9** key while the sample program is open), remove the programming cable and press the reset button on the Prototyping Board to reset the module. LEDs DS2 and DS3 will be flashing on and off.

  Now press switch S2 to load the battery-backed RAM with the encryption key. The LEDs are now on continuously. Notice that the LEDs will stay on even when you press the reset button on the Prototyping Board.

  Reconnect the programming cable briefly and unplug it again to simulate an attempt to access the onchip-encryption RAM. The LEDs will be flashing because the battery-backed onchip-encryption RAM has been erased. Notice that the LEDs will continue flashing even when you press the reset button on the Prototyping Board.

  You may press switch S2 again and repeat the last steps to watch the LEDs.

- **TOGGLESWITCH.C**—demonstrates the use of costatements to detect switch presses using the press-and-release method of debouncing. LEDs DS2 and DS3 on the Prototyping Board are turned on and off when you press switches S2 and S3. S2 and S3 are controlled by PB4 and PB5 respectively.

Once you have loaded and executed these five programs and have an understanding of how Dynamic C and the RCM4200 modules interact, you can move on and try the other sample programs, or begin building your own.

3.2.1 Use of Serial Flash

The following sample programs can be found in the **SAMPLES\RCM4200\Serial_Flash** folder.

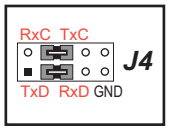
- **SERIAL_FLASHLOG.C**—This program runs a simple Web server and stores a log of hits on the home page of the serial flash "server." This log can be viewed and cleared from a browser at http://10.10.6.100/. You will likely have to first "configure" your network interface card for a "10Base-T Half-Duplex," "100Base-T Half-Duplex," or an "Auto-Negotiation" connection on the "Advanced" tab, which is accessed from the control panel (**Start > Settings > Control Panel**) by choosing **Network Connections**.
- **SFLASH_INSPECT.C**—This program is a handy utility for inspecting the contents of a serial flash chip. When the sample program starts running, it attempts to initialize a serial flash chip on Serial Port C. Once a serial flash chip is found, the user can perform five different commands to print out the contents of a specified page, set all bytes on the specified page to a single random value, clear (set to zero) all the bytes in a specified page, set all bytes on the specified page to a given value, or save user-specified text to a selected page.

3.2.2 Serial Communication

The following sample programs are found in the **SAMPLES\RCM4200\SERIAL** folder.

- **FLOWCONTROL.C**—This program demonstrates how to configure Serial Port D for CTS/RTS flow control with serial data coming from Serial Port C (TxC) at 115,200 bps. The serial data received are displayed in the **STDIO** window.

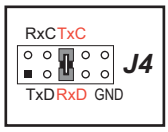
  To set up the Prototyping Board, you will need to tie TxD and RxD together on the RS-232 header at J4, and you will also tie TxC and RxC together using the jumpers supplied in the Development Kit as shown in the diagram.



  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 flow control.

  If you have two Prototyping Boards with modules, run this sample program on the sending board, then disconnect the programming cable and reset the sending board so that the module is operating in the Run mode. Connect TxC, TxD, and GND on the sending board to RxC, RxD, and GND on the other board, then, with the programming cable attached to the other module, run the sample program.

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

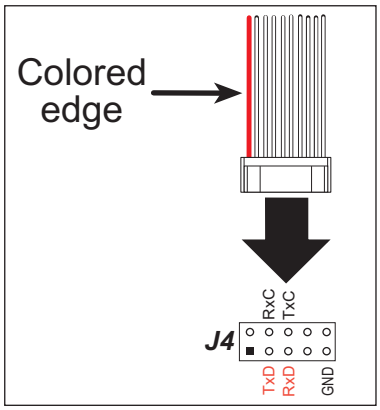


  To set up the Prototyping Board, you will need to tie TxC and RxD together on the RS-232 header at J4 using one of the jumpers supplied in the Development Kit as shown in the diagram.

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

- **SERDMA.C**—This program demonstrates using DMA to transfer data from a circular buffer to the serial port and vice versa. The Dynamic C **STDIO** window is used to view or clear the buffer.

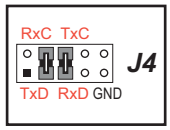
  Before you compile and run the sample program, you will need to connect the RS-232 header at J4 to your PC as shown in the diagram using the serial to DB9 cable supplied in the Development Kit.



  Once you have compiled and run the sample program, start Tera Term or another terminal emulation program to connect to the selected PC serial port at a baud rate of 115,200 bps. You can observe the output in the Dynamic C **STDIO** window as you type in Tera Term, and you can also use the Dynamic C **STDIO** window to clear the buffer.

  The Tera Term utility can be downloaded from hp.vector.co.jp/authors/VA002416/teraterm.html.

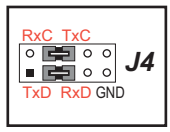
- **SIMPLE3WIRE.C**—This program demonstrates basic RS-232 serial communication. Lower case characters are sent on TxC, and are received by RxD. The received characters are converted to upper case and are sent out on TxD, are received on RxC, and are displayed in the Dynamic C **STDIO** window.



  To set up the Prototyping Board, you will need to tie TxD and RxC together on the RS-232 header at J4, and you will also tie RxD and TxC together using the jumpers supplied in the Development Kit as shown in the diagram.

- **SIMPLE5WIRE.C**—This program demonstrates 5-wire RS-232 serial communication with flow control on Serial Port D and data flow on Serial Port C.

  To set up the Prototyping Board, you will need to tie TxD and RxD together on the RS-232 header at J4, and you will also tie TxC and RxC together using the jumpers supplied in the Development Kit as shown in the diagram.

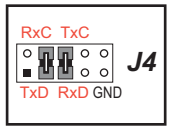


  Once you have compiled and run this program, you can test flow control by disconnecting the TxD jumper from RxD while the program is running. Characters will no longer appear in the **STDIO** window, and will display again once TxD is connected back to RxD.

  If you have two Prototyping Boards with modules, run this sample program on the sending board, then disconnect the programming cable and reset the sending board so that the module is operating in the Run mode. Connect TxC, TxD, and GND on the sending board to RxC, RxD, and GND on the other board, then, with the programming cable attached to the other module, run the sample program. Once you have compiled and run this program, you can test flow control by disconnecting TxD from RxD as before while the program is running. Since the J4 header locations on the two Prototyping Boards are connected with wires, there are no slip-on jumpers at J4 on either Prototyping Board.

- **SWITCHCHAR.C**—This program demonstrates transmitting and then receiving an ASCII string on Serial Ports C and D. It also displays the serial data received from both ports in the **STDIO** window.

  To set up the Prototyping Board, you will need to tie TxD and RxC together on the RS-232 header at J4, and you will also tie RxD and TxC together using the jumpers supplied in the Development Kit as shown in the diagram.



  Once you have compiled and run this program, press and release switches S2 and S3 on the Prototyping Board. The data sent between the serial ports will be displayed in the **STDIO** window.

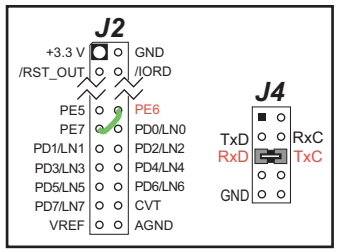
- **IOCONFIG_SWITCHECHO.C**—This program demonstrates how to set up Serial Port E, which then transmits and then receives an ASCII string when switch S2 is pressed. The echoed serial data are displayed in the Dynamic C **STDIO** window.

  Note that the I/O lines that carry the Serial Port E signals are not the Rabbit 4000 defaults. The Serial Port E I/O lines are configured by calling the library function **serEconfig()** that was generated by the Rabbit 4000 **IOCONFIG.EXE** utility program. Serial Port E is configured to use Parallel Port E bits PD6 and PD7. These signals are available on the Prototyping Board's Module Extension Header (header J2).

  Serial Port D is left in its default configuration, using Parallel Port C bits PC0 and PC1. These signals are available on the Prototyping Board's RS-232 connector (header J4). Serial Port D transmits and then receives an ASCII string when switch S3 is pressed.

  Also note that there is one library generated by **IOCONFIG.EXE** in the Dynamic C **SAMPLES\RCM4200\SERIAL** folder for the 29 MHz RCM4210.

  To set up the Prototyping Board, you will need to tie TxD and RxD together on the RS-232 header at J4 using the jumpers supplied in the Development Kit; you will also tie TxE (PD6) and RxE (PD7) together with a soldered wire or with a wire jumper if you have soldered in the IDC header supplied with the accessory parts in the Development Kit.



  Once you have compiled and run this program, press and release switches S2 or S3 on the Prototyping Board. The data echoed between the serial ports will be displayed in the **STDIO** window.

3.2.3 A/D Converter Inputs (RCM4200 only)

The following sample programs are found in the **SAMPLES\RCM4200\ADC** folder.

- **AD_CAL_ALL.C**—Demonstrates how to recalibrate all the single-ended analog input channels with one gain using two known voltages to generate the calibration constants for each channel. The constants will be written into the user block data area.

  Connect a positive voltage from 0–20 V DC (for example, the power supply positive output) to analog input channels LN0IN–LN6IN on the Prototyping Board, and connect the ground to GND. Use a voltmeter to measure the voltage, and follow the instructions in the Dynamic C **STDIO** window once you compile and run this sample program. Remember that analog input LN7 on the Prototyping Board is used with the thermistor and is not be used with this sample program.

  NOTE: The above sample program will overwrite the existing calibration constants.

- **AD_CAL_CHAN.C**—Demonstrates how to recalibrate one single-ended analog input channel with one gain using two known voltages to generate the calibration constants for that channel. The constants will be rewritten into the user block data area.

  Connect a positive voltage from 0–20 V DC (for example, the power supply positive output) to an analog input channel on the Prototyping Board, and connect the ground to GND. Use a voltmeter to measure the voltage, and follow the instructions in the Dynamic C **STDIO** window once you compile and run this sample program. Remember that analog input LN7 on the Prototyping Board is used with the thermistor and is not be used with this sample program.

  NOTE: The above sample program will overwrite the existing calibration constants for the selected channel.

- **AD_RDVOLT_ALL.C**—Demonstrates how to read all single-ended A/D input channels using previously defined calibration constants. The constants used to compute equivalent voltages are read from the user block data area, so the sample program cannot be run using the "Code and BIOS in RAM" compiler option.

  Compile and run this sample program once you have connected a positive voltage from 0–20 V DC (for example, the power supply positive output) to analog input channels LN0IN–LN6IN on the Prototyping Board, and ground to GND. Follow the prompts in the Dynamic C **STDIO** window. Raw data and the computed equivalent voltages will be displayed. Remember that analog input LN7 on the Prototyping Board is used with the thermistor and is not be used with this sample program.

- **AD_SAMPLE.C**—Demonstrates how to how to use a low level driver on single-ended inputs. The program will continuously display the voltage (averaged over 10 samples) that is present on an A/D converter channel (except LN7). The constants used to compute equivalent voltages are read from the user block data area, so the sample program cannot be run using the "Code and BIOS in RAM" compiler option.

  Compile and run this sample program once you have connected a positive voltage from 0–20 V DC to an analog input (except LN7) on the Prototyping Board, and ground to GND. Follow the prompts in the Dynamic C **STDIO** window. Raw data and the computed equivalent voltages will be displayed. If you attach a voltmeter between the analog input and ground, you will be able to observe that the voltage in the Dynamic C **STDIO** window tracks the voltage applied to the analog input as you vary it.

- **THERMISTOR.C**—Demonstrates how to use analog input LN7 to calculate temperature for display to the Dynamic C **STDIO** window. This sample program assumes that the thermistor is the one included in the Development Kit whose values for beta, series resistance, and resistance at standard temperature are given in the part specification.

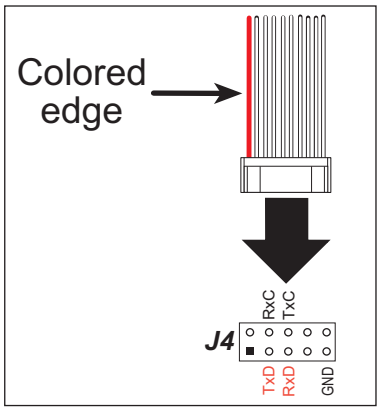
  Install the thermistor at location JP25 on the Prototyping Board before running this sample program. Observe the temperature changes shown in the Dynamic C **STDIO** window as you apply heat or cold air to the thermistor.

3.2.3.1 Downloading and Uploading Calibration Constants

The Tera Term utility called for in these sample programs can be downloaded from hp.vector.co.jp/authors/VA002416/teraterm.html.

These sample programs must be compiled to flash memory. To do so, select **Options > Project Options** in Dynamic C, then select the "Compiler" tab, and select "Code and BIOS in Flash" for the **BIOS Memory Setting**.

Before you compile and run these sample programs, you will also need to connect the RS-232 header at J4 to your PC as shown in the diagram using the serial to DB9 cable supplied in the Development Kit.



- **DNLOADCALIB.C**—Demonstrates how to retrieve analog calibration data to rewrite it back to the user block using a terminal emulation utility such as Tera Term.

  Start Tera Term or another terminal emulation program on your PC, and configure the serial parameters as follows.

  - Baud rate 19,200 bps, 8 bits, no parity, 1 stop bit
  - Enable **Local Echo** option
  - Feed options — **Receive = CR**, **Transmit = CR + LF**

  Now compile and run this sample program. Verify that the message "Waiting, Please Send Data file" message is being display in the Tera Term display window before proceeding.

  Within Tera Term, select **File-->Send File-->Path and filename**, then select the **OPEN** option within the dialog box. Once the data file has been downloaded, Tera Term will indicate whether the calibration data were written successfully.

- **UPLOADCALIB.C**—Demonstrates how to read the analog calibration constants from the user block using a terminal emulation utility such as Tera Term.

  Start Tera Term or another terminal emulation program on your PC, and configure the serial parameters as follows.

- Baud rate 19,200 bps, 8 bits, no parity, 1 stop bit
- Enable **Local Echo** option
- Feed options — **Receive = CR**, **Transmit = CR + LF**

Follow the remaining steps carefully in Tera Term to avoid overwriting previously saved calibration data when using same the file name.

- Enable the **File APPEND** option at the bottom of the dialog box
- Select the **OPEN** option at the right-hand side of the dialog box

Tera Term is now ready to log all data received on the serial port to the file you specified.

You are now ready to compile and run this sample program. A message will be displayed in the Tera Term display window once the sample program is running.

Enter the serial number you assigned to your RabbitCore module in the Tera Term display window, then press the **ENTER** key. The Tera Term display window will now display the calibration data.

Now select **CLOSE** from within the Tera Term LOG window, which will likely be a separate pop-up window minimized at the bottom of your PC screen. This finishes the logging and closes the file.

Open your data file and verify that the calibration data have been written properly. A sample is shown below.

```
Serial port transmission
========================
Uploading calibration table . . .
Enter the serial number of your controller = 9MN234
SN9MN234
ADSE
0
float_gain,float_offset,float_gain,float_offset,float_gain,float_offset,float_gain,float_offset,
float_gain,float_offset,float_gain,float_offset,float_gain,float_offset,float_gain,float_offset,
1
float_gain,float_offset,float_gain,float_offset,float_gain,float_offset,float_gain,float_offset,
float_gain,float_offset,float_gain,float_offset,float_gain,float_offset,float_gain,float_offset,
          |
          |
ADDF
0
float_gain,float_offset,float_gain,float_offset,float_gain,float_offset,float_gain,float_offset,
float_gain,float_offset,float_gain,float_offset,float_gain,float_offset,float_gain,float_offset,
2
float_gain,float_offset,float_gain,float_offset,float_gain,float_offset,float_gain,float_offset,
float_gain,float_offset,float_gain,float_offset,float_gain,float_offset,float_gain,float_offset,
          |
          |
ADMA
3
float_gain,float_offset,
4
float_gain,float_offset,
          |
          |
END
```

3.2.4 Real-Time Clock

If you plan to use the real-time clock functionality in your application, you will need to set the real-time clock. Set the real-time clock using the **SETRTCKB.C** sample program from the Dynamic C **SAMPLES\RTCLOCK** folder, using the onscreen prompts. The **RTC_TEST.C** sample program in the Dynamic C **SAMPLES\RTCLOCK** folder provides additional examples of how to read and set the real-time clock.

4. HARDWARE REFERENCE

Chapter 4 describes the hardware components and principal hardware subsystems of the RCM4200. Appendix A, "RCM4200 Specifications," provides complete physical and electrical specifications.

Figure 5 shows the Rabbit-based subsystems designed into the RCM4200.

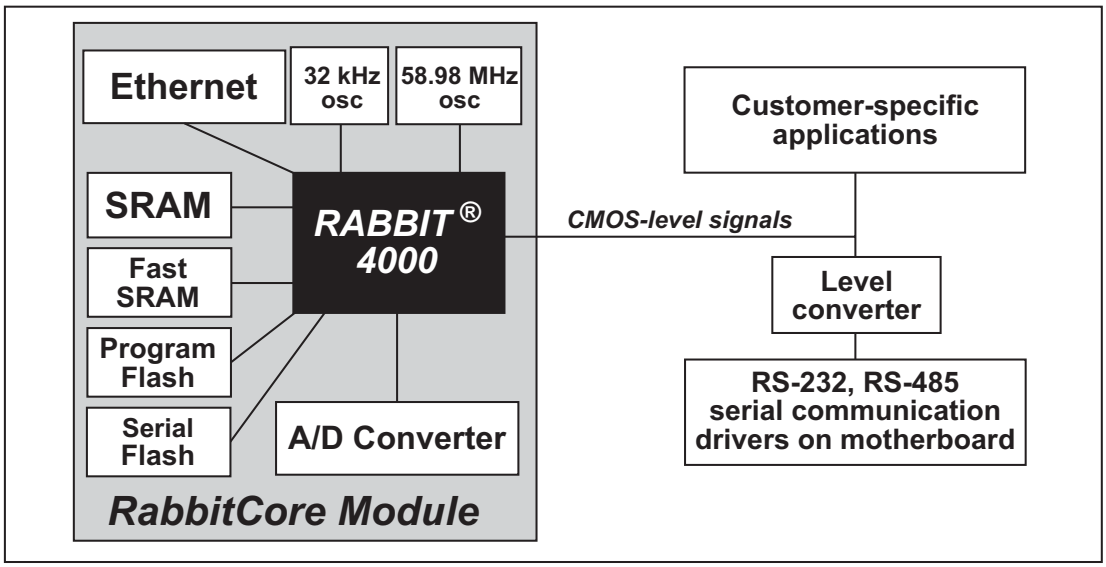


Figure 5. RCM4200 Subsystems

4.1 RCM4200 Digital Inputs and Outputs

Figure 6 shows the RCM4200 pinouts for header J2.

