Digi - 101-0404 Datasheet

Welcome to E-XFL.COM

Understanding Embedded - Microcontroller, Microprocessor, FPGA Modules

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

Applications of Embedded - Microcontroller,

Details	
Product Status	Obsolete
Module/Board Type	MPU Core
Core Processor	Rabbit 2000
Co-Processor	-
Speed	25.8MHz
Flash Size	256KB
RAM Size	512KB
Connector Type	2 IDC Headers 2x20
Size / Dimension	1.9" x 2.3" (48.3mm x 58.4mm)
Operating Temperature	-40°C ~ 85°C
Purchase URL	https://www.e-xfl.com/product-detail/digi-international/101-0404

Email: info@E-XFL.COM

Address: Room A, 16/F, Full Win Commercial Centre, 573 Nathan Road, Mongkok, Hong Kong

TABLE OF CONTENTS

1. INTRODUCTION

The RabbitCore RCM2000 series is a family of microprocessor modules designed to be the heart of embedded control systems, providing an array of I/O and addressing.

Throughout this manual, the term RCM2000 refers to the complete series of RCM2000 RabbitCore modules unless other production models are referred to specifically.

The RCM2000 is a core module designed to be the heart of your own controller built around the plug-in module. Data processing is done by a Rabbit 2000 microprocessor operating at up to 25.8 MHz (RCM2000 and RCM2010).

The RCM2000 has a Rabbit 2000 microprocessor, a static RAM, a flash memory, two quartz crystals (main oscillator and timekeeping), and the circuitry necessary for reset and management of battery backup of the Rabbit 2000’s internal real-time clock and the static RAM. Two 40-pin headers bring out the Rabbit 2000 I/O bus, address lines, data lines, parallel ports, and serial ports.

The RCM2000 receives its +5 V power from the user board on which it is mounted. The RCM2000 can interface will all kinds of digital devices through the user board.

The RCM2000 Development Kit comes with a Prototyping Board that can be used to demonstrate the operation of the RCM2000 and to prototype new circuits.

1.1 Features

- Small size: 1.90" × 2.30" (48.3 mm × 58.4 mm)
- Microprocessor: Rabbit 2000 running at 25.8 MHz (RCM2000 and RCM2010)
- 40 CMOS-compatible parallel I/O lines grouped in five 8-bit ports (shared with serial ports)
- 8 data lines (D0–D7)
- 13 address lines (A0–A12)
- I/0 read, write, buffer enable
- Status, watchdog and clock outputs
- Two startup mode inputs for master/slave configuration

2.1 Connections

1. Attach RCM2000 to Prototyping Board

Turn the RCM2000 so that the Rabbit 2000 microprocessor is facing as shown below. Plug RCM2000 headers J1 and J2 on the bottom side of the RCM2000 into the sockets of headers J1 and J3 on the Prototyping Board.

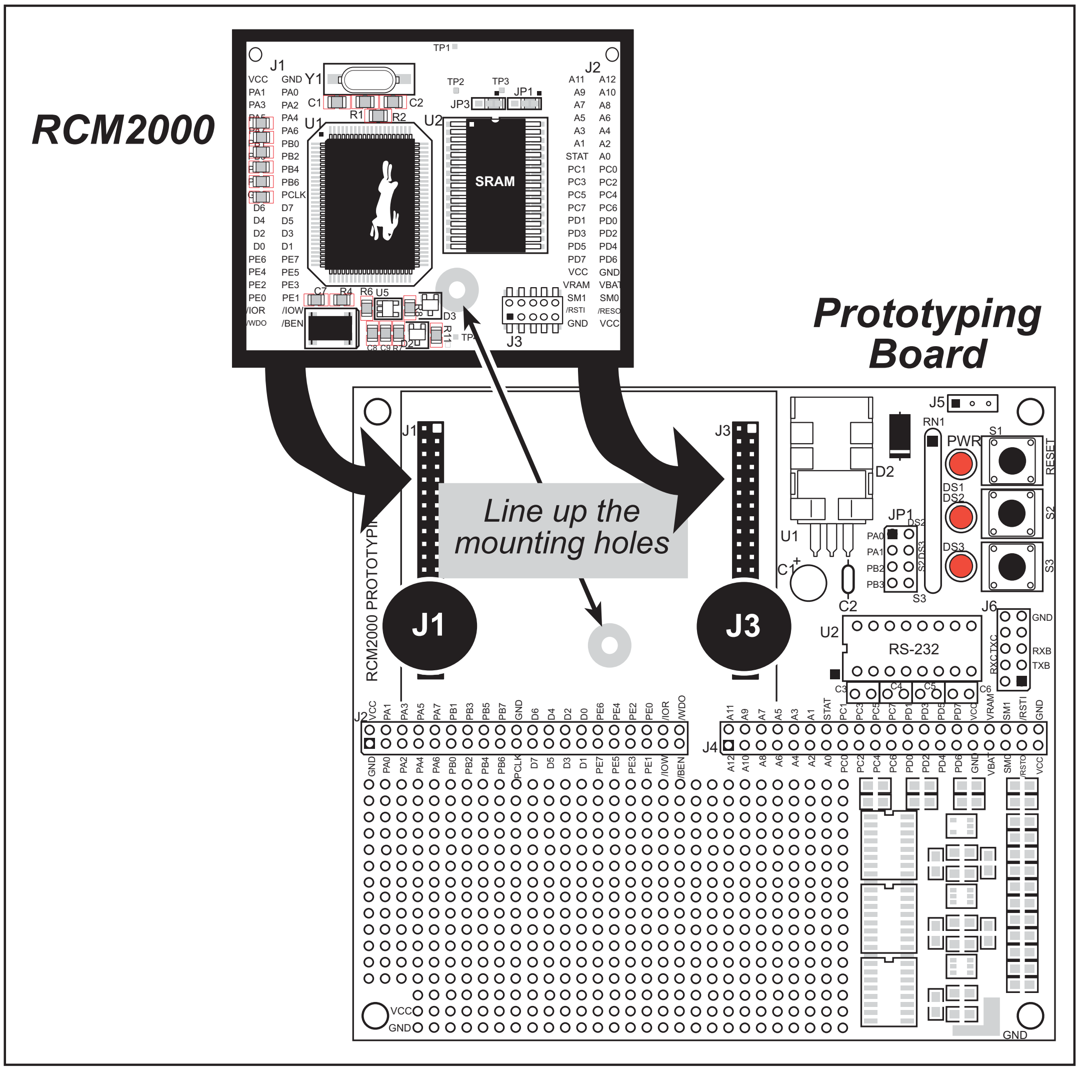


Figure 1. Attaching RCM2000 to Prototyping Board

NOTE: It is important that you line up the pins on the RCM2000 headers J1 and J2 exactly with the corresponding pins of header sockets J1 and J3 on the Prototyping Board. The header pins may become bent or damaged if the pin alignment is offset, and the RCM2000 will not work.

2. Connect RCM2000 to PC

Connect the 10-pin connector of the programming cable labeled **PROG** to header J3 on the RCM2000 module as shown in Figure 2 below. Be sure to orient the red edge of the cable towards pin 1 of the connector. (Do not use the **DIAG** connector, which is used for a normal serial connection.)