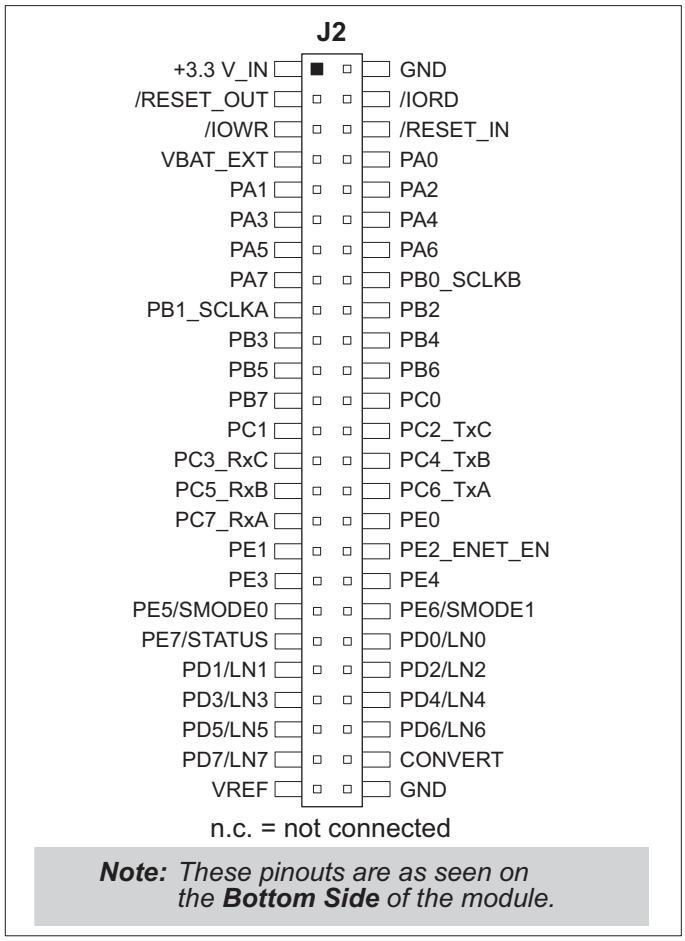


Figure 6. RCM4200 Pinout

Headers J2 is a standard 2 × 25 IDC header with a nominal 1.27 mm pitch.

Figure 7 shows the use of the Rabbit 4000 microprocessor ports in the RCM4200 modules.

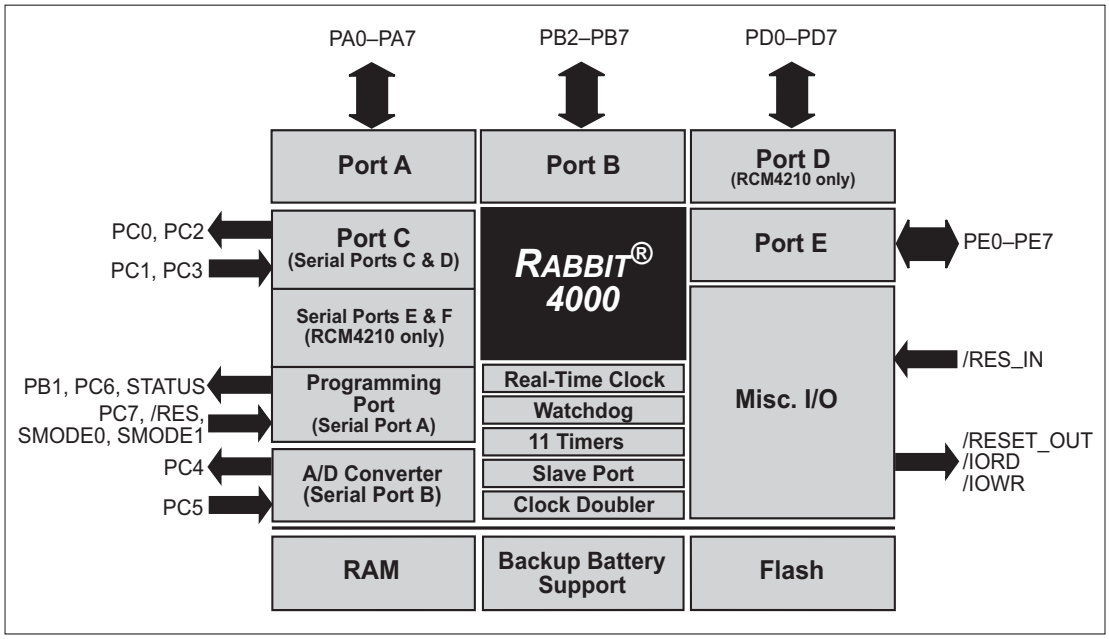


Figure 7. Use of Rabbit 4000 Ports

The ports on the Rabbit 4000 microprocessor used in the RCM4200 are configurable, and so the factory defaults can be reconfigured. Table 2 lists the Rabbit 4000 factory defaults and the alternate configurations.

Table 2. RCM4200 Pinout Configurations

Pin	Pin Name	Default Use	Alternate Use	Notes
1	+3.3 V_IN			
2	GND			
3	/RES_OUT	Reset output	Reset input	Reset output from Reset Generator or external reset input
4	/IORD	Output		External I/O read strobe
5	/IOWR	Output		External I/O write strobe
6	/RESET_IN	Input		Input to Reset Generator
7	VBAT_EXT	Battery input		
8–15	PA[0:7]	Input/Output	Slave port data bus (SD0–SD7) External I/O data bus (ID0–ID7)	
16	PB0	Input/Output	SCLKB External I/O Address IA6	SCLKB (used by RCM4200 A/D converter — see Section 4.2.1)
17	PB1	Input/Output	SCLKA External I/O Address IA7	Programming port CLKA
18	PB2	Input/Output	/SWR External I/O Address IA0	
19	PB3	Input/Output	/SRD External I/O Address IA1	
20	PB4	Input/Output	SA0 External I/O Address IA2	
21	PB5	Input/Output	SA1 External I/O Address IA3	
22	PB6	Input/Output	/SCS External I/O Address IA4	
23	PB7	Input/Output	/SLAVATN External I/O Address IA5	

Table 2. RCM4200 Pinout Configurations (continued)

Pin	Pin Name	Default Use	Alternate Use	Notes
24	PC0	Input/Output	TXD I/O Strobe I0 Timer C0 TCLKF	Serial Port D
25	PC1	Input/Output	RXD/TXD I/O Strobe I1 Timer C1 RCLKF Input Capture	
26	PC2	Input/Output	TXC/TXF I/O Strobe I2 Timer C2	Serial Port C (shared by serial flash)
27	PC3	Input/Output	RXC/TXC/RXF I/O Strobe I3 Timer C3 SCLKD Input Capture	
28	PC4	Input/Output	TXB I/O Strobe I4 PWM0 TCLKE	Serial Port B (shared by RCM4200 A/D converter)
29	PC5	Input/Output	RXB/TXB I/O Strobe I5 PWM1 RCLKE Input Capture	
30	PC6	Input/Output	TXA/TXE I/O Strobe I6 PWM2	Programming port
31	PC7	Input/Output	RXA/TXA/RXE I/O Strobe I7 PWM3 SCLKC Input Capture	
32	PE0	Input/Output	I/O Strobe I0 A20 Timer C0 TCLKF INT0 QRD1B	

Table 2. RCM4200 Pinout Configurations (continued)

Pin	Pin Name	Default Use	Alternate Use	Notes
33	PE1	Input/Output	I/O Strobe I1 A21 Timer C1 RXD/RCLKF INT1 QRD1A Input Capture	
34	PE2	Input/Output	I/O Strobe I2 A22 Timer C2 TXF DREQ0 QRD2B	Ethernet enable
35	PE3	Input/Output	I/O Strobe I3 A23 Timer C3 RXC/RXF/SCLKD DREQ1 QRD2A Input Capture	
36	PE4	Input/Output	I/O Strobe I4 /A0 INT0 PWM0 TCLKE	
37	PE5/SMODE0	Input/Output	I/O Strobe I5 INT1 PWM1 RXB/RCLKE Input Capture	PE5 is the default configuration
38	PE6/SMODE1	Input/Output	I/O Strobe I6 PWM2 TXE DREQ0	PE6 is the default configuration
39	PE7/STATUS	Input/Output	I/O Strobe I7 PWM3 RXA/RXE/SCLKC DREQ1 Input Capture	PE7 (SCLKC) is the default configuration

Table 2. RCM4200 Pinout Configurations (continued)

Pin	Pin Name	Default Use	Alternate Use	Notes
40–47	LN[0:7]	Analog Input		A/D converter (RCM4200 only)
40	PD0	Input/Output	I/O Strobe I0 Timer C0 D8 INT0 SCLKD/TCLKF QRD1B	
41	PD1	Input/Output	IA6 I/O Strobe I1 Timer C1 D9 INT1 RXD/RCLKF QRD1A Input Capture	RCM4210 only
42	PD2	Input/Output	I/O Strobe I2 Timer C2 D10 DREQ0 TXF/SCLKC QRD2B	SCLKC (see Section 4.2.1)
43	PD3	Input/Output	IA7 I/O Strobe I3 Timer C3 D11 DREQ1 RXC/RXF QRD2A Input Capture	
44	PD4	Input/Output	I/O Strobe I4 D12 PWM0 TXB/TCLKE	RCM4210 only
45	PD5	Input/Output	IA6 I/O Strobe I5 D13 PWM1 RXB/RCLKE Input Capture	

Table 2. RCM4200 Pinout Configurations (continued)

Pin	Pin Name	Default Use	Alternate Use	Notes
46	PD6	Input/Output	I/O Strobe I6 D14 PWM2 TXA/TXE	Serial Port E (RCM4210 only)
47	PD7	Input/Output	IA7 I/O Strobe I7 D15 PWM3 RXA/RXE Input Capture	
48	CONVERT	Digital Input		A/D converter (RCM4200 only)
49	VREF	Analog reference voltage		1.15 V/2.048 V/2.500 V on-chip ref. voltage (RCM4200 only)
50	GND	Ground		Analog ground

4.1.1 Memory I/O Interface

The Rabbit 4000 address lines (A0–A19) and all the data lines (D0–D7) are routed internally to the onboard flash memory and SRAM chips. I/0 write (/IOWR) and I/0 read (/IORD) are available for interfacing to external devices, and are also used by the RCM4200.

Parallel Port A can also be used as an external I/O data bus to isolate external I/O from the main data bus. Parallel Port B pins PB2–PB7 can also be used as an auxiliary address bus.

When using the auxiliary I/O bus for any reason, you must add the following line at the beginning of your program.

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

Selected pins on Parallel Ports D and E as specified in Table 2 may be used for input capture, quadrature decoder, DMA, and pulse-width modulator purposes.

4.1.2 Other Inputs and Outputs

The PE5–PE7 pins can be brought out to header J2 instead of the STATUS and the two SMODE pins, SMODE0 and SMODE1, as explained in Appendix A.6.

/RESET_IN is normally associated with the programming port, but may be used as an external input to reset the Rabbit 4000 microprocessor and the RCM4200 memory. /RESET_OUT is an output from the reset circuitry that can be used to reset other peripheral devices.

4.2 Serial Communication

The RCM4200 module does not have any serial driver or receiver chips directly on the board. However, a serial interface may be incorporated on the board the RCM4200 is mounted on. For example, the Prototyping Board has an RS-232 transceiver chip.

4.2.1 Serial Ports

There are five serial ports designated as Serial Ports A, B, C, D, and E. All five serial ports can operate in an asynchronous mode up to the baud rate of the system clock divided by 8. 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 Port A is normally used as a programming port, but may be used either as an asynchronous or as a clocked serial port once application development has been completed and the RCM4200 is operating in the Run Mode.

Serial Port B is shared by the RCM4200 module's A/D converter, and is set up as a clocked serial port. Since this serial port is set up for synchronous serial communication on the RCM4200 model, you will lose the A/D converter's functionality if you try to use the serial port in the asynchronous mode. Serial Port B is available without any restrictions on the RCM4210.

Serial Port C is shared with the serial flash, and is set up as a clocked serial port. PE7 is set up to provide the SCLKC output to the serial flash, but PD2 also provides the SCLKC ouput automatically when Serial Port C is used as a clocked serial port. Since this serial port is available for synchronous serial communication on either RCM4200 model, you will lose the serial flash's functionality if you try to use the serial port in the asynchronous mode.

> **NOTE:** Since Serial Port C is shared with the serial flash, exercise care if you attempt to use Serial Port C for other serial communication. Your application will have to manage the sharing negotiations to avoid conflicts when reading or writing to the serial flash.

Serial Port D may also be used as a clocked serial port. Note that PD0 provides the SCLKD ouput automatically when Serial Port D is set up as a clocked serial port.

Serial Port E, which is available only on the RCM4210, can also be configured as an SDLC/HDLC serial port. The IrDA protocol is also supported in SDLC format by Serial Port E. Serial Port E must be configured before it can be used. The sample program **IOCONFIG_SWITCHECHO.C** in the Dynamic C **SAMPLES\RCM4200\SERIAL** folder shows how to configure Serial Port E.

Table 3 summarizes the possible parallel port pins for the serial ports and their clocks.

Table 3. Rabbit 4000 Serial Port and Clock Pins

Serial Port A (programming port)	TXA	PC6, PC7, PD6	Serial Port D	TXD	PC0, PC1
	RXA	PC7, PD7, PE7		RXD	PC1, PD1, PE1
	SCLKA	PB1		SCLKD	PD0, PE0, PE3, PC3
Serial Port B (used by A/D converter on RCM4200)	TXB	PC4, PC5, PD4	Serial Port E (RCM4210 only)	TXE	PD6, PC6, PE6
	RXB	PC5, PD5, PE5		RXE	PD7, PC7, PE7
	SCLKB	PB0		RCLKE	PD5, PC5, PE5
Serial Port C (shared by serial flash)	TXC	PC2, PC3		TCLKE	PD4, PC4, PE4
	RXC	PC3, PD3, PE3			
	SCLKC	PD2, PE2, PE7, PC7			

RCLKE must be selected to be on the same parallel port as TXE.

4.2.1.1 Using the Serial Ports

The receive lines on the RCM4200 serial ports do not have pull-up resistors. If you are using the serial ports without a receiver chip (for example, for RS-422, RS-232, or RS-485 serial communication), the absence of a pull-up resistor on the receive line will likely lead to line breaks being generated since line breaks are normally generated whenever the receive line is pulled low. If you are operating a serial port asynchronously, you can inhibit character assembly during breaks by setting bit 1 in the corresponding Serial Port Extended Register to 1. Should you need line breaks, you will have to either add a pull-up resistor on your motherboard or use a receiver that incorporates the circuits to have the output default to the nonbreak levels.

The Dynamic C **RS232.LIB** library requires you to define the macro **RS232_NOCHARASSYINBRK** to inhibit break-character assembly for all the serial ports.

```
#define RS232_NOCHARASSYINBRK
```

This macro is already defined so that it is the default behavior for the sample programs in the Dynamic C **SAMPLES\RCM4200\SERIAL** folder.

4.2.2 Ethernet Port

Figure 8 shows the pinout for the RJ-45 Ethernet port (J3). Note that some Ethernet connectors are numbered in reverse to the order used here.

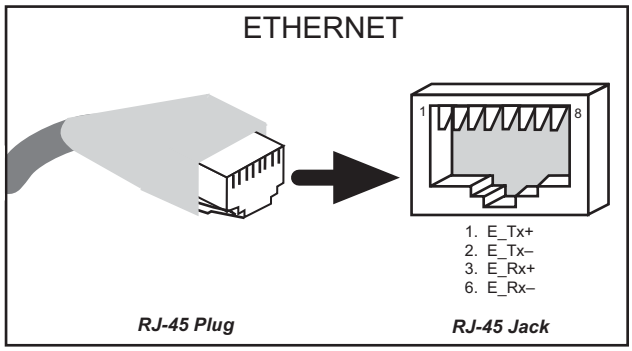


Figure 8. RJ-45 Ethernet Port Pinout

Three LEDs are placed next to the RJ-45 Ethernet jack, one to indicate Ethernet link/activity (**LINK/ACT**), one to indicate when the RCM4200 is connected to a functioning 100Base-T network (**SPEED**), and one (**FDX/COL**) to indicate that the current connection is in full-duplex mode (steady on) or that a half-duplex connection is experiencing collisions (blinks).

The RJ-45 connector is shielded to minimize EMI effects to/from the Ethernet signals.

4.2.3 Programming Port

The RCM4200 is programmed via the 10-pin header labeled J1. The programming port uses the Rabbit 4000's Serial Port A for communication. Dynamic C uses the programming port to download and debug programs.

Serial Port A is also used for the following operations.

- Cold-boot the Rabbit 4000 on the RCM4200 after a reset.
- 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 Programming Port

All three Serial Port A signals are available as

- a synchronous serial port
- an asynchronous serial port, with the clock line usable as a general CMOS I/O pin

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

In addition to Serial Port A, the Rabbit 4000 startup-mode (SMODE0, SMODE1), STATUS, and reset pins are available on the programming port.

The two startup-mode pins determine what happens after a reset—the Rabbit 4000 is either cold-booted or the program begins executing at address 0x0000.

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 once a program has been downloaded and is running.

The reset pin is an external input that is used to reset the Rabbit 4000.

Refer to the ***Rabbit 4000 Microprocessor User's Manual*** for more information.

4.3 Programming Cable

The programming cable is used to connect the programming port of the RCM4200 to a PC serial COM port. The programming cable converts the RS-232 voltage levels used by the PC serial port to the CMOS voltage levels used by the Rabbit 4000.

When the **PROG** connector on the programming cable is connected to the programming port on the RCM4200, programs can be downloaded and debugged over the serial interface.

The **DIAG** connector of the programming cable may be used on header J1 of the RCM4200 with the RCM4200 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 RCM4200 is automatically in Program Mode when the **PROG** connector on the programming cable is attached, and is automatically in Run Mode when no programming cable is attached. When the Rabbit 4000 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 4000 in the Program Mode. When the programming cable's **PROG** connector is not attached, the SMODE pins are pulled low, causing the Rabbit 4000 to operate in the Run Mode.

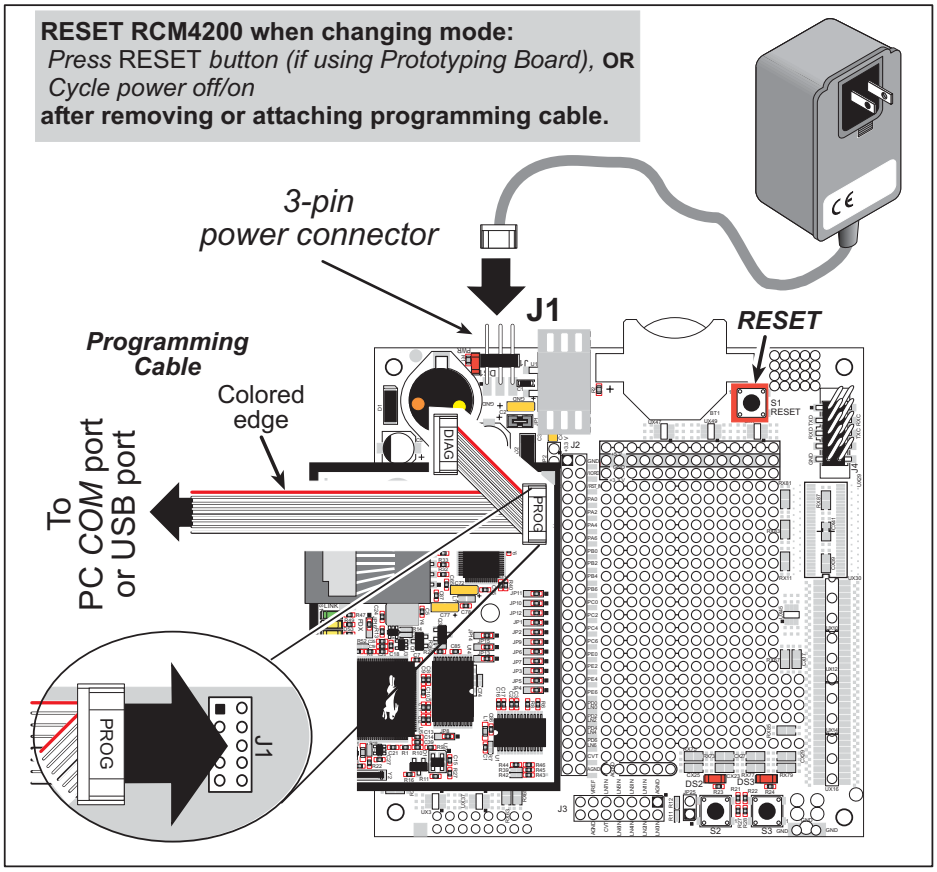


Figure 9. Switching Between Program Mode and Run Mode

A program “runs” in either mode, but can only be downloaded and debugged when the RCM4200 is in the Program Mode.