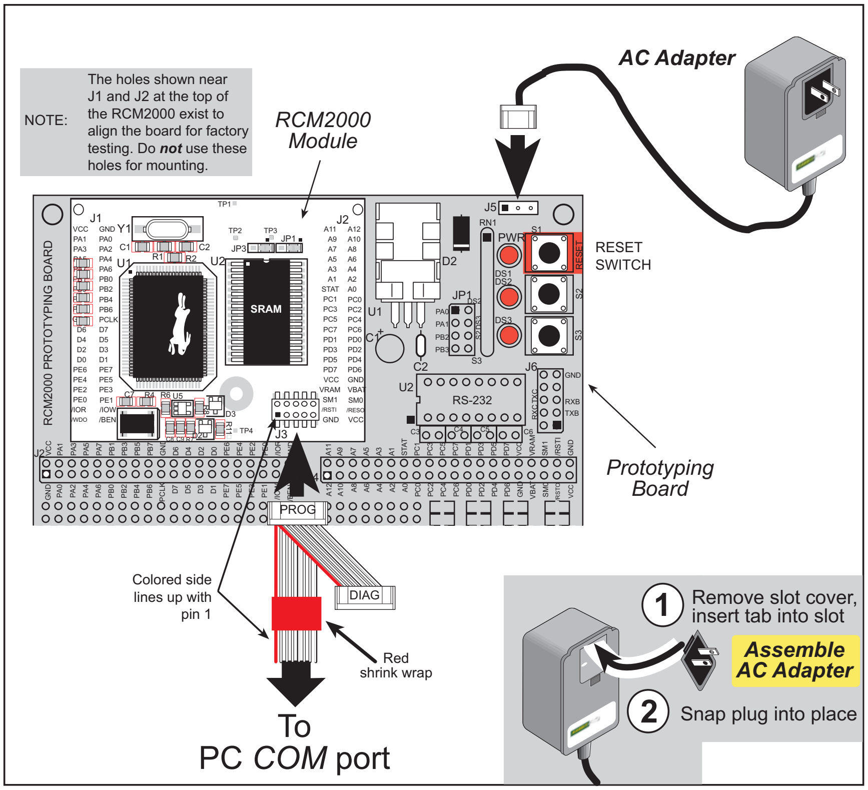


Figure 2. RCM2000 Power and Programming Connections

NOTE: Some PCs now come equipped only with a USB port. It may be possible to use an RS-232/USB converter (Part No. 20-151-0178) with the programming cable supplied with the RCM2000 Development Kit. Note that not all RS-232/USB converters work with Dynamic C.

2.3 Where Do I Go From Here?

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

1. Run all of the sample programs described in Chapter 3 to get a basic familiarity with Dynamic C and the RCM2000's capabilities.
2. For further development, refer to the ***RabbitCore RCM2000 User's Manual*** for details of the module's hardware and software components.

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

2.3.1 Technical Support

> **NOTE:** If you purchased your RCM2000 through a distributor or through a Rabbit partner, contact the distributor or partner first for technical support.

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

- A message reports “No Rabbit Processor Detected” in cases where the RCM2000 and the Prototyping Board are not connected together, the wall transformer is not connected, or is not plugged in. (The red power LED lights whenever power is connected.)
- The programming cable must be connected to the RCM2000. (The colored wire on the programming cable is closest to pin 1 on header J3 on the RCM2000, as shown in Figure 2.) The other end of the programming cable must be connected to the PC serial port. The COM port specified in the Dynamic C **Options** menu must be the same as the one the programming cable is connected to.
- To check if you have the correct serial port, select **Compile**, then **Compile BIOS**, or type **<Ctrl-Y>**. If the “BIOS successfully compiled …” message does not display, try a different serial port using the Dynamic C **Options** menu until you find the serial port you are plugged into. Don’t change anything in this menu except the COM number. The baud rate should be 115,200 bps and the stop bits should be 1.

3.1.1.1 Single-Stepping

Compile or re-compile `FLASHLED.C` by clicking the **Compile** button on the task bar. The program will compile and the screen will come up with a highlighted character (green) at the first executable statement of the program. Use the **F8** key to single-step. Each time the **F8** key is pressed, the cursor will advance one statement. When you get to the `for(j=0, j< ...` statement, it becomes impractical to single-step further because you would have to press **F8** thousands of times. We will use this statement to illustrate watch expressions.

3.1.1.2 Watch Expressions

Type **<Ctrl-W>** or chose **Add/Del Watch Expression** in the **Inspect** menu. A box will come up. Type the lower case letter j and click on *add to top* and *close*. Now continue single-stepping with **F8**. Each time you step, the watch expression (`j`) will be evaluated and printed in the watch window. Note how the value of `j` advances when the statement `j++` is executed.

3.1.1.3 Break Point

Move the cursor to the start of the statement:

```
for(j=0; j<25000; j++);
```

To set a break point on this statement, type **F2** or select **Toggle Breakpoint** from the **Run** menu. A red highlight will appear on the first character of the statement. To get the program running at full speed, type **F9** or select **Run** on the **Run** menu. The program will advance until it hits the break point. Then the break point will start flashing and show both red and green colors. Note that LED DS3 is now solidly turned on. This is because we have passed the statement turning on LED DS3. Note that `j` in the watch window has the value 32000. This is because the loop above terminated when `j` reached 32000.

To remove the break point, type **F2** or select **Toggle Breakpoint** on the **Run** menu. To continue program execution, type **F9** or select **Run** from the **Run** menu. Now the LED should be flashing again since the program is running at full speed.

```
#define DS2 0            // predefine for LED DS2
#define DS3 1            // predefine for LED DS3

//  This cofunction flashes LED on for ontime, then off for offtime
cofunc flashled[4](int led, int ontime, int offtime) {
    for(;;) {
        waitfor(DelayMs(ontime));                           // on delay
        WrPortI(PADR,&PADRShadow,(1<<led)|PADR);            // turn LED off
        waitfor(DelayMs(offtime);                           // off delay
        WrPortI(PADR,&PADRShadow,(1<<led)^0xff&PADR);       // turn LED on
    }
}

main {
    // Initialize ports
    WrPortI(SPCR,&SPCRShadow,0x84);    // Set Port A all outputs, LEDs on
    WrPortI(PEFR,&PEFRShadow,0x00);    // Set Port E normal I/O
    WrPortI(PEDDR,&PEDDRShadow,0x01);  // Set Port E bits 7…1 input, 0 output
    WrPortI(PECR,&PECRShadow,0x00);    // Set transfer clock as pclk/2

    for(;;) {                          // run forever
        costate {                      // start costatement
            wfd {                // use wfd (waitfordone) with cofunctions
              flashled[0](DS2,200,200); // flash DS2 on 200 ms, off 200 ms
              flashled[1](DS3,1000,500);// flash DS3 on 1000 ms, off 500 ms
            }
        }                               // end costatement
    }                                   // end for loop
}                                        // end of main, never come here
```

The flashing of the LEDs is performed by the costatement. Costatements need to be executed regularly, often at least every 25 ms. To accomplish this, the costatements are enclosed in a **`while`** loop or a **`for`** loop. The term **`while`** loop is used as a handy way to describe a style of real-time programming in which most operations are done in one loop.

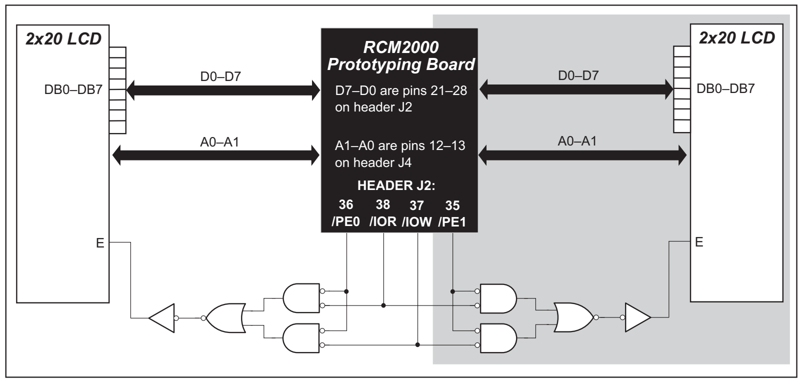
The costatement is executed on each pass through the big loop. When a **`waitfor`** or a **`wfd`** condition is encountered the first time, the current value of **`MS_TIMER`** is saved and then on each subsequent pass the saved value is compared to the current value. If a **`waitfor`** condition is not encountered, then a jump is made to the end of the costatement, and on the next pass of the loop, when the execution thread reaches the beginning of the costatement, execution passes directly to the **`waitfor`** statement. The costatement has the property that it can wait for long periods of time, but not use a lot of execution time. Each costatement is a little program with its own statement pointer that advances in response to conditions. On each pass through the big loop, as little as one statement in the costatement is executed, starting at the current position of the costatement's statement pointer. Consult the ***Dynamic C User's Manual*** for more details.