Refer to the ***Rabbit 4000 Microprocessor User’s Manual*** for more information on the programming port.

4.3.2 Standalone Operation of the RCM4200

Once the RCM4200 has been programmed successfully, remove the programming cable from the programming connector and reset the RCM4200. The RCM4200 may be reset by cycling, the power off/on or by pressing the **RESET** button on the Prototyping Board. The RCM4200 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 RCM4200 module to protect against inadvertent shorts across the pins or damage to the RCM4200 if the pins are not plugged in correctly. Do not reapply power until you have verified that the RCM4200 module is plugged in correctly.

4.4 A/D Converter (RCM4200 only)

The RCM4200 has an onboard ADS7870 A/D converter whose scaling and filtering are done via the motherboard on which the RCM4200 module is mounted. The A/D converter multiplexes converted signals from eight single-ended or four differential inputs to Serial Port B on the Rabbit 4000.

The eight analog input pins, LN0–LN7, each have an input impedance of 6–7 MΩ, depending on whether they are used as single-ended or differential inputs. The input signal can range from -2 V to +2 V (differential mode) or from 0 V to +2 V (single-ended mode).

Use a resistor divider such as the one shown in Figure 10 to measure voltages above 2 V on the analog inputs.

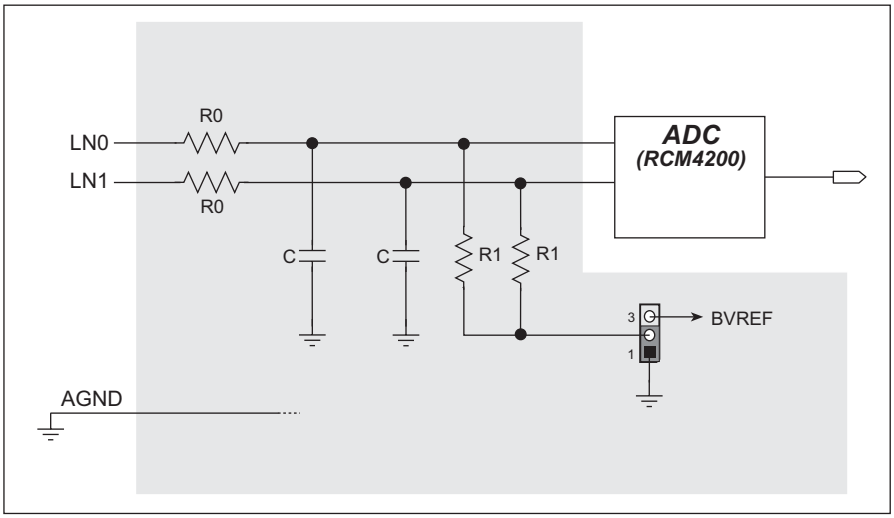


Figure 10. Resistor Divider Network for Analog Inputs

The R1 resistors are typically 20 kΩ to 100 kΩ, with a lower resistance leading to more accuracy, but at the expense of a higher current draw. The R0 resistors would then be 180 kΩ to 900 kΩ for a 10:1 attenuator. The capacitor filters noise pulses on the A/D converter input.

The actual voltage range for a signal going to the A/D converter input is also affected by the 1, 2, 4, 5, 8, 10, 16, and 20 V/V software-programmable gains available on each channel of the ADS7870 A/D converter. Thus, you must scale the analog signal with an attenuator circuit and a software-programmable gain so that the actual input presented to the A/D converter is within the range limits of the ADS7870 A/D converter chip (-2 V to + 2 V or 0 V to + 2 V).

The A/D converter chip can only accept positive voltages. With the R1 resistors connected to ground, your analog circuit is well-suited to perform positive A/D conversions. When the R1 resistors are tied to ground for differential measurements, both differential inputs must be referenced to analog ground, and ***both inputs must be positive with respect to analog ground***.

If a device such as a battery is connected across two channels for a differential measurement, and it is ***not*** referenced to analog ground, then the current from the device will flow through both sets of attenuator resistors without flowing back to analog ground as shown in Figure 11. This will generate a negative voltage at one of the inputs, LN1, which will almost certainly lead to inaccurate A/D conversions. To make such differential measurements, connect the R1 resistors to the A/D converter's internal reference voltage, which is software-configurable for 1.15 V, 2.048 V, or 2.5 V. This internal reference voltage is available on pin 49 of header J2 as VREF, and allows you to convert analog input voltages that are negative with respect to analog ground.

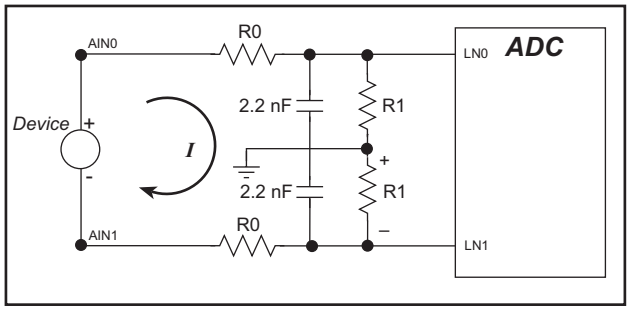


Figure 11. Current Flow from Ungrounded or Floating Source

NOTE: The amplifier inside the A/D converter's internal voltage reference circuit has a very limited output-current capability. The internal buffer can source up to 20 mA and sink only up to 200 µA. Use a separate buffer amplifier if you need to supply any load current.

The A/D converter's CONVERT pin is available on pin 48 of header J2 and can be used as a hardware means of forcing the A/D converter to start a conversion cycle at a specific time. The CONVERT signal is an edge-triggered event and has a hold time of two CCLK periods for debounce.

A conversion is started by an active (rising) edge on the CONVERT pin. The CONVERT pin must stay low for at least two CCLK periods before going high for at least two CCLK periods. Figure 12 shows the timing of a conversion start. The double falling arrow on CCLK indicates the actual start of the conversion cycle.

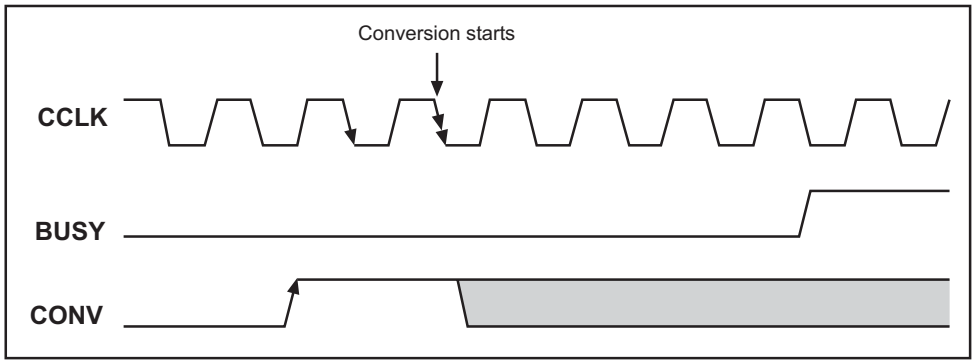


Figure 12. Timing Diagram for Conversion Start Using CONVERT Pin

Appendix B explains the implementation examples of these features on the Prototyping Board.

4.4.1 A/D Converter Power Supply

The analog section is isolated from digital noise generated by other components by way of a low-pass filter composed of C1, L1, and C86 on the RCM4200 as shown in Figure 13. The +V analog power supply powers the A/D converter chip.

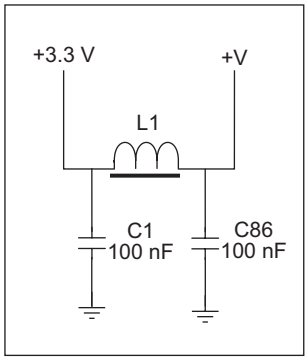


Figure 13. Analog Supply Circuit

4.5 Other Hardware

4.5.1 Clock Doubler

The RCM4200 takes advantage of the Rabbit 4000 microprocessor's internal clock doubler. A built-in clock doubler allows half-frequency crystals to be used to reduce radiated emissions. The 58.98 MHz frequency specified for the RCM4200 model is generated using a 29.49 MHz crystal.

The clock doubler may be disabled if 58.98 MHz clock speeds are not required. Disabling the Rabbit 4000 microprocessor's internal clock doubler 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.
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

The Rabbit 4000 features a spectrum spreader, which helps to mitigate EMI problems. The spectrum spreader is on by default, but it 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 not recommended since it may limit the maximum clock speed or the maximum baud rate. It is unlikely that the strong setting will be used in a real application.
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.

NOTE: Refer to the ***Rabbit 4000 Microprocessor User's Manual*** for more information on the spectrum-spreading setting and the maximum clock speed.

4.6 Memory

4.6.1 SRAM

All RCM4200 modules have 512K of battery-backed data SRAM installed at U10, and the RCM4200 model has 512K of fast SRAM installed at U12.

4.6.2 Flash EPROM

All RCM4200 modules also have 512K of flash EPROM installed at U11.

> **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 discouraged. Instead, define a "user block" area to store persistent data. The functions **`writeUserBlock`** and **`readUserBlock`** are provided for this. Refer to the ***Rabbit 4000 Microprocessor Designer's Handbook*** for additional information.

4.6.3 Serial Flash

Up to 8 Mbytes of serial flash memory is available to store data and Web pages. Sample programs in the **`SAMPLES\RCM4200\Serial_Flash`** folder illustrate the use of the serial flash memory.

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 single-board computers and other devices based on the Rabbit microprocessor. Chapter 5 describes the libraries and function calls related to the RCM4200.

5.1 More About Dynamic C

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

You have a choice of doing your software development in the flash memory or in the static SRAM included on the RCM4200. The flash memory and SRAM options are selected with the **Options > Program 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 compiled directly to the battery-backed data SRAM on the RCM4200 module, but should be run from the fast SRAM after the serial programming cable is disconnected. Your final code must always be stored in flash memory for reliable operation. RCM4200 modules have a fast program execution SRAM that is not battery-backed. Select **Code and BIOS in Flash, Run in RAM** from the Dynamic C **Options > Project Options > Compiler** menu to store the code in flash and copy it to the fast program execution SRAM at run-time to take advantage of the faster clock speed. This option optimizes the performance of RCM4200 modules running at 58.98 MHz.

NOTE: Do not depend on the flash memory sector size or type in your program logic. The RCM4200 and Dynamic C were designed to accommodate flash devices with various sector sizes in response to the volatility of the flash-memory market.

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 and later. Programs can be downloaded at baud rates of up to 460,800 bps after the program compiles.

Dynamic C has a number of standard features.

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

5.2 Dynamic C Function Calls

5.2.1 Digital I/O

The RCM4200 was designed to interface with other systems, and so there are no drivers written specifically for the I/O. The general Dynamic C read and write functions allow you to customize the parallel I/O to meet your specific needs. For example, use

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

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

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

to set all the Port E bits as outputs.

When using the auxiliary I/O bus on the Rabbit 4000 chip, add the line

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

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

The sample programs in the Dynamic C **SAMPLES/RCM4200** folder provide further examples.

5.2.2 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, allowing other functions to be performed between calls. For more information, see the ***Dynamic C Function Reference Manual*** and Technical Note TN213, ***Rabbit Serial Port Software***.

5.2.3 User Block

Certain function calls involve reading and storing calibration constants from/to the simulated EEPROM in flash memory located at the top 2K of the reserved user block memory area (3800–39FF). This leaves the address range 0–37FF in the user block available for your application.

These address ranges may change in the future in response to the volatility in the flash memory market, in particular sector size. The sample program **USERBLOCK_INFO.C** in the Dynamic C **SAMPLES\USERBLOCK** folder can be used to determine the version of the ID block, the size of the ID and user blocks, whether or not the ID/user blocks are mirrored, the total amount of flash memory used by the ID and user blocks, and the area of the user block available for your application.

The **USERBLOCK_CLEAR.C** sample program shows you how to clear and write the contents of the user block that you are using in your application (the calibration constants in the reserved area and the ID block are protected).

5.2.4 SRAM Use

The RCM4200 module has a battery-backed data SRAM and a program-execution SRAM. Dynamic C provides the **protected** keyword to identify variables that are to be placed into the battery-backed SRAM. The compiler generates code that maintains two copies of each protected variable in the battery-backed SRAM. The compiler also generates a flag to indicate which copy of the protected variable is valid at the current time. This flag is also stored in the battery-backed SRAM. When a protected variable is updated, the "inactive" copy is modified, and is made "active" only when the update is 100% complete. This assures the integrity of the data in case a reset or a power failure occurs during the update process. At power-on the application program uses the active copy of the variable pointed to by its associated flag.

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

```
main() {
   protected int state1, state2, state3;
   ...
   _sysIsSoftReset();    // restore any protected variables
```

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

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

5.2.4.1 SRAM Chip Select Considerations

The basic SRAM memory on Rabbit-based boards is always connected to /CS1, /OE1, and /WE1. Both the data SRAM and the program execution fast SRAM on the RCM4200 share /OE1.

The BIOS-defined macro, **CS1_ALWAYS_ON**, is set to 0 by default to disable /CS1 (set it high). The macro may be redefined in the BIOS to 1, which will set a bit in the MMIDR register that forces /CS1 to stay enabled (low). This capability is normally used to speed up access time for battery-backed SRAM as long as no other memory chips are connected to /OE1 and /WE1. Therefore, the **CS1_ALWAYS_ON** macro must remain at its default setting of 0 to avoid conflicts between the data SRAM and the program execution fast SRAM.

5.2.5 RCM4200 Cloning

The RCM4200 does not have a pull-up resistor on the PB1 (CLKA) line of the programming port. Because of this, the procedure to generate clones from the RCM4200 differs from that used for other RabbitCore modules and single-boards computers. You must set the **CL_FORCE_MASTER_MODE** macro to 1 in the Dynamic C **CLONECONFIG.LIB** library to use the RCM4200 as a master for cloning. An RCM4200 master will not run the application, and further debugging is not possible as long as the **CL_FORCE_MASTER_MODE** macro is set to 1. Any cloned RCM4200 modules will be "sterile," meaning that they cannot be used as a master for cloning. To develop and debug an application on an RCM4200, comment out the **CL_FORCE_MASTER_MODE** macro or set it to 0.

> **NOTE:** Instead of defining this macro is your application, you may simply add the line **CL_FORCE_MASTER_MODE=1** under the Dynamic C **Options > Project Options** "Defines" tab, then click **OK**. When you recompile your program, this will have the same effect as setting the macro to 1 within the **CLONECONFIG.LIB** library.

See Technical Note TN207, ***Rabbit Cloning Board***, for additional information on Rabbit Semiconductor's cloning board and how cloning is done.

5.2.6 Serial Flash Drivers

The Dynamic C **LIB\SerialFlash\SFLASH.LIB** and **LIB\SerialFlash\SFLASH_FAT.LIB** libraries provide the function calls needed to use the serial flash. The FAT file system function calls are in the Dynamic C **LIB\FileSystem\FAT_CONFIG.LIB** library.

5.2.7 Prototyping Board Function Calls

The functions described in this section are for use with the Prototyping Board features. The source code is in the Dynamic C **LIB\RCM4xxx\RCM42xx.LIB** library if you need to modify it for your own board design.

> **NOTE:** The analog input function calls are supported only by the RCM4200 model since the RCM4210 does not have an A/D converter.

The sample programs in the Dynamic C **SAMPLES\RCM4200** folder illustrate the use of the function calls.

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

5.2.7.1 Board Initialization

---

brdInit

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

DESCRIPTION

Call this function at the beginning of your program. This function initializes Parallel Ports A through E for use with the Prototyping Board, and on the RCM4200 model loads the stored calibration constants for the A/D converter. This function call is intended for demonstration purposes only, and can be modified for your applications.

Summary of Initialization

1. I/O port pins are configured for Prototyping Board operation.
2. Unused configurable I/O are set as tied outputs.
3. RS-232 is not enabled.
4. LEDs are off.
5. The slave port is disabled.

RETURN VALUE

None.

5.2.7.2 Alerts

These function calls can be found in the Dynamic C **LIB\RCM4xxx\RCM4xxx.LIB** library.

timedAlert

```
void timedAlert(unsigned long timeout);
```

DESCRIPTION

Polls the real-time clock until a timeout occurs. The RCM4200 will be in a low-power mode during this time. Once the timeout occurs, this function call will enable the normal power source.

PARAMETER

timeout	the duration of the timeout in seconds

RETURN VALUE

None.

digInAlert

```
void digInAlert(int dataport, int portbit, int value,
   unsigned long timeout);
```

DESCRIPTION

Polls a digital input for a set value or until a timeout occurs. The RCM4400W will be in a low-power mode during this time. Once a timeout occurs or the correct byte is received, this function call will enable the normal power source and exit.

PARAMETERS

dataport	the input port data register to poll (e.g., PADR)
portbit	the input port bit (0–7) to poll
value	the value of 0 or 1 to receive
timeout	the duration of the timeout in seconds (enter 0 for no timeout)

RETURN VALUE

None.

5.2.8 Analog Inputs (RCM4200 only)

The function calls used with the Prototyping Board features and the A/D converter on the RCM4200 model are in the Dynamic C **LIB\RCM4xxx\ADC_ADS7870.LIB** library. Dynamic C v. 10.07 or later is required to use the A/D converter function calls.

anaInConfig

```
unsigned int anaInConfig(unsigned int instructionbyte,
   unsigned int cmd, long brate);
```

DESCRIPTION

Use this function to configure the A/D converter. This function will address the A/D converter chip in Register Mode only, and will return an error if you try the Direct Mode. Appendix B.4.3 provides additional addressing and command information.

ADS7870 Signal	ADS7870 State	RCM4200 Function/State
LN0	Input	AIN0
LN1	Input	AIN1
LN2	Input	AIN2
LN3	Input	AIN3
LN4	Input	AIN4
LN5	Input	AIN5
LN6	Input	AIN6
LN7	Input	AIN7
/RESET	Input	Board reset device
RISE/FALL	Input	Pulled up for SCLK active on rising edge
I/O0	Input	Pulled down
I/O1	Input	Pulled down
I/O2	Input	Pulled down
I/O3	Input	Pulled down
CONVERT	Input	Pulled down when not driven
BUSY	Output	PE0 pulled down; logic high state converter is busy
CCLKCNTRL	Input	Pulled down; 0 state sets CCLK as input
CCLK	Input	Pulled down; external conversion clock
SCLK	Input	PB0; serial data transfer clock
SDI	Input	PC4; 3-wire mode for serial data input
SDO	Output	PC5; serial data output /CS driven
/CS	Input	BUFEN pulled up; active-low enables serial interface
BUFIN	Input	Driven by VREF
VREF	Output	Connected to BUFIN and BUFOUT
BUFOUT	Output	Driven by VREF

PARAMETERS

instructionbyte the instruction byte that will initiate a read or write operation at 8 or 16 bits on the designated register address. For example,

```
checkid = anaInConfig(0x5F, 0, 9600);
   // read ID and set byte rate
```

cmd the command data that configure the registers addressed by the instruction byte. Enter 0 if you are performing a read operation. For example,

```
i = anaInConfig(0x07, 0x3b, 0);
   // write ref/osc reg and enable
```

brate the serial clock transfer rate of 9600 to 115,200 bytes per second. **brate** must be set the first time this function is called. Enter 0 for this parameter thereafter, for example,

```
anaInConfig(0x00, 0x00, 9600);
   // resets device and sets byte rate
```

RETURN VALUE

0 on write operations

data value on read operations

SEE ALSO

anaInDriver, **anaIn**, **brdInit**

anaInDriver

```
int anaInDriver(unsigned int cmd);
```

DESCRIPTION

Reads the voltage of an analog input channel by serial-clocking an 8-bit command to the A/D converter by its Direct Mode method. This function assumes that Mode1 (most significant byte first) and the A/D converter oscillator have been enabled. See **anaInConfig()** for the setup.