This program also illustrates a use for a shadow register. A shadow register is used to keep track of the contents of an I/O port that is write only—it can't be read back. If every time a write is made to the port the same bits are set in the shadow register, then the shadow register has the same data as the port register.

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

  Connect the LCD to the RCM2000 address and data lines on the Prototyping Board.

  D0—DB0
  D1—DB1
  D2—DB2
  D3—DB3
  D4—DB4
  D5—DB5
  D6—DB6
  D7—DB7



  A0—RS (Register Select: 0 = command, 1 = data)
  A1—R/W (0=write, 1=read)
  *—E (normally low: latches on high-to-low transition)

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

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

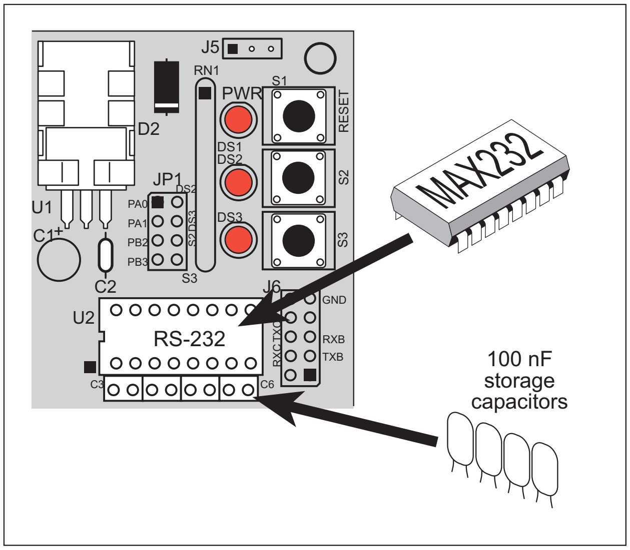
  Parallel Port B bit 2 = switch S2
  Parallel Port B bit 3 = switch S3

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

3.1.3 Serial Communication

The following sample programs can be found in the `SAMPLES\RCM2000` folder.

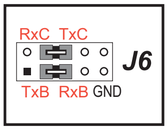
Two sample programs, `CORE_FLOWCONTROL.C` and `CORE_PARITY.C`, are available to illustrate RS-232 communication. To run these sample programs, you will have to add an RS-232 transceiver such as the MAX232 at location U2 and four 100 nF charge-storage capacitors at C3–C6 on the Prototyping Board. Also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



The diagram shows the connections.

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

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

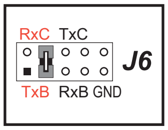
  

  A repeating triangular pattern should print out in the **STDIO** window. The program will periodically switch flow control on or off to demonstrate the effect of no flow control.

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

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

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

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

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

4.1.1 Dedicated Inputs

PB0 and PB1 are designated as inputs because the Rabbit 2000 is operating in an asynchronous mode. Four of the input-only pins are located on PB2–PB5. When Port C is used as a parallel port, PC1, PC3, PC5, and PC7 are also inputs only. All the inputs are pulled up with 47 kΩ resistors. Figure 6 shows the locations of these pullup resistors.

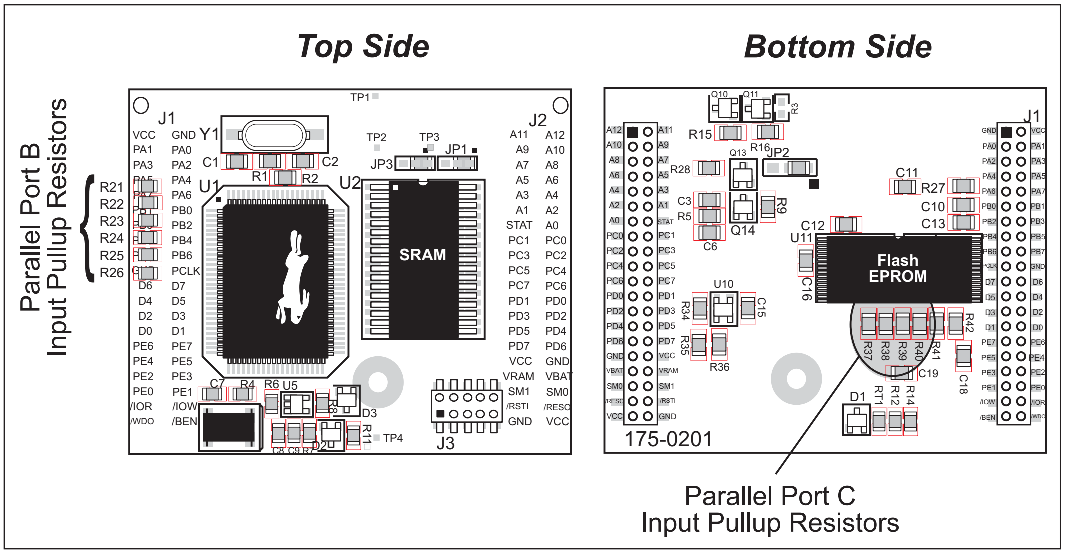


Figure 6. Locations of Digital Input Pullup Resistors

> **NOTE:** All the digital input pullup resistors are located on the bottom side of RCM2000 boards marked 175-0168

PB2–PB5 can instead be used for the slave port. PB2 and PB3 are slave write and slave read strobes, while PB4 and PB5 serve as slave address lines SA0 and SA1, and are used to access the slave registers (SD0–SD7), which is the alternate assignment for Parallel Port A. Parallel Port C pins PC1, PC3, PC5, and PC7 are inputs only can alternately be selectively enabled to serve as the serial data inputs for Serial Ports D, C, B, and A.

4.1.2 Dedicated Outputs

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

4.6 Memory

4.6.1 SRAM

The RCM2000 is designed to accept 32K to 512K of SRAM packaged in an SOIC case.

4.6.2 Flash EPROM

The RCM2000 is also designed to accept 128K to 512K of flash EPROM packaged in a TSOP case.

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

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

A Flash Memory Bank Select jumper configuration option based on 0 Ω surface-mounted resistors exists at header JP3. This option, used in conjunction with some configuration macros, allows Dynamic C to compile two different co-resident programs for the upper and lower halves of the 512K flash in such a way that both programs start at logical address 0000. This is useful for applications that require a resident download manager and a separate downloaded program. See Technical Note 218, ***Implementing a Serial Download Manager for a 256K Flash***, for details.

4.6.3 Dynamic C BIOS Source Files

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

- 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.

B.1 Overview of the Prototyping Board

The Prototyping Board included in the Development Kit makes it easy to connect an RCM2000 module to a power supply and a PC workstation for development. It also provides an array of basic I/O peripherals (switches and LEDs), as well as a prototyping area for more advanced hardware development.

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

As you progress to more sophisticated experimentation and hardware development, modifications and additions can be made to the board without modifying or damaging the RCM2000 itself.