The conversion begins immediately after the last data bit has been transferred. An exception error will occur if Direct Mode bit D7 is not set.

An exception error will occur if Direct Mode bit D7 is not set.

PARAMETERS

cmd contains a gain code and a channel code as follows.

D7—1; D6–D4—Gain Code; D3–D0—Channel Code

Use the following calculation and the tables below to determine **cmd**:

```
cmd = 0x80 | (gain_code*16) + channel_code
```

Gain Code	Gain Multiplier
0	×1
1	×2
2	×4
3	×5
4	×8
5	×10
6	×16
7	×20

Channel Code	Differential Input Lines	Channel Code	Single-Ended Input Lines*	4–20 mA Lines
0	+AIN0 -AIN1	8	AIN0	AIN0*
1	+AIN2 -AIN3	9	AIN1	AIN1*
2	+AIN4 -AIN5	10	AIN2	AIN2*
3†	+AIN6 -AIN7	11	AIN3	AIN3
4	-AIN0 +AIN1	12	AIN4	AIN4
5	-AIN2 +AIN3	13	AIN5	AIN5
6	-AIN4 +AIN5	14	AIN6	AIN6
7‡	-AIN6 +AIN7	15	AIN7	AIN7*

* Negative input is ground.
† Not accessible on Prototyping Board
‡ Not accessible on Prototyping Board

RETURN VALUE

A value corresponding to the voltage on the analog input channel:

0–2047 for 11-bit conversions
-2048–2047 for 12-bit conversions
ADTIMEOUT (-4095) if the conversion is incomplete or busy bit timeout
ADOVERFLOW (-4096) for overflow or out of range

SEE ALSO

anaInConfig, **anaIn**, **brdInit**

anaIn

```
int anaIn(unsigned int channel, int opmode, int gaincode);
```

DESCRIPTION

Reads the value of an analog input channel using the Direct Mode method of addressing the A/D converter. Note that it takes about 1 second to ensure an internal capacitor on the A/D converter is charged when the function is called the first time.

PARAMETERS

channel the channel number (0 to 7) corresponding to LN0 to LN7.

opmode the mode of operation:

SINGLE—single-ended input
DIFF—differential input
mAMP—4–20 mA input

channel	SINGLE	DIFF	mAMP
0	+AIN0	+AIN0 -AIN1	+AIN0*
1	+AIN1	+AIN1 -AIN0*	+AIN1*
2	+AIN2	+AIN2 -AIN3	+AIN2*
3	+AIN3	+AIN3 -AIN2*	+AIN3
4	+AIN4	+AIN4 -AIN5	+AIN4
5	+AIN5	+AIN5 -AIN4*	+AIN5
6	+AIN6	+AIN6 -AIN7*	+AIN6
7	+AIN7	+AIN7 -AIN6*	+AIN7*

* Not accessible on Prototyping Board.

gaincode the gain code of 0 to 7 (applies only to Prototyping Board):

Gain Code	Gain Multiplier	Voltage Range (V)
0	×1	0–22.5
1	×2	0–11.25
2	×4	0–5.6
3	×5	0–4.5
4	×8	0–2.8
5	×10	0–2.25
6	×16	0–1.41
7	×20	0–1.126

RETURN VALUE

A value corresponding to the voltage on the analog input channel:

0–2047 for single-ended conversions
-2048–2047 for differential conversions
ADTIMEOUT (-4095) if the conversion is incomplete or busy bit timeout
ADOVERFLOW (-4096) for overflow or out of range

SEE ALSO

anaIn, **anaInConfig**, **anaInDriver**

anaInCalib

```
int anaInCalib(int channel, int opmode, int gaincode,
   int value1, float volts1, int value2, float volts2);
```

DESCRIPTION

Calibrates the response of the desired A/D converter channel as a linear function using the two conversion points provided. Four values are calculated and placed into global tables **_adcCalibS**, **_adcCalibD**, and **adcCalibM** to be later stored into simulated EEPROM using the function **anaInEEWr()**. Each channel will have a linear constant and a voltage offset.

PARAMETERS

channel the channel number (0 to 7) corresponding to LN0 to LN7.

opmode the mode of operation:

SINGLE—single-ended input
DIFF—differential input
mAMP—4–20 mA input

channel	SINGLE	DIFF	mAMP
0	+AIN0	+AIN0 -AIN1	+AIN0*
1	+AIN1	+AIN1 -AIN0*	+AIN1*
2	+AIN2	+AIN2 -AIN3	+AIN2*
3	+AIN3	+AIN3 -AIN2*	+AIN3
4	+AIN4	+AIN4 -AIN5	+AIN4
5	+AIN5	+AIN5 -AIN4*	+AIN5
6	+AIN6	+AIN6 -AIN7*	+AIN6
7	+AIN7	+AIN7 -AIN6*	+AIN7*

* Not accessible on Prototyping Board.

gaincode the gain code of 0 to 7 (applies only to Prototyping Board):

Gain Code	Gain Multiplier	Voltage Range (V)
0	×1	0–22.5
1	×2	0–11.25
2	×4	0–5.6
3	×5	0–4.5
4	×8	0–2.8
5	×10	0–2.25
6	×16	0–1.41
7	×20	0–1.126

value1 the first A/D converter channel raw count value (0–2047)

volts1 the voltage or current corresponding to the first A/D converter channel value (0 to +20 V or 4 to 20 mA)

value2 the second A/D converter channel raw count value (0–2047)

volts2 the voltage or current corresponding to the first A/D converter channel value (0 to +20 V or 4 to 20 mA)

RETURN VALUE

0 if successful.

-1 if not able to make calibration constants.

SEE ALSO

anaIn, **anaInVolts**, **anaInmAmps**, **anaInDiff**, **anaInCalib**, **brdInit**

anaInVolts

```
float anaInVolts(unsigned int channel, unsigned int gaincode);
```

DESCRIPTION

Reads the state of a single-ended analog input channel and uses the previously set calibration constants to convert it to volts.

PARAMETERS

channel the channel number (0 to 7) corresponding to LN0 to LN7.

Channel Code	Single-Ended Input Lines*	Voltage Range† (V)
0	+AIN0	0–22.5
1	+AIN1	0–22.5
2	+AIN2	0–22.5
3	+AIN3	0–22.5
4	+AIN4	0–22.5
5	+AIN5	0–22.5
6	+AIN6	0–22.5
7	+AIN7	0–2‡

* Negative input is ground.
† Applies to Prototyping Board.
‡ Used for thermistor in sample program.

gaincode the gain code of 0 to 7 (applies only to Prototyping Board):

Gain Code	Gain Multiplier	Voltage Range* (V)
0	×1	0–22.5
1	×2	0–11.25
2	×4	0–5.6
3	×5	0–4.5
4	×8	0–2.8
5	×10	0–2.25
6	×16	0–1.41
7	×20	0–1.126

* Applies to Prototyping Board.

RETURN VALUE

A voltage value corresponding to the voltage on the analog input channel.

ADTIMEOUT (-4095) if the conversion is incomplete or busy bit timeout.

ADOVERFLOW (-4096) for overflow or out of range.

SEE ALSO

anaInCalib, **anaIn**, **anaInmAmps**, **brdInit**

anaInDiff

```
float anaInDiff(unsigned int channel, unsigned int gaincode);
```

DESCRIPTION

Reads the state of differential analog input channels and uses the previously set calibration constants to convert it to volts.

PARAMETERS

channel the channel number (0 to 7) corresponding to LN0 to LN7.

channel	DIFF	Voltage Range (V)
0	+AIN0 -AIN1	-22.5 to +22.5*
1	+AIN1 -AIN1	—
2	+AIN2 -AIN3	-22.5 to +22.5*
3	+AIN3 -AIN3	—
4	+AIN4 -AIN5	-22.5 to +22.5*
5	+AIN5 -AIN5	—
6	+AIN6 -AIN7	—
7	+AIN7 -AIN7	—

* Accessible on Prototyping Board.

gaincode the gain code of 0 to 7 (applies only to Prototyping Board):

Gain Code	Gain Multiplier	Voltage Range (V)
0	×1	-22.5 – +22.5
1	×2	-11.25 – +11.25
2	×4	-5.6 – +5.6
3	×5	-4.5 – +4.5
4	×8	-2.8 – +2.8
5	×10	-2.25 – +2.25
6	×16	-1.41 – +1.41
7	×20	-1.126 – +1.126

RETURN VALUE

A voltage value corresponding to the voltage differential on the analog input channel.

ADTIMEOUT (-4095) if the conversion is incomplete or busy bit timeout.

ADOVERFLOW (-4096) for overflow or out of range.

SEE ALSO

anaInCalib, **anaIn**, **anaInmAmps**, **brdInit**

anaInmAmps

```
float anaInmAmps(unsigned int channel);
```

DESCRIPTION

Reads the state of an analog input channel and uses the previously set calibration constants to convert it to current.

PARAMETERS

channel the channel number (0 to 7) corresponding to LN0 to LN7.

Channel Code	4–20 mA Input Lines*
0	+AIN0
1	+AIN1
2	+AIN2
3	+AIN3†
4	+AIN4*
5	+AIN5*
6	+AIN6*
7	+AIN7

* Negative input is ground.
† Applies to Prototyping Board.

RETURN VALUE

A current value between 4.00 and 20.00 mA corresponding to the current on the analog input channel.

ADTIMEOUT (-4095) if the conversion is incomplete or busy bit timeout.

ADOVERFLOW (-4096) for overflow or out of range.

SEE ALSO

anaInCalib, **anaIn**, **anaInVolts**

anaInEERd

```
root int anaInEERd(unsigned int channel, unsigned int opmode,
   unsigned int gaincode);
```

DESCRIPTION

Reads the calibration constants, gain, and offset for an input based on their designated position in the flash memory, and places them into global tables **_adcCalibS**, **_adcCalibD**, and **_adcCalibM** for analog inputs. Depending on the flash size, the following macros can be used to identify the starting address for these locations.

ADC_CALIB_ADDRS, address start of single-ended analog input channels

ADC_CALIB_ADDRD, address start of differential analog input channels

ADC_CALIB_ADDRM, address start of milliamp analog input channels

NOTE: This function cannot be run in RAM.

PARAMETER

channel the channel number (0 to 7) corresponding to LN0 to LN7.

opmode the mode of operation:

SINGLE—single-ended input
DIFF—differential input
mAMP—4–20 mA input

channel	SINGLE	DIFF	mAMP
0	+AIN0	+AIN0 -AIN1	+AIN0*
1	+AIN1	+AIN1 -AIN0*	+AIN1*
2	+AIN2	+AIN2 -AIN3	+AIN2*
3	+AIN3	+AIN3 -AIN2*	+AIN3
4	+AIN4	+AIN4 -AIN5	+AIN4
5	+AIN5	+AIN5 -AIN4*	+AIN5
6	+AIN6	+AIN6 -AIN7*	+AIN6
7	+AIN7	+AIN7 -AIN6*	+AIN7*
ALLCHAN	read all channels for selected **opmode**		

* Not accessible on Prototyping Board.

gaincode the gain code of 0 to 7. The **gaincode** parameter is ignored when **channel** is **ALLCHAN**.

Gain Code	Gain Multiplier	Voltage Range* (V)
0	×1	0–22.5
1	×2	0–11.25
2	×4	0–5.6
3	×5	0–4.5
4	×8	0–2.8
5	×10	0–2.25
6	×16	0–1.41
7	×20	0–1.126

* Applies to Prototyping Board.

RETURN VALUE

0 if successful.

-1 if address is invalid or out of range.

SEE ALSO

anaInEEWr, **anaInCalib**

anaInEEWr

```
int anaInEEWr(unsigned int channel, int opmode,
   unsigned int gaincode);
```

DESCRIPTION

Writes the calibration constants, gain, and offset for an input based from global tables **_adcCalibS**, **_adcCalibD**, and **_adcCalibM** to designated positions in the flash memory. Depending on the flash size, the following macros can be used to identify the starting address for these locations.

ADC_CALIB_ADDRS, address start of single-ended analog input channels

ADC_CALIB_ADDRD, address start of differential analog input channels

ADC_CALIB_ADDRM, address start of milliamp analog input channels

NOTE: This function cannot be run in RAM.

PARAMETER

channel the channel number (0 to 7) corresponding to LN0 to LN7.

opmode the mode of operation:

SINGLE—single-ended input
DIFF—differential input
mAMP—4–20 mA input

channel	SINGLE	DIFF	mAMP
0	+AIN0	+AIN0 -AIN1	+AIN0*
1	+AIN1	+AIN1 -AIN0*	+AIN1*
2	+AIN2	+AIN2 -AIN3	+AIN2*
3	+AIN3	+AIN3 -AIN2*	+AIN3
4	+AIN4	+AIN4 -AIN5	+AIN4
5	+AIN5	+AIN5 -AIN4*	+AIN5
6	+AIN6	+AIN6 -AIN7*	+AIN6
7	+AIN7	+AIN7 -AIN6*	+AIN7*
ALLCHAN	read all channels for selected **opmode**		

* Not accessible on Prototyping Board.

gaincode the gain code of 0 to 7. The **gaincode** parameter is ignored when **channel** is **ALLCHAN**.

Gain Code	Gain Multiplier	Voltage Range* (V)
0	×1	0–22.5
1	×2	0–11.25
2	×4	0–5.6
3	×5	0–4.5
4	×8	0–2.8
5	×10	0–2.25
6	×16	0–1.41
7	×20	0–1.126

* Applies to Prototyping Board.

RETURN VALUE

0 if successful

-1 if address is invalid or out of range.

SEE ALSO

anaInEEWr, **anaInCalib**

5.3 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.

5.3.1 Add-On Modules

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. Rabbit Semiconductor offers for purchase add-on Dynamic C modules including the popular µC/OS-II real-time operating system, as well as PPP, Advanced Encryption Standard (AES), FAT file system, Rabbit-Web, and other select libraries.

> **NOTE:** Version 2.10 or later of the Dynamic C FAT file system module is required for the RCM4200 modules.

Each Dynamic C add-on module has complete documentation and sample programs to illustrate the functionality of the software calls in the module. Visit our Web site at www.rabbit.com for further information and complete documentation for each module.

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 RCM4200 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 RCM4200 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.

Figure 14 shows how to identify the two Ethernet cables based on the wires in the transparent RJ-45 connectors.

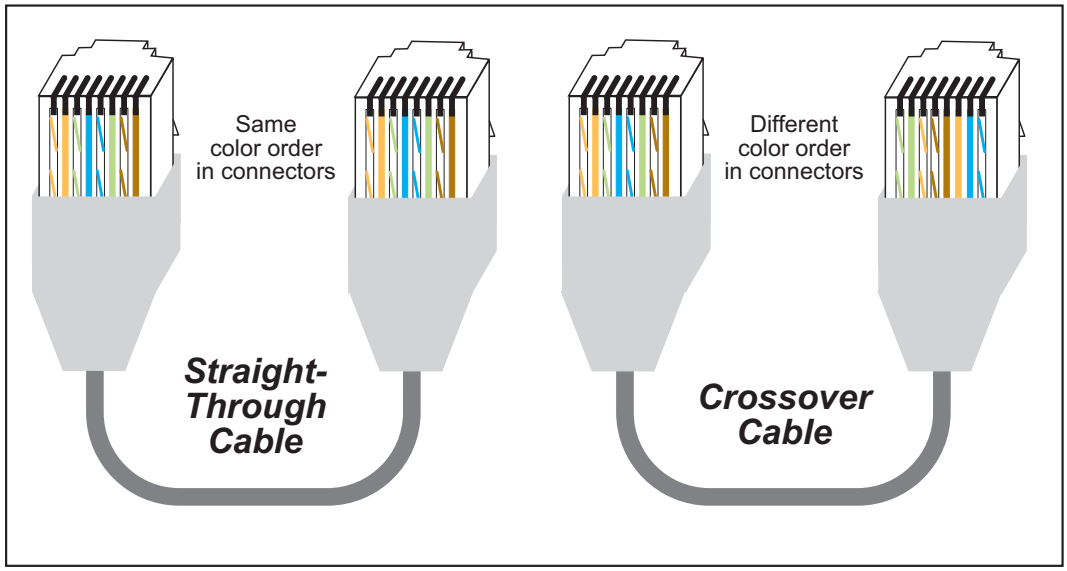


Figure 14. How to Identify Straight-Through and Crossover Ethernet Cables

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.

Now you should be able to make your connections.

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

   There are four options for connecting the RCM4200 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 RCM4200 module's Ethernet port directly to the PC's network interface card using an RJ-45 *crossover cable*. A crossover cable is a special cable that flips some connections between the two connectors and permits direct connection of two client systems. A standard RJ-45 network cable will not work for this purpose.
- **Micro-LAN —** Another simple alternative for desktop development. Use a small Ethernet 10Base-T hub and connect both the PC's network interface card and the RCM4200 module's Ethernet port to it using standard network cables.

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

- **LAN —** Connect the RCM4200 module's Ethernet port to an existing LAN, preferably one to which the development PC is already connected. You will need to obtain IP addressing information from your network administrator.
- **WAN —** The RCM4200 is capable of direct connection to the Internet and other Wide Area Networks, but exceptional care should be used with IP address settings and all network-related programming and development. We recommend that development and debugging be done on a local network before connecting a RabbitCore system to the Internet.

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

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

3. Apply Power

   Plug in the AC adapter. The RCM4200 module and Prototyping Board are 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 RCM4200 using an Ethernet crossover cable or a simple arrangement with a hub. (A crossover cable should not be confused with regular straight through cables.)