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

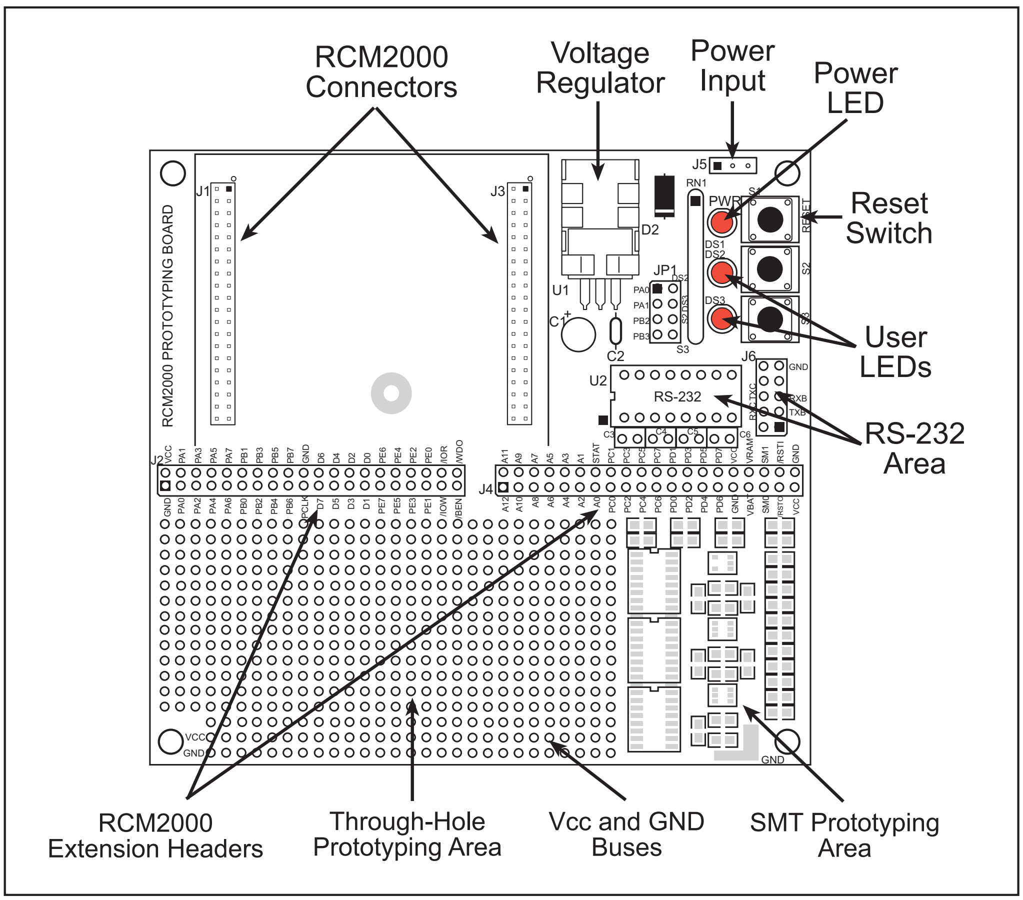


Figure B-1. RCM2000 Prototyping Board

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

Table B-1. Prototyping Board Specifications

Parameter	Specification
Board Size	4.00" × 4.30" × 1.19" (102 mm × 110 mm × 30 mm)
Operating Temperature	–40°C to +70°C
Humidity	5% to 95%, noncondensing
Input Voltage	7.5 V to 25 V DC
Maximum Current Draw (including user-added circuits)	1 A at 12 V and 25°C, 0.7 A at 12 V and 70ºC
Prototyping Area	2" × 3" (51 mm × 76 mm) throughhole, 0.1" spacing
Standoffs/Spacers	4, accept 6-32 × 3/8 screws

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 RCM2000 right out of the box without any modifications to either board. There are no jumpers or dip switches to configure or misconfigure on the Prototyping Board so that the initial setup is very straightforward.

The Prototyping Board comes with the basic components necessary to demonstrate the operation of the RCM2000. Two LEDs (DS2 and DS3) are connected to PA0 and PA1, and two switches (S2 and S3) are connected to PB2 and PB3 to demonstrate the interface to the Rabbit 2000 microprocessor. Reset switch S1 is the hardware reset for the RCM2000.

To maximize the availability of RCM2000 resources, the demonstration hardware (LEDs and switches) on the Prototyping Board may be disconnected. This is done by cutting the traces below the silk-screen outline of header JP1 on the bottom side of the Prototyping Board. Figure B-4 shows the four places where cuts should be made. An exacto knife would work nicely to cut the traces. Alternatively, a small standard screwdriver may be carefully and forcefully used to wipe through the PCB traces.