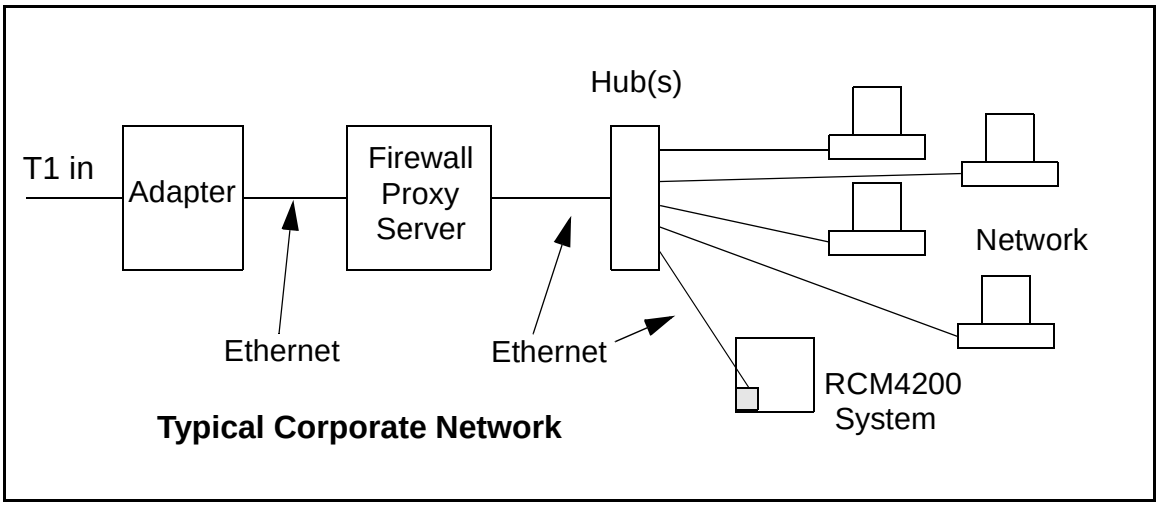
In order to set up this direct connection, the user will have to use a PC without networking, or disconnect a PC from the corporate network, or install a second Ethernet adapter and set up a separate private network attached to the second Ethernet adapter. Disconnecting your PC from the corporate network 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 RCM4200 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 RCM4200 uses 10 Mbps Ethernet, so the hub or Ethernet adapter can be a 10 Mbps unit or a 10/100 Mbps unit.

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 RCM4200 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 setup a direct connection between your computer and the RCM4200. 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.

6.2.1 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 (216.102.126.0) is used to identify the local network. The highest address (216.102.126.255) 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.

6.2.2 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 Ethernet 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.

Some IP address ranges are reserved for use on internal networks, and can be allocated freely as long as no two internal hosts have the same IP address. These internal IP addresses are not routed to the Internet, and any internal hosts using one of these reserved IP addresses cannot communicate on the external Internet without being connected to a host that has a valid Internet IP address. The host would either translate the data, or it would act as a proxy.

Each RCM4200 RabbitCore module has its own unique MAC address, which consists of the prefix 0090C2 followed by a code that is unique to each RCM4200 module. For example, a MAC address might be 0090C2C002C0.

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

6.2.3 Dynamically Assigned Internet Addresses

In many instances, devices on a network do not have 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 RCM4200 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 addresses can be used to perform some Internet tasks such as sending e-mail or browsing the Web, but it is more difficult to participate in conversations that originate elsewhere on the Internet. If you want to find out this dynamically assigned IP address, under Windows NT or later you can run the **ipconfig** command (**Start > Run >cmd**) while you are connected and look at the interface used to connect to the Internet.

Many networks use 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.

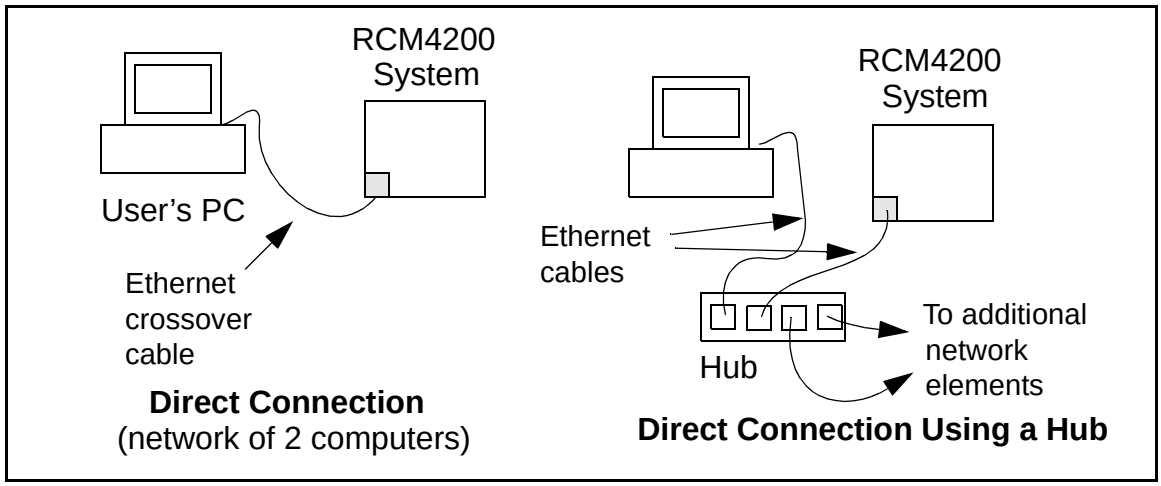
If you are not concerned about accessing the RCM4200 from the Internet, you can place the RCM4200 on the internal network using an IP address assigned either statically or through DHCP.

6.3 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 RCM4200, you have several options. You can either place the RCM4200 directly on the Internet with a real Internet address or place it behind the firewall. If you place the RCM4200 behind the firewall, you need to configure the firewall to translate and forward packets from the Internet to the RCM4200.

6.4 Running TCP/IP Sample Programs

We have provided a number of sample programs demonstrating various uses of TCP/IP for networking embedded systems. These programs require you to connect your PC and the RCM4200 module 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.



6.4.1 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. 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 RCM4200 module, and **10.1.1.1** for your PC, you can edit the values in the section that directly follows the "General Configuration" comment in the **TCP_CONFIG.LIB** library. You will find this library in the **LIB\TCPIP** directory.
3. You can create a **CUSTOM_CONFIG.LIB** library and use a **TCPCONFIG** value greater than 100. Instructions for doing this are at the beginning of the **TCP_CONFIG.LIB** library in the **LIB\TCPIP** directory.

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

6.4.2 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.

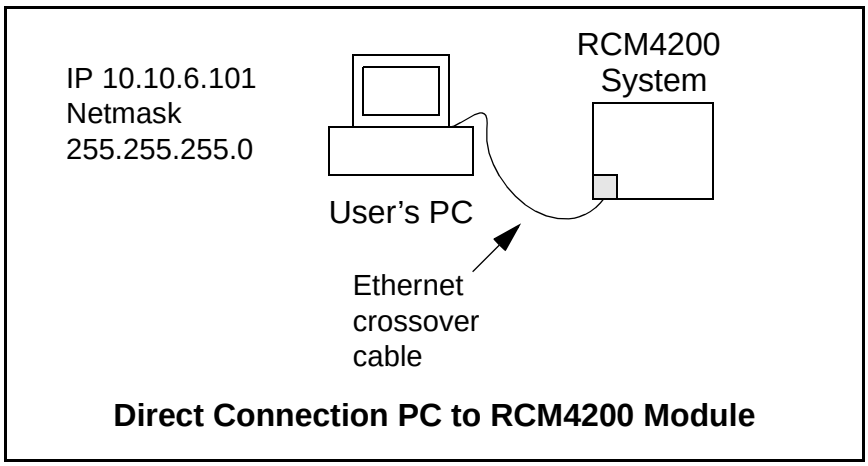
3. Now select the **IP Address** tab, and check **Specify an IP Address**, or select TCP/IP and click on "Properties" to assign an IP address to your computer (this will disable "obtain an IP address automatically"):

   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 RCM4200 Module

6.5 Run the PINGME.C Sample Program

Connect the crossover cable from your computer's Ethernet port to the RCM4200 module'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. The crossover cable is connected from your computer's Ethernet adapter to the RCM4200 module's RJ-45 Ethernet connector. When the program starts running, the green **LINK** light on the RCM4200 module should be on to indicate an Ethernet connection is made. (Note: If the **LINK** light does not light, you may not be using a crossover cable, or if you are using a hub with straight-through cables perhaps the power is off on the hub.)

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

```
ping 10.10.6.101
```

or by **Start > Run**

and typing the entry

```
ping 10.10.6.101
```

The ping routine will ping the module four times and write a summary message on the screen describing the operation.

6.6 Running Additional Sample Programs With Direct Connect

The following sample programs are in the Dynamic C **SAMPLES\RCM4200\TCPIP** folder.

- **BROWSELED.C**—This program demonstrates a basic controller running a Web page. Two "device LEDs" are created along with two buttons to toggle them. Users can use their Web browser to change the status of the lights. The DS2 and DS3 LEDs on the Prototyping Board will match those on the Web page. As long as you have not modified the **TCPCONFIG 1** macro in the sample program, enter the following server address in your Web browser to bring up the Web page served by the sample program.

  http://10.10.6.100.

  Otherwise use the TCP/IP settings you entered in the **TCP_CONFIG.LIB** library.

- **PINGLED.C**—This program demonstrates ICMP by pinging a remote host. It will flash LEDs DS2 and DS3 on the Prototyping Board when a ping is sent and received.

- **SMTP.C**—This program demonstrates using the SMTP library to send an e-mail when the S2 and S3 switches on the Prototyping Board are pressed. LEDs DS2 and DS3 on the Prototyping Board will light up when e-mail is being sent.

6.7 Where Do I Go From Here?

NOTE: If you purchased your RCM4200 through a distributor or through a Rabbit partner, contact the distributor or partner first for technical support.

If there are any problems at this point:

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

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***.

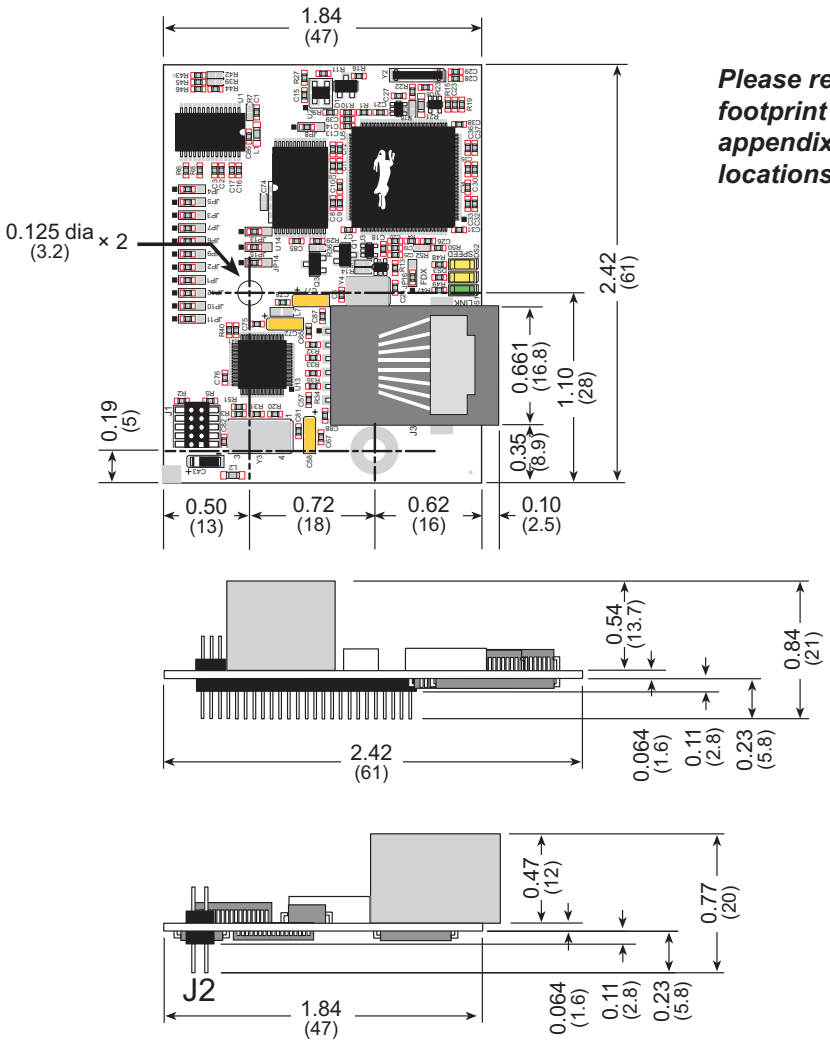
Please refer to the ***Dynamic C TCP/IP User's Manual*** to develop your own applications. ***An Introduction to TCP/IP*** provides background information on TCP/IP, and is available on the CD and on our Web site.

APPENDIX A. RCM4200 SPECIFICATIONS

Appendix A provides the specifications for the RCM4200, and describes the conformal coating.

A.1 Electrical and Mechanical Characteristics

Figure A-1 shows the mechanical dimensions for the RCM4200.



Please refer to the RCM4200 footprint diagram later in this appendix for precise header locations.

Figure A-1. RCM4200 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 RCM4200 in all directions when the RCM4200 is incorporated into an assembly that includes other printed circuit boards. An “exclusion zone” of 0.08" (2 mm) is recommended below the RCM4200 when the RCM4200 is plugged into another assembly. Figure A-2 shows this “exclusion zone.”

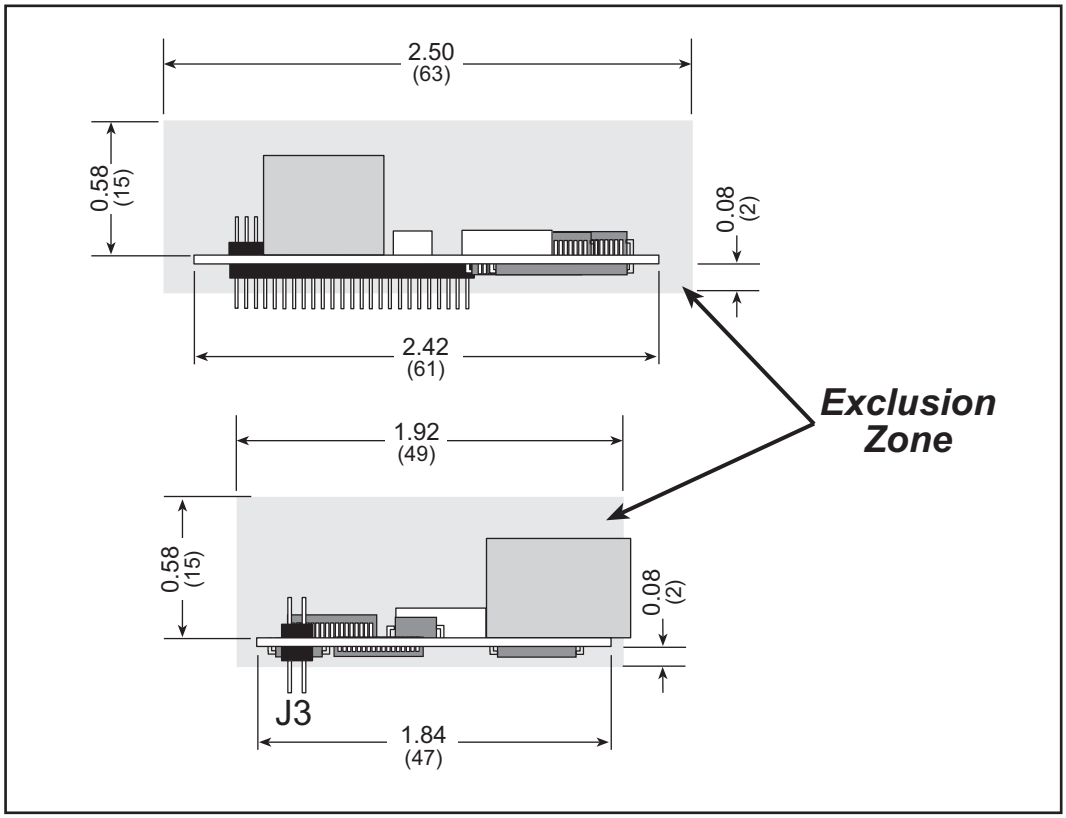


Figure A-2. RCM4200 “Exclusion Zone”

Table A-1 lists the electrical, mechanical, and environmental specifications for the RCM4200.

Table A-1. RCM4200 Specifications

Parameter	RCM4200	RCM4210
Microprocessor	Rabbit® 4000 at 58.98 MHz	Rabbit® 4000 at 29.49 MHz
EMI Reduction	Spectrum spreader for reduced EMI (radiated emissions)	
Ethernet Port	10/100Base-T, RJ-45, 3 LEDs	
Data SRAM	512K (8-bit)	
Program Execution Fast SRAM	512K (8-bit)	—
Flash Memory	512K (8-bit)	
Serial Flash Memory	8 Mbytes	4 Mbytes
Backup Battery	Connection for user-supplied backup battery (to support RTC and data SRAM)	
General Purpose I/O	25 parallel digital I/O lines: • configurable with four layers of alternate functions	35 parallel digital I/O lines: • configurable with four layers of alternate functions
Additional Inputs	2 startup mode, reset in, CONVERT	2 startup mode, reset in
Additional Outputs	Status, reset out, analog VREF	Status, reset out
Analog Inputs	8 channels single-ended or 4 channels differential Programmable gain 1, 2, 4, 5, 8, 10, 16, and 20 V/V	—
• A/D Converter Resolution	12 bits (11 bits single-ended)	
• A/D Conversion Time (including 120 µs raw	180 µs	
Auxiliary I/O Bus	Can be configured for 8 data lines and 6 address lines (shared with parallel I/O lines), plus I/O read/write	
Serial Ports	4 shared high-speed, CMOS-compatible ports: • all 4 configurable as asynchronous (with IrDA), 4 as clocked serial (SPI) • 1 asynchronous clocked serial port shared with programming port • 1 clocked serial port shared with serial flash • 1 clocked serial port shared with A/D converter	5 shared high-speed, CMOS-compatible ports: • all 5 configurable as asynchronous (with IrDA), 4 as clocked serial (SPI), and 1 as SDLC/HDLC • 1 clocked serial port shared with serial flash • 1 asynchronous clocked serial port dedicated for programming

Table A-1. RCM4200 Specifications (continued)

Parameter	RCM4200	RCM4210
Serial Rate	Maximum asynchronous baud rate = CLK/8	
Slave Interface	Slave port allows the RCM4200 to be used as an intelligent peripheral device slaved to a master processor	
Real Time Clock	Yes	
Timers	Ten 8-bit timers (6 cascadable from the first), one 10-bit timer with 2 match registers, and one 16-bit timer with 4 outputs and 8 set/reset registers	
Watchdog/Supervisor	Yes	
Pulse-Width Modulators	• 3 channels synchronized PWM with 10-bit counter • 3 channels variable-phase or synchronized PWM with 16-bit counter	• 4 channels synchronized PWM with 10-bit counter • 4 channels variable-phase or synchronized PWM with 16-bit counter
Input Capture	2 input capture channels can be used to time input signals from various port pins	
Quadrature Decoder	1 quadrature decoder channel accepts inputs from external incremental encoder modules	2 quadrature decoder channels accept inputs from external incremental encoder modules
Power (pins unloaded)	3.0–3.6 V DC, 240 mA (typ.) @ 3.3 V, 275 mA @ 3.6 V and 85°C (max.)	3.0–3.6 V DC, 200 (typ.) mA @ 3.3 V, 225 mA @ 3.6 V and 85°C (max.)
Operating Temperature	-40°C to +85°C	
Humidity	5% to 95%, noncondensing	
Connectors	One 2 × 25, 1.27 mm pitch IDC signal header One 2 × 5, 1.27 mm pitch IDC programming header	
Board Size	1.84" × 2.42" × 0.84" (47 mm × 61 mm × 21 mm)	

A.1.1 A/D Converter

Table A-2 shows some of the important A/D converter specifications. For more details, refer to the ADC7870 data sheet.

Table A-2. A/D Converter Specifications

Parameter	Test Conditions	Typ	Max
Analog Input Characteristics			
Input Capacitance		4 – 9.7 pF	
Input Impedance			
Common-Mode		6 MΩ	
Differential Mode		7 MΩ	
Static Accuracy			
Resolution			
Single-Ended Mode		11 bits	
Differential Mode		12 bits	
Integral Linearity		±1 LSB	±2.5 LSB
Differential Linearity		±0.5 LSB	
Dynamic Characteristics			
Throughput Rate		52 ksamples/s	
Voltage Reference			
Accuracy	V_{ref} = 2.048 V and 2.5 V	±0.05%	±0.25%
Buffer Amp Source Current		20 mA	
Buffer Amp Sink Current		200 µA	
Short-Circuit Current		20 mA	

A.1.2 Headers

The RCM4200 uses a header at J2 for physical connection to other boards. J2 is a 2 × 25 SMT header with a 1.27 mm pin spacing. J1, the programming port, is a 2 × 5 header with a 1.27 mm pin spacing.

Figure A-3 shows the layout of another board for the RCM4200 to be plugged into. These reference design values are relative to one of the mounting holes.

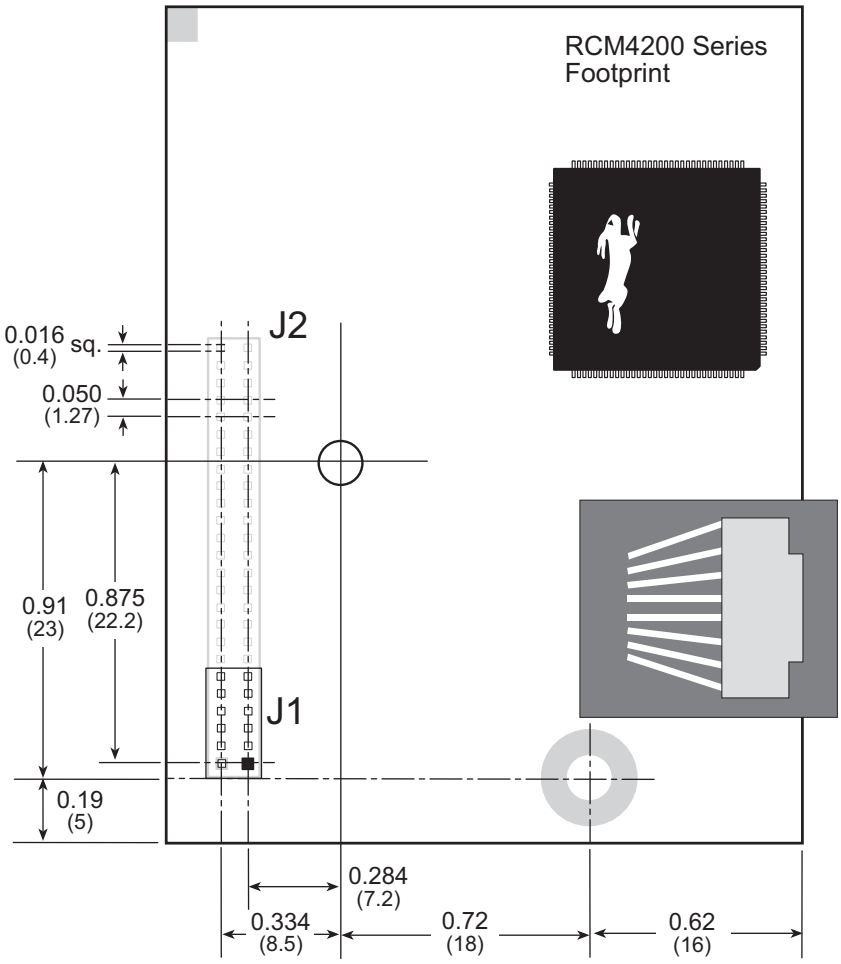


Figure A-3. User Board Footprint for RCM4200

A.2 Rabbit 4000 DC Characteristics

Table A-3. Rabbit 4000 Absolute Maximum Ratings

Symbol	Parameter	Maximum Rating
T_A	Operating Temperature	-40° to +85°C
T_S	Storage Temperature	-55° to +125°C
V_{IH}	Maximum Input Voltage	VDD_{IO} + 0.3 V (max. 3.6 V)
VDD_{IO}	Maximum Operating Voltage	3.6 V

Stresses beyond those listed in Table A-3 may cause permanent damage. The ratings are stress ratings only, and functional operation of the Rabbit 4000 chip at these or any other conditions beyond those indicated in this section is not implied. Exposure to the absolute maximum rating conditions for extended periods may affect the reliability of the Rabbit 4000 chip.

Table A-4 outlines the DC characteristics for the Rabbit 4000 at 3.3 V over the recommended operating temperature range from T_A = –40°C to +85°C, VDD_{IO} = 3.0 V to 3.6 V.

Table A-4. 3.3 Volt DC Characteristics

Symbol	Parameter	Min	Typ	Max
VDD_{IO}	I/O Ring Supply Voltage, 3.3 V	3.0 V	3.3 V	3.6 V
	I/O Ring Supply Voltage, 1.8 V	1.65 V	1.8 V	1.90 V
V_{IH}	High-Level Input Voltage (VDD_{IO} = 3.3 V)		2.0 V	
V_{IL}	Low-Level Input Voltage (VDD_{IO} = 3.3 V)		0.8 V	
V_{OH}	High-Level Output Voltage (VDD_{IO} = 3.3 V)		2.4 V	
V_{OL}	Low-Level Output Voltage (VDD_{IO} = 3.3 V)		0.4 V	
I_{IO}	I/O Ring Current @ 29.4912 MHz, 3.3 V, 25°C			12.2 mA
I_{DRIVE}	All other I/O (except TXD+, TXDD+, TXD-, TXDD-)			8 mA

A.3 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 29.4 MHz CPU clock with the clock doubler enabled and capacitive loading on address and data lines of less than 70 pF per pin. The absolute maximum operating voltage on all I/O is 3.6 V.

A.4 Bus Loading

You must pay careful attention to bus loading when designing an interface to the RCM4200. This section provides bus loading information for external devices.

Table A-5 lists the capacitance for the various RCM4200 I/O ports.

Table A-5. Capacitance of Rabbit 4000 I/O Ports

I/O Ports	Input Capacitance (pF)	Output Capacitance (pF)
Parallel Ports A to E	12	14

Table A-6 lists the external capacitive bus loading for the various RCM4200 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-6.

Table A-6. External Capacitive Bus Loading -40°C to +85°C

Output Port	Clock Speed (MHz)	Maximum External Capacitive Loading (pF)
All I/O lines with clock doubler enabled	58.98	100

Table A-7 lists the loadings for the A/D converter inputs.

Table A-7. A/D Converter Inputs

Parameter	Value
Input Capacitance	4–9.7 pF
Input Impedance	Common-Mode 6 MΩ Differential 7 MΩ

Figure A-4 shows a typical timing diagram for the Rabbit 4000 microprocessor external I/O read and write cycles.

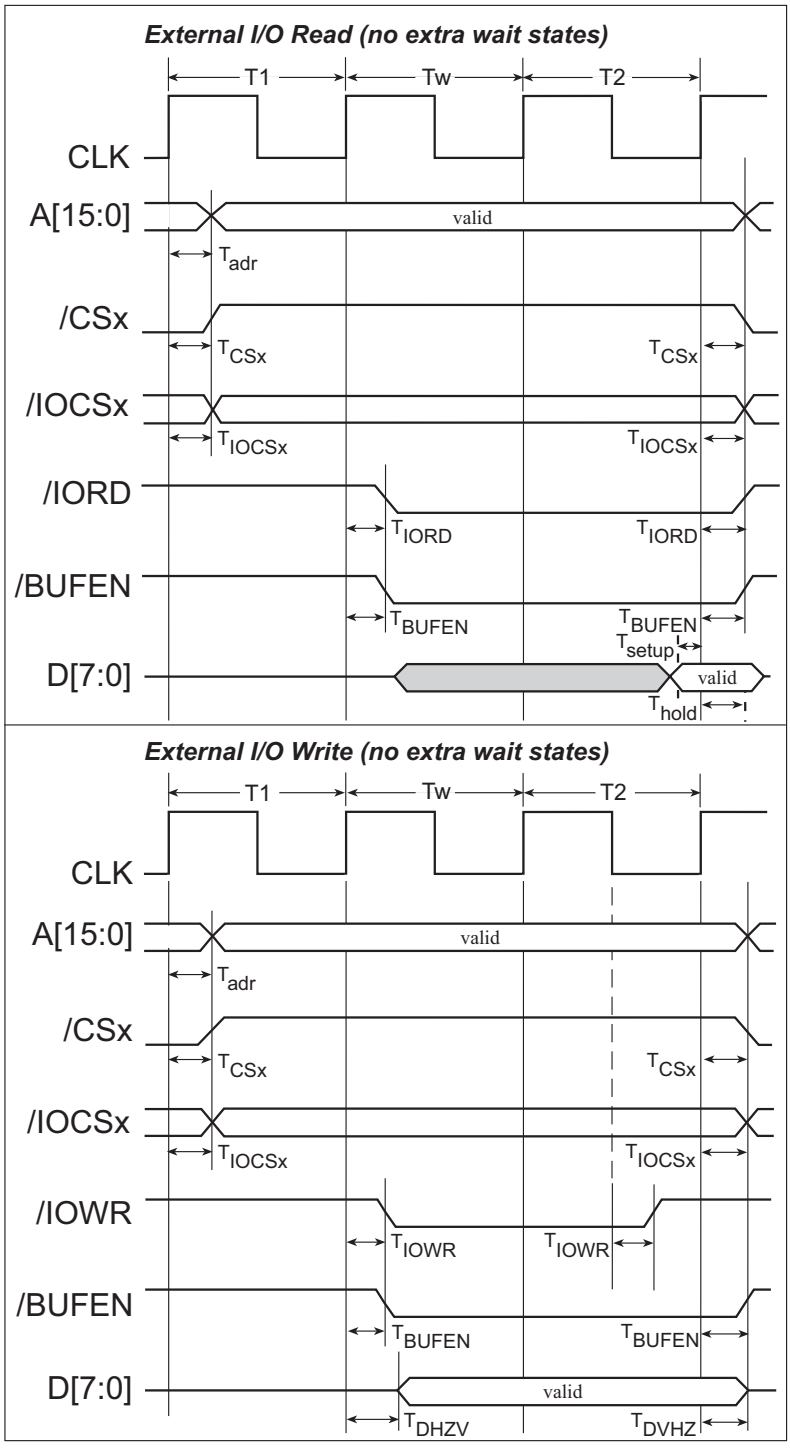


Figure A-4. External I/O Read and Write Cycles—No Extra Wait States

NOTE: **/IOCSx** can be programmed to be active low (default) or active high.

Table A-8 lists the delays in gross memory access time for several values of VDD_{IO}.

Table A-8. Preliminary Data and Clock Delays

VDD_{IO} (V)	Clock to Address Output Delay (ns)			Data Setup Time Delay (ns)	Worst-Case Spectrum Spreader Delay (ns)		
	30 pF	60 pF	90 pF		0.5 ns setting no dbl / dbl	1 ns setting no dbl / dbl	2 ns setting no dbl / dbl
3.3	6	8	11	1	2.3 / 2.3	3 / 4.5	4.5 / 9
1.8	18	24	33	3	7 / 6.5	8 / 12	11 / 22

The measurements are taken at the 50% points under the following conditions.

- T = -40°C to 85°C, V = VDD_{IO} ±10%
- Internal clock to nonloaded CLK pin delay ≤ 1 ns @ 85°C/3.0 V

The clock to address output delays are similar, and apply to the following delays.

- T_{adr}, the clock to address delay
- T_{CSx}, the clock to memory chip select delay
- T_{IOCSx}, the clock to I/O chip select delay
- T_{IORD}, the clock to I/O read strobe delay
- T_{IOWR}, the clock to I/O write strobe delay
- T_{BUFEN}, the clock to I/O buffer enable delay

The data setup time delays are similar for both T_{setup} and T_{hold}.

When the spectrum spreader is enabled with the clock doubler, every other clock cycle is shortened (sometimes lengthened) by a maximum amount given in the table above. The shortening takes place by shortening the high part of the clock. If the doubler is not enabled, then every clock is shortened during the low part of the clock period. The maximum shortening for a pair of clocks combined is shown in the table.

Technical Note TN227, ***Interfacing External I/O with Rabbit Microprocessor Designs***, contains suggestions for interfacing I/O devices to the Rabbit 4000 microprocessors.

A.5 Conformal Coating

The areas around the 32 kHz real-time clock crystal oscillator have had the Dow Corning silicone-based 1-2620 conformal coating applied. The conformally coated area is shown in Figure A-5. The conformal coating protects these high-impedance circuits from the effects of moisture and contaminants over time.

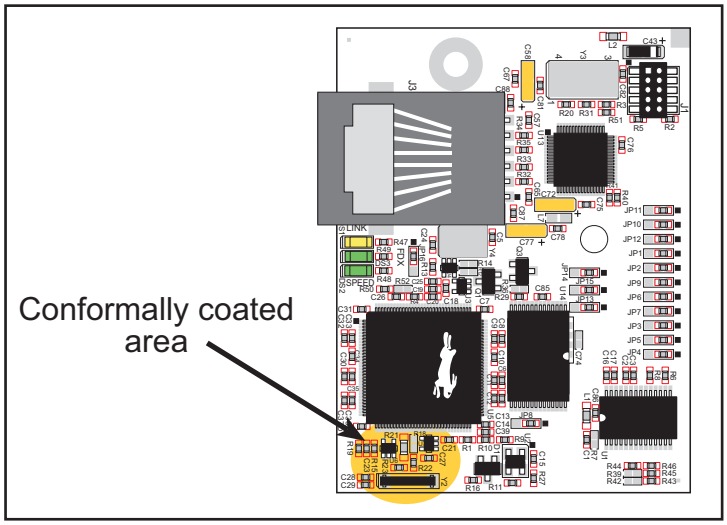


Figure A-5. RCM4200 Areas Receiving Conformal Coating

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

NOTE: For more information on conformal coatings, refer to Technical Note 303, ***Conformal Coatings***.

A.6 Jumper Configurations

Figure A-6 shows the header locations used to configure the various RCM4200 options via jumpers.

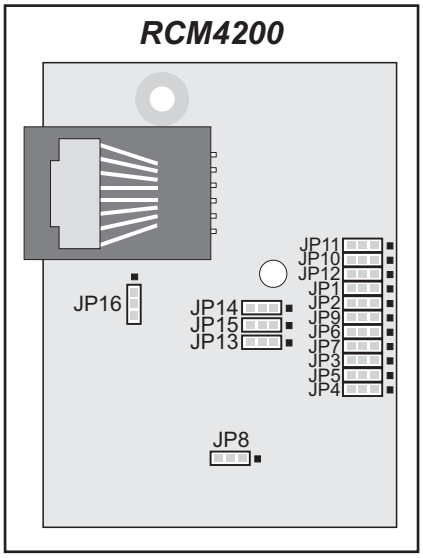


Figure A-6. Location of RCM4200 Configurable Positions

Table A-9 lists the configuration options.

Table A-9. RCM4200 Jumper Configurations

Header	Description	Pins Connected		Factory Default
JP1	LN0 or PD0 on J2 pin 40	1–2	LN0	RCM4200
		2–3	PD0	RCM4210
JP2	LN2 or PD2 on J2 pin 42	1–2	LN2	RCM4200
		2–3	PD2	RCM4210
JP3	LN6 or PD6 on J2 pin 46	1–2	LN6	RCM4200
		2–3	PD6	RCM4210
JP4	LN7 or PD7 on J2 pin 47	1–2	LN7	RCM4200
		2–3	PD7	RCM4210
JP5	LN5 or PD5 on J2 pin 45	1–2	LN5	RCM4200
		2–3	PD5	RCM4210
JP6	LN4 or PD4 on J2 pin 44	1–2	LN4	RCM4200
		2–3	PD4	RCM4210

Table A-9. RCM4200 Jumper Configurations (continued)

Header	Description	Pins Connected		Factory Default
JP7	LN3 or PD3 on J2 pin 43	1–2	LN3	RCM4200
		2–3	PD3	RCM4210
JP8	Data SRAM Size	1–2	512K	✕
		2–3	256K	
JP9	LN1 or PD1 on J2 pin 41	1–2	LN1	RCM4200
		2–3	PD1	RCM4210
JP10	PE5 or SMODE0 Output on J2 pin 37	1–2	PE5	✕
		2–3	SMODE0	
JP11	PE6 or SMODE1 Output on J2 pin 38	1–2	PE6	✕
		2–3	SMODE1	
JP12	PE7 or STATUS Output on J2 pin 39	1–2	PE7	✕
		2–3	STATUS	
JP13	Clocked Synchronous or Programmed I/O Access to Serial Flash	1–2	RxC to Serial Flash	✕
		2–3	Programmed I/O to Serial Flash	
JP14	Clocked Synchronous or Programmed I/O Access to Serial Flash	1–2	TxC to Serial Flash	✕
		2–3	Programmed I/O to Serial Flash	
JP15	Clocked Synchronous or Programmed I/O Access to Serial Flash	1–2	SCLKC to Serial Flash	✕
		2–3	Programmed I/O to Serial Flash	
JP16	LED DS3 Display	1–2	FDX/COL displayed by LED DS3	✕
		2–3	optional ACT displayed by LED DS3	

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

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 RCM4200 and to build prototypes of your own circuits. The Prototyping Board has power-supply connections and also provides some basic I/O peripherals (RS-232, LEDs, and switches), as well as a prototyping area for more advanced hardware development.

B.1 Introduction

The Prototyping Board included in the Development Kit makes it easy to connect an RCM4200 module to a power supply and a PC workstation for development. It also provides some basic I/O peripherals (RS-232, LEDs, and switches), 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 the RCM4200 module.