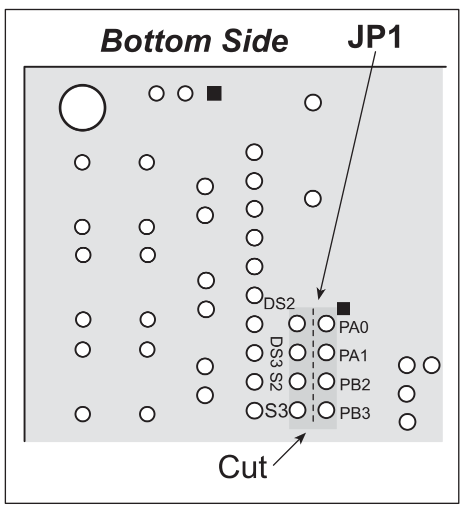


Figure B-4. Where to Cut Traces to Permanently Disable Demonstration Hardware on Prototyping Board

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

The drain on the battery by the RCM2000 is typically 10 µA when no other power is supplied. If a 950 mA·h battery is used, the battery can last more than 6 years:

$$\frac{950\ \text{mA}\cdot\text{h}}{10\ \mu\text{A}} = 10.8\ \text{years (shelf life = 10 years).}$$

Since the shelf life of the battery is 10 years, the battery can last for its full shelf life. The actual life in your application will depend on the current drawn by components not on the RCM2000 and the storage capacity of the battery.

C.1.2 Battery-Backup 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 U5, which keeps the 32.768 kHz oscillator working when the voltage begins to drop.

VRAM and Vcc are nearly equal (<100 mV, typically 10 mV) when power is supplied to the RCM2000.

Figure C-2 shows the RCM2000 battery-backup circuit.

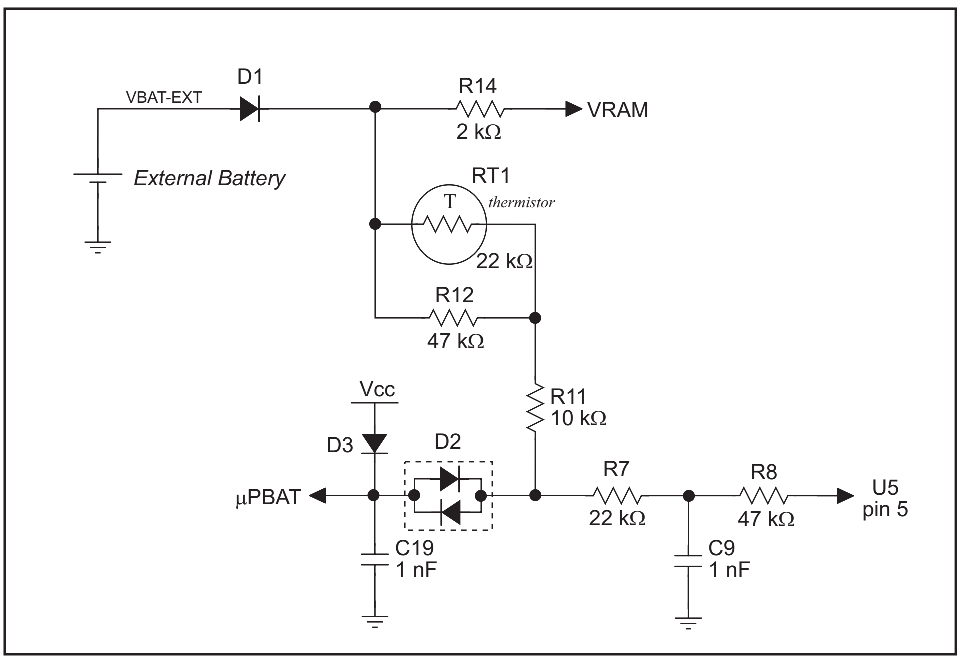


Figure C-2. RCM2000 Battery-Backup Circuit

VRAM is also available on pin 34 of header J2 to facilitate battery backup of the external circuit. Note that the recommended minimum resistive load at VRAM is 100 kΩ, and new battery life calculations should be done to take external loading into account.

SCHEMATICS

090-0097 RCM2000 Schematic

www.rabbit.com/documentation/schemat/090-0097.pdf

090-0099 RCM2000 Prototyping Board Schematic

www.rabbit.com/documentation/schemat/090-0099.pdf

090-0128 Programming Cable Schematic

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

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