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

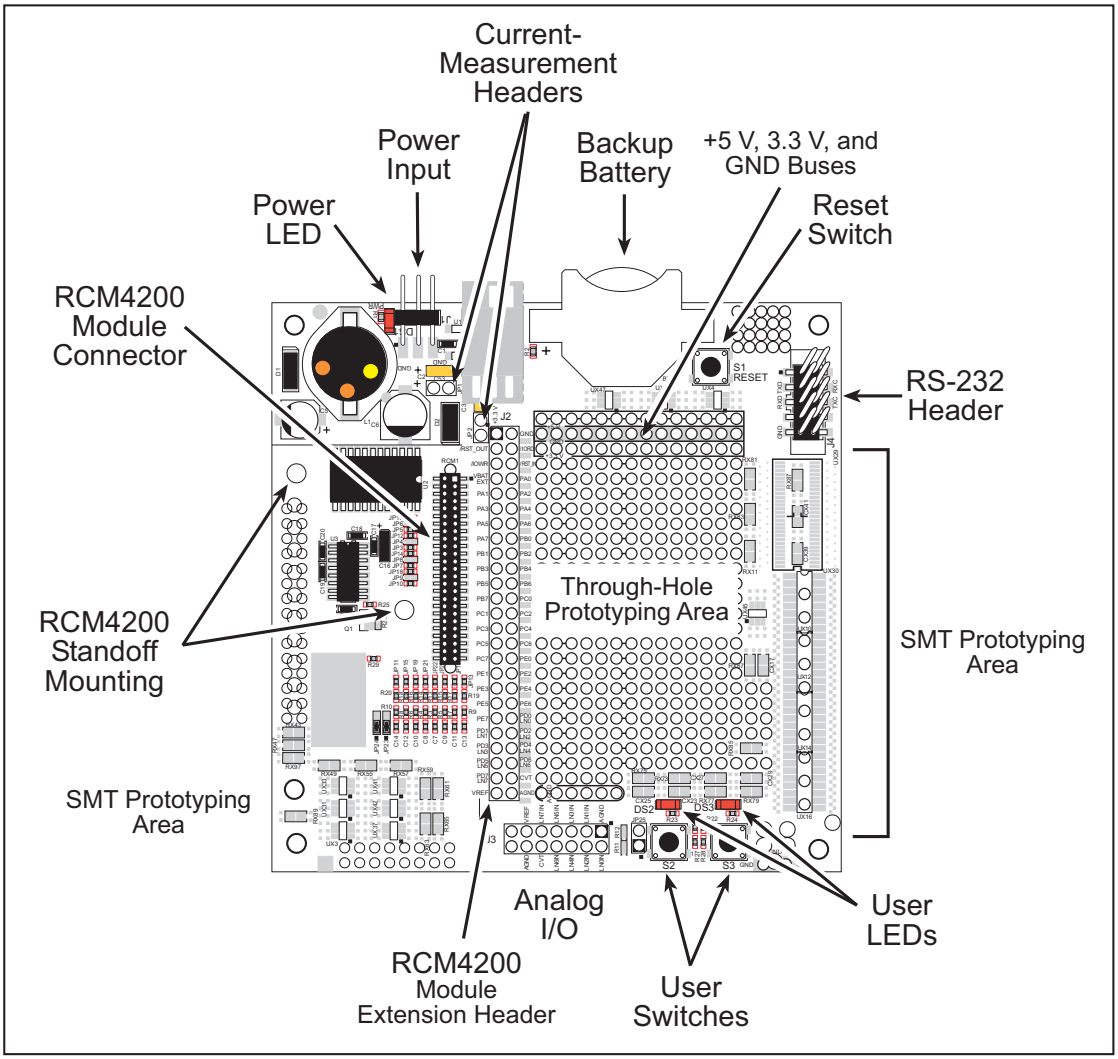


Figure B-1. Prototyping Board

B.1.1 Prototyping Board Features

- **Power Connection**—A a 3-pin header is provided for connection to the power supply. Note that the 3-pin header is symmetrical, with both outer pins connected to ground and the center pin connected to the raw V+ input. The cable of the AC adapter provided with the North American version of the Development Kit is terminated with a header plug that connects to the 3-pin header in either orientation. The header plug leading to bare leads provided for overseas customers can be connected to the 3-pin header in either orientation.

  Users providing their own power supply should ensure that it delivers 8–24 V DC at 8 W. The voltage regulators will get warm while in use.

- **Regulated Power Supply**—The raw DC voltage provided at the 3-pin header is routed to a 5 V switching voltage regulator, then to a separate 3.3 V linear regulator. The regulators provide stable power to the RCM4200 module and the Prototyping Board.

- **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 RCM4200's **/RESET_IN** pin. Pressing the switch forces a hardware reset of the system.

- **I/O Switches and LEDs**—Two momentary-contact, normally open switches are connected to the PB4 and PB5 pins of the RCM4200 module and may be read as inputs by sample applications.

  Two LEDs are connected to the PB2 and PB3 pins of the RCM4200 module, and may be driven as output indicators by sample applications.

- **Prototyping Area**—A generous prototyping area has been provided for the installation of through-hole components. +3.3 V, +5 V, and Ground buses run around the edge of this area. Several areas for surface-mount devices are also available. (Note that there are SMT device pads on both top and bottom of the Prototyping Board.) Each SMT pad is connected to a hole designed to accept a 30 AWG solid wire.

- **Module Extension Header**—The complete pin set of the RCM4200 module is duplicated at header J2. Developers can solder wires directly into the appropriate holes, or, for more flexible development, a 2 × 25 header strip with a 0.1" pitch can be soldered into place. See Figure B-4 for the header pinouts.

  NOTE: The same Prototyping Board can be used for several series of RabbitCore modules, and so the signals at J2 depend on the signals available on the specific RabbitCore module.

- **Analog Inputs Header**—The Prototyping Board's analog signals are presented at header J3. These analog signals are connected via attenuator/filter circuits on the Prototyping Board to the corresponding analog inputs on the RCM4200 module. Developers can solder wires directly into the appropriate holes, or, for more flexible development, a 2 × 7 header strip with a 0.1" pitch can be soldered into place. See Figure B-4 for the header pinouts.

- **RS-232**—Two 3-wire or one 5-wire RS-232 serial ports are available on the Prototyping Board at header J4. A 10-pin 0.1" pitch header strip installed at J4 allows you to connect a ribbon cable that leads to a standard DE-9 serial connector.
- **Current Measurement Option**—You may cut the trace below header JP1 on the bottom side of the Prototyping Board and install a 1 × 2 header strip from the Development Kit to allow you to use an ammeter across the pins to measure the current drawn from the +5 V supply. Similarly, you may cut the trace below header JP2 on the bottom side of the Prototyping Board and install a 1 × 2 header strip from the Development Kit to allow you to use an ammeter across the pins to measure the current drawn from the +3.3 V supply.
- **Backup Battery**—A 2032 lithium-ion battery rated at 3.0 V, 220 mA·h, provides battery backup for the RCM4200 SRAM and real-time clock.

B.2 Mechanical Dimensions and Layout

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

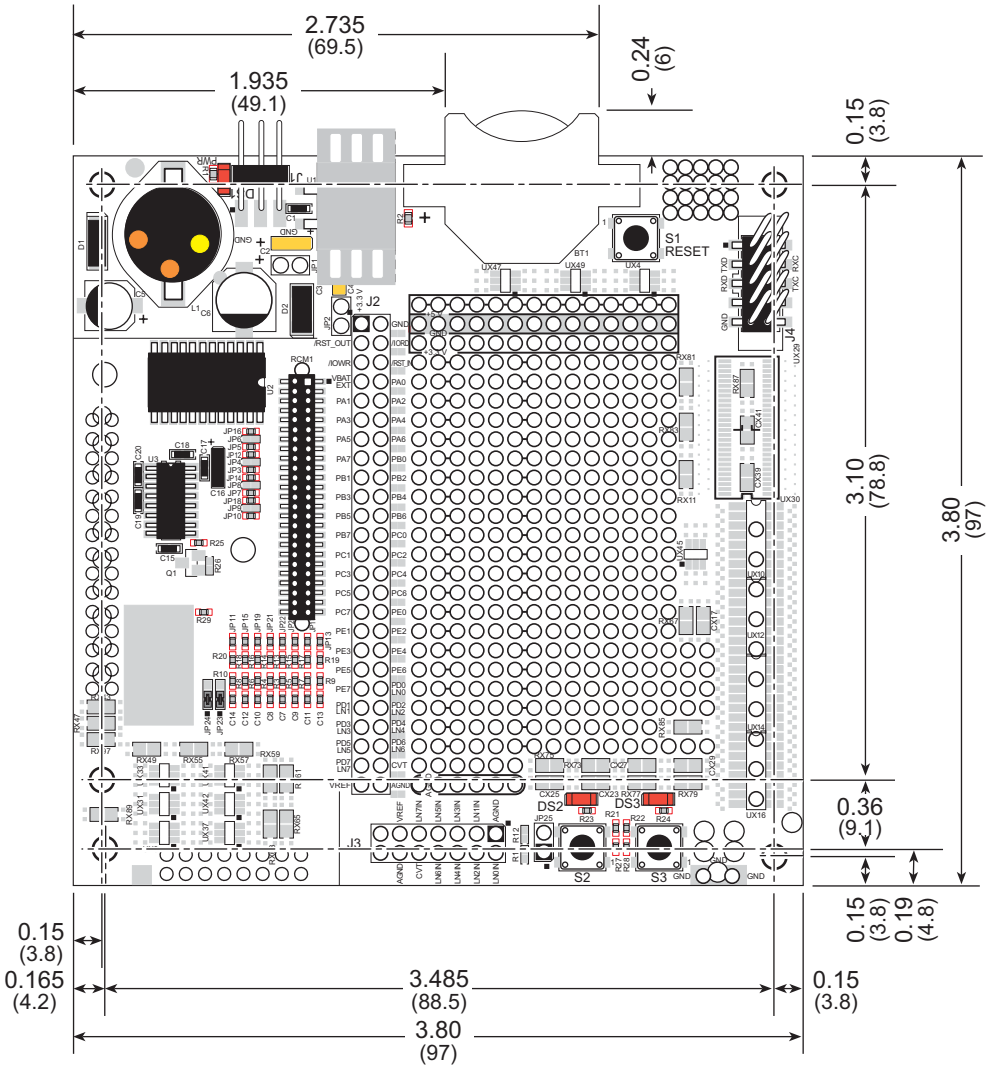


Figure B-2. Prototyping Board Dimensions

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

Table B-1. Prototyping Board Specifications

Parameter	Specification
Board Size	3.80" × 3.80" × 0.48" (97 mm × 97 mm × 12 mm)
Operating Temperature	0°C to +70°C
Humidity	5% to 95%, noncondensing
Input Voltage	8 V to 24 V DC
Maximum Current Draw (including user-added circuits)	800 mA max. for +3.3 V supply, 1 A total +3.3 V and +5 V combined
Prototyping Area	1.3" × 2.0" (33 mm × 50 mm) throughhole, 0.1" spacing, additional space for SMT components
Connectors	One 2 × 25 header socket, 1.27 mm pitch, to accept RCM4200 One 1 × 3 IDC header for power-supply connection One 2 × 5 IDC RS-232 header, 0.1" pitch Two unstuffed header locations for analog and RCM4200 signals 25 unstuffed 2-pin header locations for optional configurations

B.3 Power Supply

The RCM4200 requires a regulated 3.0 V – 3.6 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 +5 V switching power regulator from which a +3.3 V linear regulator draws its supply. Thus both +5 V and +3.3 V are available on the Prototyping Board.

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

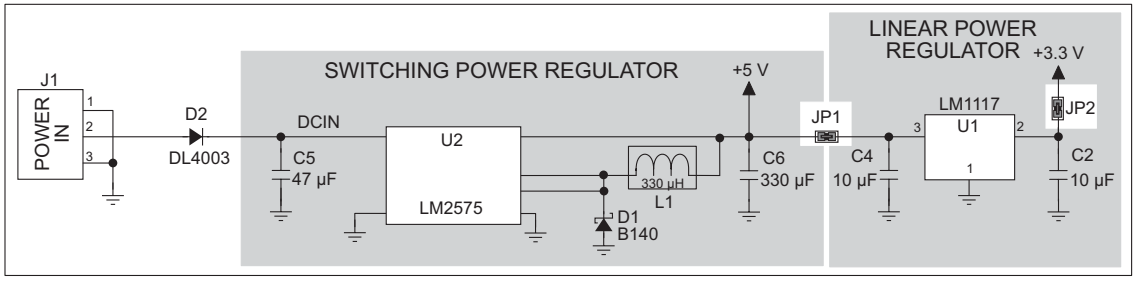


Figure B-3. Prototyping Board Power Supply

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 RCM4200 right out of the box without any modifications to either board.

The Prototyping Board comes with the basic components necessary to demonstrate the operation of the RCM4200. Two LEDs (DS2 and DS3) are connected to PB2 and PB3, and two switches (S2 and S3) are connected to PB4 and PB5 to demonstrate the interface to the Rabbit 4000 microprocessor. Reset switch S1 is the hardware reset for the RCM4200.

The Prototyping Board provides the user with RCM4200 connection points brought out conveniently to labeled points at header J2 on the Prototyping Board. Although header J2 is unstuffed, a 2 × 25 header is included in the bag of parts. RS-232 signals (Serial Ports C and D) are available on header J4. A header strip at J4 allows you to connect a ribbon cable, and a ribbon cable to DB9 connector is included with the Development Kit. The pinouts for these locations are shown in Figure B-4.

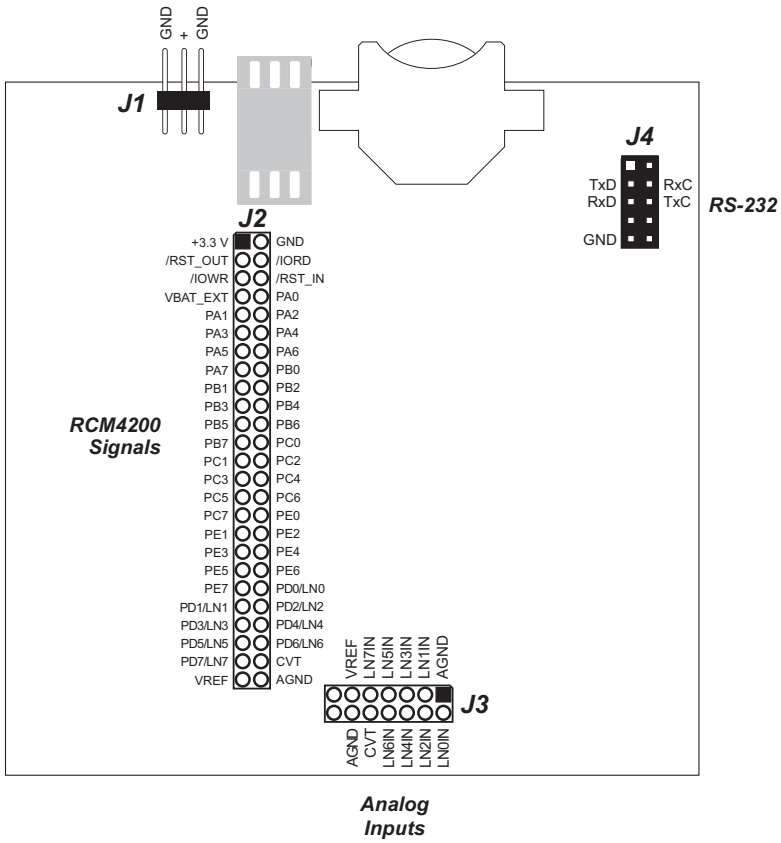


Figure B-4. Prototyping Board Pinout

The analog signals are brought out to labeled points at header location J3 on the Prototyping Board. Although header J3 is unstuffed, a 2 × 7 header can be added. Note that analog signals are only available from the RCM4200 included in the Development Kit — the RCM4210 model does not have an A/D converter.

All signals from the RCM4200 module are available on header J2 of the Prototyping Board. The remaining ports on the Rabbit 4000 microprocessor are used for RS-232 serial communication. Table B-2 lists the signals on header J2 and explains how they are used on the Prototyping Board.

Table B-2. Use of RCM4200 Signals on the Prototyping Board

Pin	Pin Name	Prototyping Board Use
1	+3.3 V	+3.3 V power supply
2	GND	
3	/RST_OUT	Reset output from reset generator
4	/IORD	External read strobe
5	/IOWR	External write strobe
6	/RESET_IN	Input to reset generator
8–15	PA0–PA7	Output, pulled high
16	PB0	CLKB (used by A/D converter RCM4200 only)
17	PB1	Programming port CLKA
18	PB2	LED DS2 (normally high/off)
19	PB3	LED DS3 (normally high/off)
20	PB4	Switch S2 (normally open/pulled up)
21	PB5	Switch S3 (normally open/pulled up)
22–23	PB6–PB7	Output, pulled high
24–25	PC0–PC1	Serial Port D (RS-232, header J4) (high)
26–27	PC2–PC3	Serial Port C (used by serial flash) (high)
28–29	PC4–PC5	Serial Port B (used by A/D converter RCM4200 only)
30–31	PC6–PC7	Serial Port A (programming port) (high)
32–33	PE0–PE1	Output (high)
34	PE2	External I/O strobe, Ethernet
35–38	PE3–PE6	Output (high)
39	PE7	Serial flash SCLK
40–47	LN0–LN7	A/D converter inputs (RCM4200 only)*
48	CONVERT	A/D converter CONVERT input (RCM4200 only)<Superscript>*
49	VREF	A/D converter reference voltage (RCM4200 only)<Superscript>*
50	AGND	A/D converter ground (RCM4200 only)<Superscript>*

* PD0–PD7 (output, high) are available on these pins for the RCM4210.

There is a 1.3" × 2" through-hole prototyping space available on the Prototyping Board. The holes in the prototyping area are spaced at 0.1" (2.5 mm). +3.3 V, +5 V, and GND traces run along the top edge of the prototyping area for easy access. Small to medium circuits can be prototyped using point-to-point wiring with 20 to 30 AWG wire between the prototyping area, the +3.3 V, +5 V, and GND traces, and the surrounding area where surface-

mount components may be installed. Small holes are provided around the surface-mounted components that may be installed around the prototyping area.

B.4.1 Adding Other Components

There are pads for 28-pin TSSOP devices, 16-pin SOIC devices, and 6-pin SOT devices that can be used for surface-mount prototyping with these devices. There are also 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.

B.4.2 Measuring Current Draw

The Prototyping Board has a current-measurement feature available at header locations JP1 and JP2 for the +5 V and +3.3 V supplies respectively. To measure current, you will have to cut the trace on the bottom side of the Prototyping Board corresponding to the power supply or power supplies whose current draw you will be measuring. Header locations JP1 and JP2 are shown in Figure B-5. Then install a 1 × 2 header strip from the Development Kit on the top side of the Prototyping Board at the header location(s) whose trace(s) you cut. The header strip(s) will allow you to use an ammeter across their pins to measure the current drawn from that supply. Once you are done measuring the current, place a jumper across the header pins to resume normal operation.

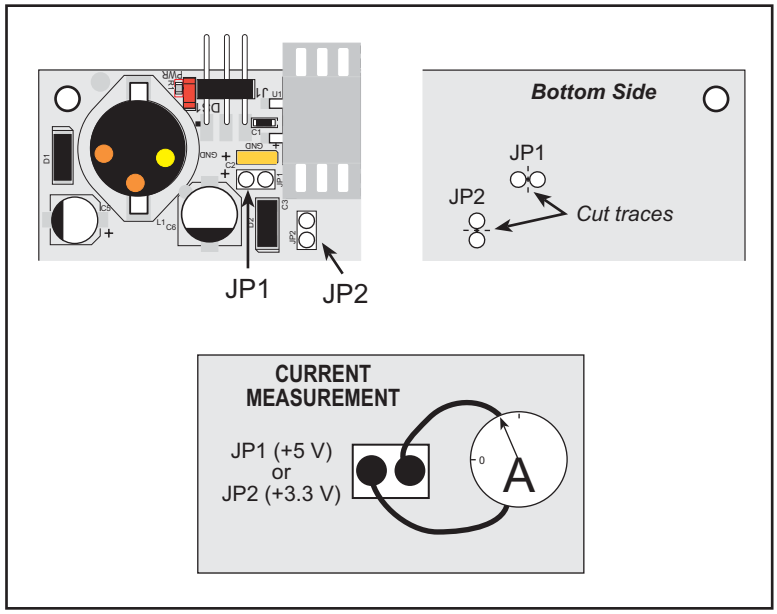


Figure B-5. Prototyping Board Current-Measurement Option

NOTE: Once you have cut the trace below header location JP1 or JP2, you must either be using the ammeter or have a jumper in place in order for power to be delivered to the Prototyping Board.

B.4.3 Analog Features (RCM4200 only)

The Prototyping Board has typical support circuitry installed to complement the ADS7870 A/D converter on the RCM4200 model (the A/D converter is not available on the RCM4210 model).

B.4.3.1 A/D Converter Inputs

Figure B-6 shows a pair of A/D converter input circuits. The resistors form an approx. 11:1 attenuator, and the capacitor filters noise pulses from the A/D converter input. The 470 Ω inline jumpers allow other configurations (see Table B-6) and provide digital isolation when you are not using an A/D converter (Parallel Port D is available). These jumpers optimize using RabbitCore modules with or without A/D converters—if you are designing your own circuit, the best performance for the A/D converter would be realized with 0 Ω resistors.

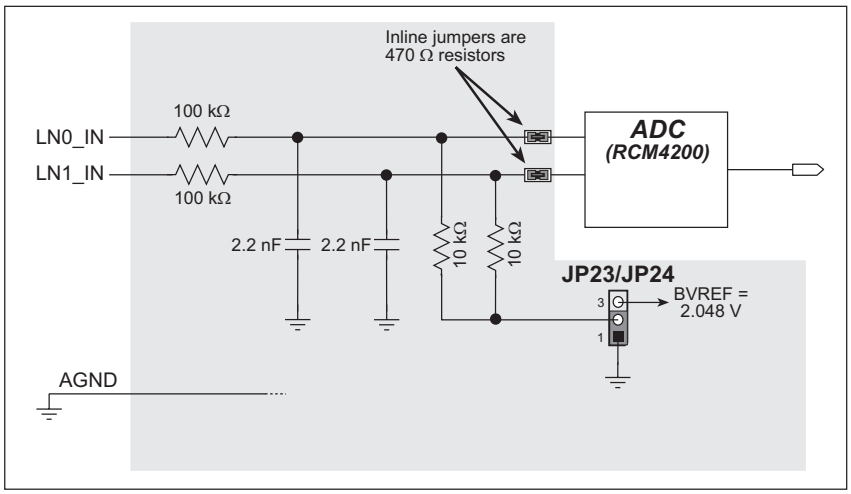


Figure B-6. A/D Converter Inputs

The A/D converter chip can make either single-ended or differential measurements depending on the value of the **opmode** parameter in the software function call. Adjacent A/D converter inputs are paired to make differential measurements. The default setup on the Prototyping Board is to measure only positive voltages for the ranges listed in Table B-3.

Table B-3. Positive A/D Converter Input Voltage Ranges

Min. Voltage (V)	Max. Voltage (with prescaler) (V)	Gain Multiplier	A/D Converter Actual Gain	Resolution (mV)
0.0	+22.528	×1	1	11
0.0	+11.264	×2	1.8	5.5
0.0	+5.632	×4	3.6	2.75
0.0	+4.506	×5	4.5	2.20
0.0	+2.816	×8	7.2	1.375
0.0	+2.253	×10	9.0	1.100
0.0	+1.408	×16	14.4	0.688
0.0	+1.126	×20	18	0.550

Many other possible ranges are possible by physically changing the resistor values that make up the attenuator circuit.

NOTE: Analog input LN7_IN does not have the 10 kΩ resistor installed, and so no resistor attenuator is available, limiting its maximum input voltage to 2 V. This input is intended to be used for a thermistor that you may install at header location JP25.

It is also possible to read a negative voltage on LN0_IN–LN5_IN by moving the 0 Ω jumper (see Figure B-6) on header JP23 or JP24 associated with the A/D converter input from analog ground to the reference voltage generated and buffered by the A/D converter. Adjacent input channels are paired; moving the jumper on JP 23 changes both of the paired channels (LN4_IN–LN5_IN), and moving the jumper on JP24 changes LN0_IN–LN1_IN and LN2_IN–LN3_IN. At the present time Rabbit Semiconductor does not offer the software drivers to work with single-ended negative voltages, but the differential mode described below may be used to measure negative voltages.

Differential measurements require two channels. As the name *differential* implies, the difference in voltage between the two adjacent channels is measured rather than the difference between the input and analog ground. Voltage measurements taken in differential mode have a resolution of 12 bits, with the 12th bit indicating whether the difference is positive or negative.

The A/D converter chip can only accept positive voltages, as explained in Section 4.4. Both differential inputs must be referenced to analog ground, and ***both inputs must be positive with respect to analog ground***. Table B-4 provides the differential voltage ranges for this setup.

Table B-4. Differential Voltage Ranges

Min. Differential Voltage (V)	Max. Differential Voltage (with prescaler) (V)	Gain Multiplier	A/D Converter Actual Gain	Resolution (mV)
0	±22.528	×1	1	11
0	±11.264	×2	1.8	5.5
0	±5.632	×4	3.6	2.75
0	±4.506	×5	4.5	2.20
0	±2.816	×8	7.2	1.375
0	±2.253	×10	9.0	1.100
0	±1.408	×16	14.4	0.688
0	±1.126	×20	18	0.550

B.4.3.2 Thermistor Input

Analog input LN7_IN on the Prototyping Board was designed specifically for use with a thermistor at JP25 in conjunction with the **THERMISTOR.C** sample program, which demonstrates how to use the analog input to measure temperature, which will be displayed in the Dynamic C **STDIO** window. The sample program is targeted specifically for the thermistor included with the Development Kit with R_0 @ 25°C = 3 kΩ and β 25/85 = 3965. Be sure to use the applicable R_0 and β values for your thermistor if you use another thermistor.

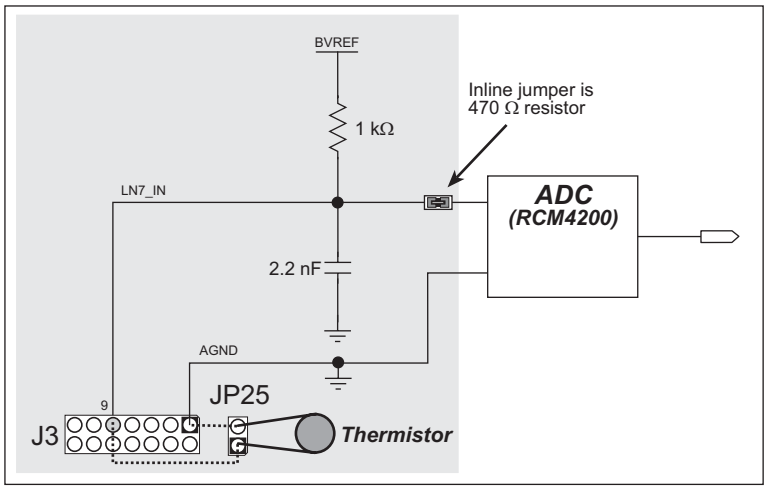


Figure B-7. Prototyping Board Thermistor Input

B.4.3.3 A/D Converter Calibration

To get the best results form the A/D converter, it is necessary to calibrate each mode (single-ended or differential) for each of its gains. It is imperative that you calibrate each of the A/D converter inputs in the same manner as they are to be used in the application. For example, if you will be performing floating differential measurements or differential measurements using a common analog ground, then calibrate the A/D converter in the corresponding manner. The calibration must be done with the JP23/JP24 selection jumpers in the desired position (see Figure B-6). If a calibration is performed and a jumper is subsequently moved, the corresponding input(s) must be recalibrated. The calibration table in software only holds calibration constants based on mode, channel, and gain. ***Other factors affecting the calibration must be taken into account by calibrating using the same mode and gain setup as in the intended use.***

Sample programs are available to illustrate how to read and calibrate the various A/D inputs for the single-ended operating mode.

Mode	Read	Calibrate
Single-Ended, one channel	—	**AD_CAL_CHAN.C**
Single-Ended, all channels	**AD_RDVOLT_ALL.C**	**AD_CAL_ALL.C**

B.4.4 Serial Communication

The Prototyping Board allows you to access the serial ports from the RCM4200 module. Table B-5 summarizes the configuration options. Note that Serial Ports E can be used only when the RCM4210 is installed on the Prototyping Board.

Table B-5. Prototyping Board Serial Port Configurations

Serial Port	Header	Default Use	Alternate Use
A	J2	Programming Port	RS-232
B	J2	A/D Converter (RCM4200 only)	RS-232
C	J2, J4	Serial Flash	—
D	J2, J4	RS-232	—
E	J2	RS-232	—

Serial Ports E may be used as a serial port, or the corresponding pins at header location J2 may be used as parallel ports.

B.4.4.1 RS-232

RS-232 serial communication on header J4 on both Prototyping Boards is supported by an RS-232 transceiver installed at U3. This transceiver provides the voltage output, slew rate, and input voltage immunity required to meet the RS-232 serial communication protocol. Basically, the chip translates the Rabbit 4000's signals to RS-232 signal levels. Note that the polarity is reversed in an RS-232 circuit so that a +3.3 V output becomes approximately -10 V and 0 V is output as +10 V. The RS-232 transceiver also provides the proper line loading for reliable communication.

RS-232 can be used effectively at the RCM4200 module's maximum baud rate for distances of up to 15 m.

RS-232 flow control on an RS-232 port is initiated in software using the **serXflowcontrolOn** function call from **RS232.LIB**, where **X** is the serial port (C or D). The locations of the flow control lines are specified using a set of five macros.

SERX_RTS_PORT—Data register for the parallel port that the RTS line is on (e.g., PCDR).

SERA_RTS_SHADOW—Shadow register for the RTS line's parallel port (e.g., PCDRShadow).

SERA_RTS_BIT—The bit number for the RTS line.

SERA_CTS_PORT—Data register for the parallel port that the CTS line is on (e.g., PCDRShadow).

SERA_CTS_BIT—The bit number for the CTS line.

Standard 3-wire RS-232 communication using Serial Ports C and D is illustrated in the following sample code.

```
#define CINBUFSIZE  15   // set size of circular buffers in bytes
#define COUTBUFSIZE 15

#define DINBUFSIZE  15
#define DOUTBUFSIZE 15

#define MYBAUD 115200   // set baud rate
#endif

main(){
    serCopen(_MYBAUD);   // open Serial Ports C and D
    serDopen(_MYBAUD);
    serCwrFlush();       // flush their input and transmit buffers
    serCrdFlush();
    serDwrFlush();
    serDrdFlush();
    serCclose(_MYBAUD);  // close Serial Ports C and D
    serDclose(_MYBAUD);
}
```

B.5 Prototyping Board Jumper Configurations

Figure B-8 shows the header locations used to configure the various Prototyping Board options via jumpers.

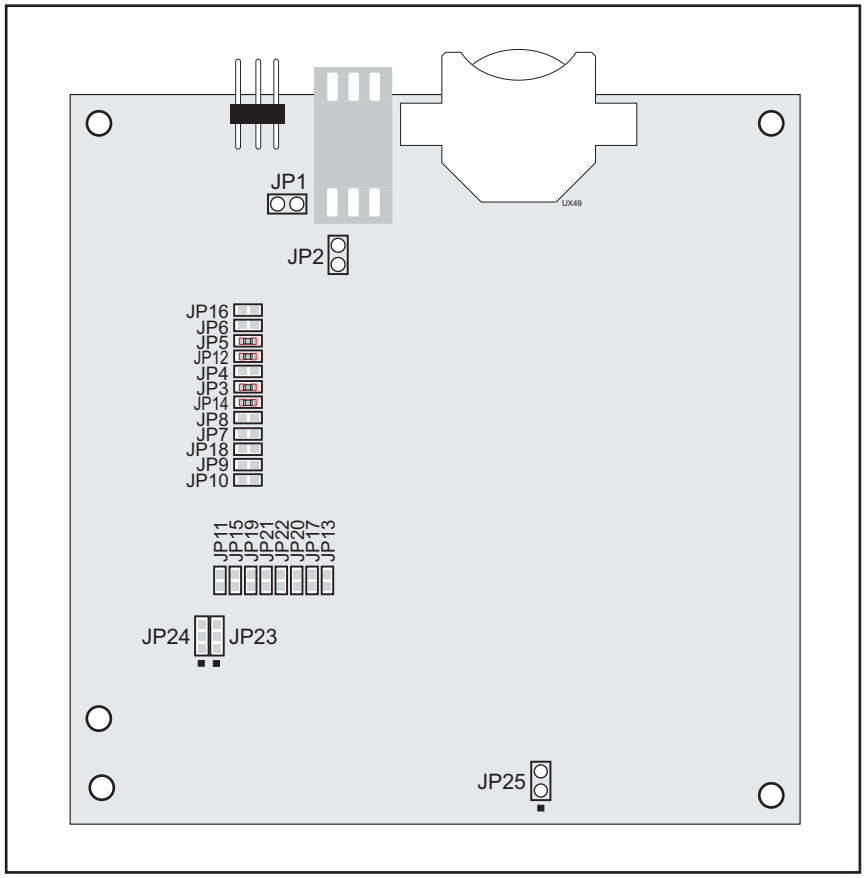


Figure B-8. Location of Configurable Jumpers on Prototyping Board

Table B-6 lists the configuration options using either jumpers or 0 Ω surface-mount resistors.

Table B-6. RCM4200 Prototyping Board Jumper Configurations

Header	Description	Pins Connected		Factory Default
JP1	+5 V Current Measurement	1–2	Via trace or jumper	Connected
JP2	+3.3 V Current Measurement	1–2	Via trace or jumper	Connected
JP3 JP4	PC0/TxD/LED DS2	JP3 1–2	TxD on header J4	✕
		JP4 1–2	PC0 to LED DS2	
		n.c.	PC0 available on header J2	

Table B-6. RCM4200 Prototyping Board Jumper Configurations (continued)

Header	Description	Pins Connected		Factory Default
JP5 JP6	PC1/RxD/Switch S2	JP5 1–2	RxD on header J4	✕
		JP6 1–2	PC1 to Switch S2	
		n.c.	PC1 available on header J2	
JP7 JP8	PC2/TxC/LED DS3	JP7 1–2	TxC on header J4	✕
		JP6 1–2	PC2 to LED DS3	
		n.c.	PC2 available on header J2	
JP9 JP10	PC3/RxC/Switch S3	JP9 1–2	PC3 to Switch S3	
		JP10 1–2	RxC on header J4	✕
		n.c.	PC3 available on header J2	
JP11	LN0 buffer/filter to RCM4200	1–2		Connected
JP12	PB2/LED DS2	1–2	Connected: PB2 to LED DS2	✕
		n.c.	PB2 available on header J2	
JP13	LN1 buffer/filter to RCM4200	1–2		Connected
JP14	PB3/LED DS3	1–2	Connected: PB3 to LED DS3	✕
		n.c.	PB3 available on header J2	
JP15	LN2 buffer/filter to RCM4200	1–2		Connected
JP16	PB4/Switch S2	1–2	Connected: PB4 to Switch S2	✕
		n.c.	PB4 available on header J2	
JP17	LN3 buffer/filter to RCM4200	1–2		Connected
JP18	PB5/Switch S3	1–2	Connected: PB5 to Switch S3	✕
		n.c.	PB5 available on header J2	
JP19	LN4 buffer/filter to RCM4200	1–2		Connected
JP20	LN5 buffer/filter to RCM4200	1–2		Connected
JP21	LN6 buffer/filter to RCM4200	1–2		Connected
JP22	LN7 buffer/filter to RCM4200	1–2		Connected

Table B-6. RCM4200 Prototyping Board Jumper Configurations (continued)

Header	Description	Pins Connected		Factory Default
JP23	LN4_IN–LN6_IN	1–2	Tied to analog ground	✕
		2–3	Tied to VREF	
JP24	LN0_IN–LN3_IN	1–2	Tied to analog ground	✕
		2–3	Tied to VREF	
JP25	Thermistor Location	1–2		n.c.

NOTE: Jumper connections JP3–JP10, JP12, JP14, JP16, JP18, JP23, and JP24 are made using 0 Ω surface-mounted resistors. Jumper connections JP11, JP13, JP15, JP17, and JP19–JP22 are made using 470 Ω surface-mounted resistors.

APPENDIX C. POWER SUPPLY

Appendix C provides information on the current requirements of the RCM4200, and includes some background on the chip select circuit used in power management.

C.1 Power Supplies

The RCM4200 requires a regulated 3.0 V – 3.6 V DC power source. The RabbitCore design presumes that the voltage regulator is on the user board, and that the power is made available to the RCM4200 board through header J2.

An RCM4200 with no loading at the outputs operating at 58.98 MHz typically draws 240 mA, and may draw up to 275 mA at 3.6 V and 85°C; the corresponding current draw for the RCM4210 is typically 200 mA, and up to 225 mA at 3.6 V and 85°C.

C.1.1 Battery Backup

The RCM4200 does not have a battery, but there is provision for a customer-supplied battery to back up the data SRAM and keep the internal Rabbit 4000 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 4000 real-time clock to retain data with the RCM4200 powered down.

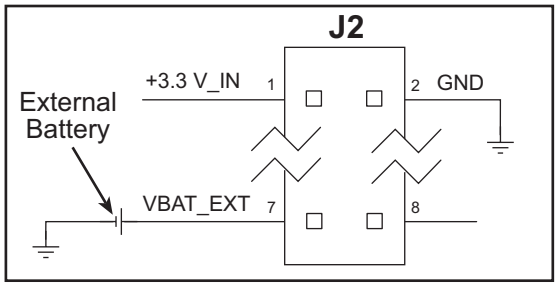


Figure C-1. External Battery Connections at Header J2

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 RCM4200 is typically 7.5 µA when no other power is supplied. If a 165 mA·h battery is used, the battery can last about 2.5 years:

$$\frac{165\ \text{mA}\cdot\text{h}}{7.5\ \mu\text{A}} = 2.5\ \text{years}.$$

The actual battery life in your application will depend on the current drawn by components not on the RCM4200 and on the storage capacity of the battery. The RCM4200 does not drain the battery while it is powered up normally.

Cycle the main power off/on after you install a backup battery for the first time, and whenever you replace the battery. This step will minimize the current drawn by the real-time clock oscillator circuit from the backup battery should the RCM4200 experience a loss of main power.

NOTE: Remember to cycle the main power off/on any time the RCM4200 is removed from the Prototyping Board or motherboard since that is where the backup battery would be located.

Rabbit Semiconductor's Technical Note TN235, ***External 32.768 kHz Oscillator Circuits***, provides additional information about the current draw by the real-time clock oscillator circuit.

C.1.2 Battery-Backup Circuit

Figure C-2 shows the battery-backup circuit.

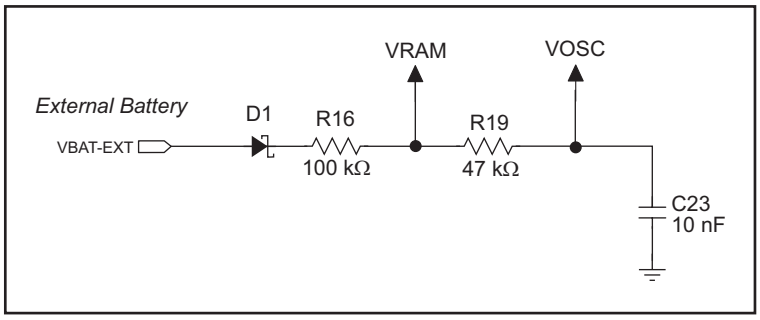


Figure C-2. RCM4200 Backup Battery Circuit

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 U6, which keeps the 32.768 kHz oscillator working when the voltage begins to drop.

C.1.3 Reset Generator

The RCM4200 uses a reset generator to reset the Rabbit 4000 microprocessor when the voltage drops below the voltage necessary for reliable operation. The reset occurs between 2.85 V and 3.00 V, typically 2.93 V. Since the RCM4200 will operate at voltages as low as 3.0 V, exercise care when operating close to the 3.0 V minimum voltage (for example, keep the power supply as close as possible to the RCM4200) since your RCM4200 could reset unintentionally.

The RCM4200 has a reset output, pin 3 on header J2.

INDEX

J

M

O

P

R

S

T

U

SCHEMATICS

090-0241 RCM4200 Schematic

www.rabbit.com/documentation/schemat/090-0241.pdf

090-0230 Prototyping Board Schematic

www.rabbit.com/documentation/schemat/090-0230.pdf

090-0128 Programming Cable Schematic

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

090-0252 USB Programming Cable Schematic

www.rabbit.com/documentation/schemat/090-0252.pdf

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