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	128KB
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/101-0405

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.

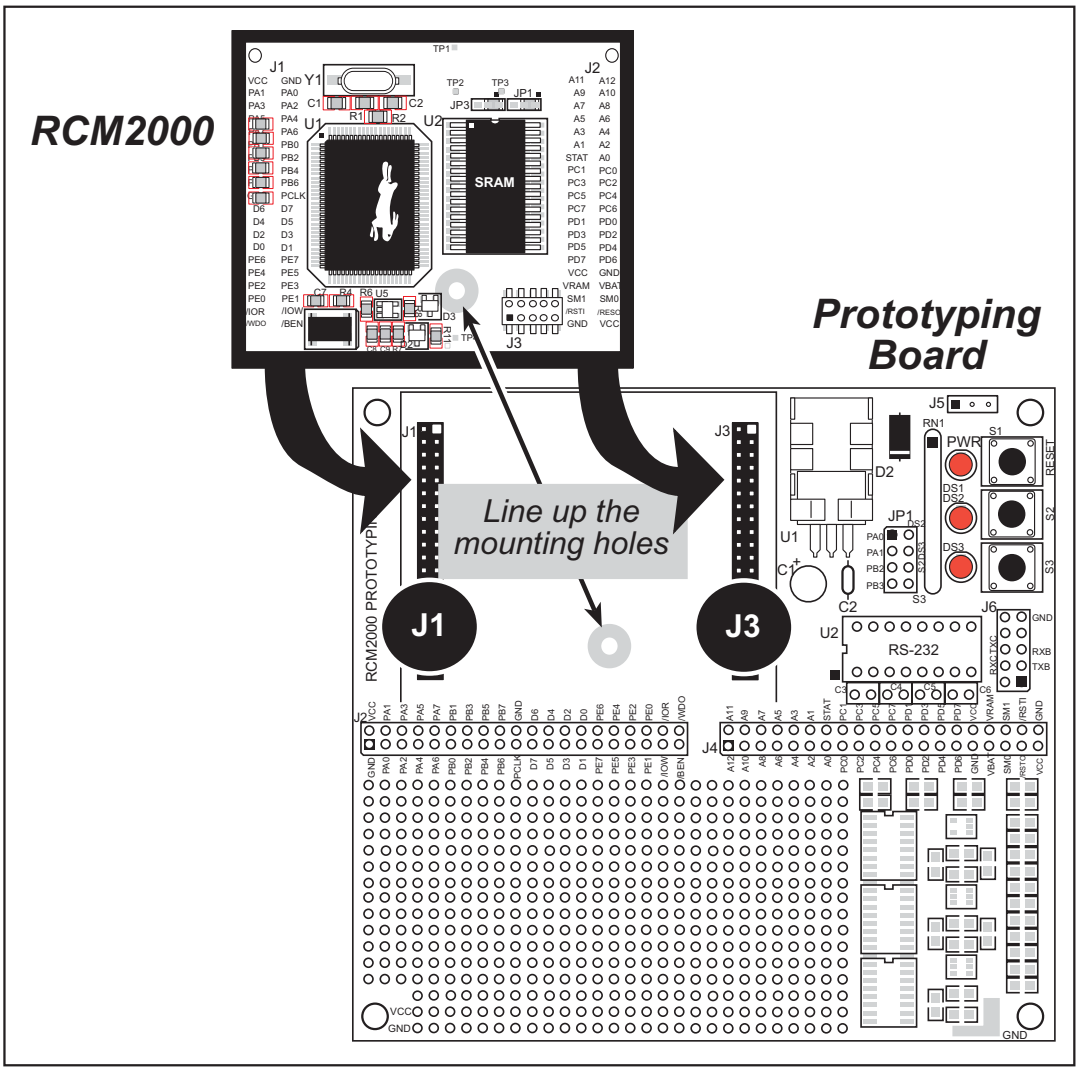

Figure 1. Attaching RCM2000 to Prototyping Board

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.)

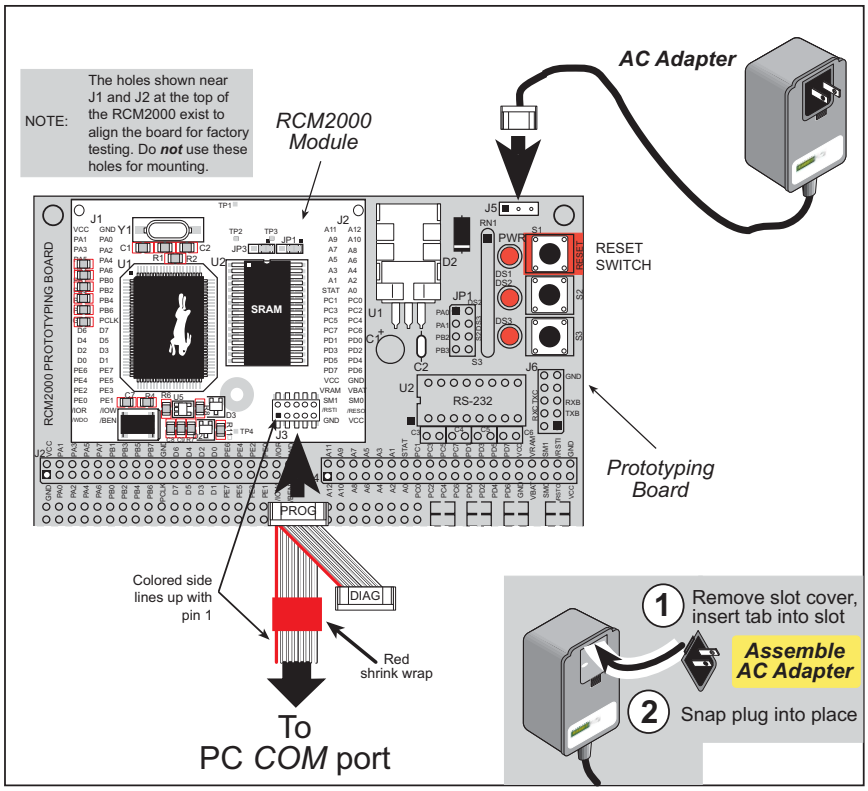

Figure 2. RCM2000 Power and Programming Connections

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.

3. Power Supply Connections

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

First, prepare the AC adapter for the country where it will be used by selecting the plug. The RCM2000 Development Kit presently includes Canada/Japan/U.S., Australia/N.Z., U.K., and European style plugs. Snap in the top of the plug assembly into the slot at the top of the AC adapter as shown in Figure 2, then press down on the spring-loaded clip below the plug assembly to allow the plug assembly to click into place.

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

Plug in the AC adapter. The power LED on the Prototyping Board should light up. The RCM2000 and the Prototyping Board are now ready to be used.

NOTE: A RESET button is provided on the Prototyping Board to allow a hardware reset.

To power down the Prototyping Board, unplug the power connector from J5. You should disconnect power before making any circuit adjustments in the prototyping area, changing any connections to the board, or removing the RCM2020 from the Prototyping Board.

2.1.1 Alternate Power Supply Connections

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

3.1.1 Running Sample Program FLASHLED.C

This sample program will be used to illustrate some of the functions of Dynamic C.

First, open the file **FLASHLED.C**, which is in the **SAMPLES/RCM2000** folder. The program will appear in a window, as shown in Figure 3 below (minus some comments). Use the mouse to place the cursor on the function name **WrPortI** in the program and type **<Ctrl-H>**. This will bring up a documentation box for the function **WrPortI**. In general, you can do this with all functions in Dynamic C libraries, including libraries you write yourself. Close the documentation box and continue.

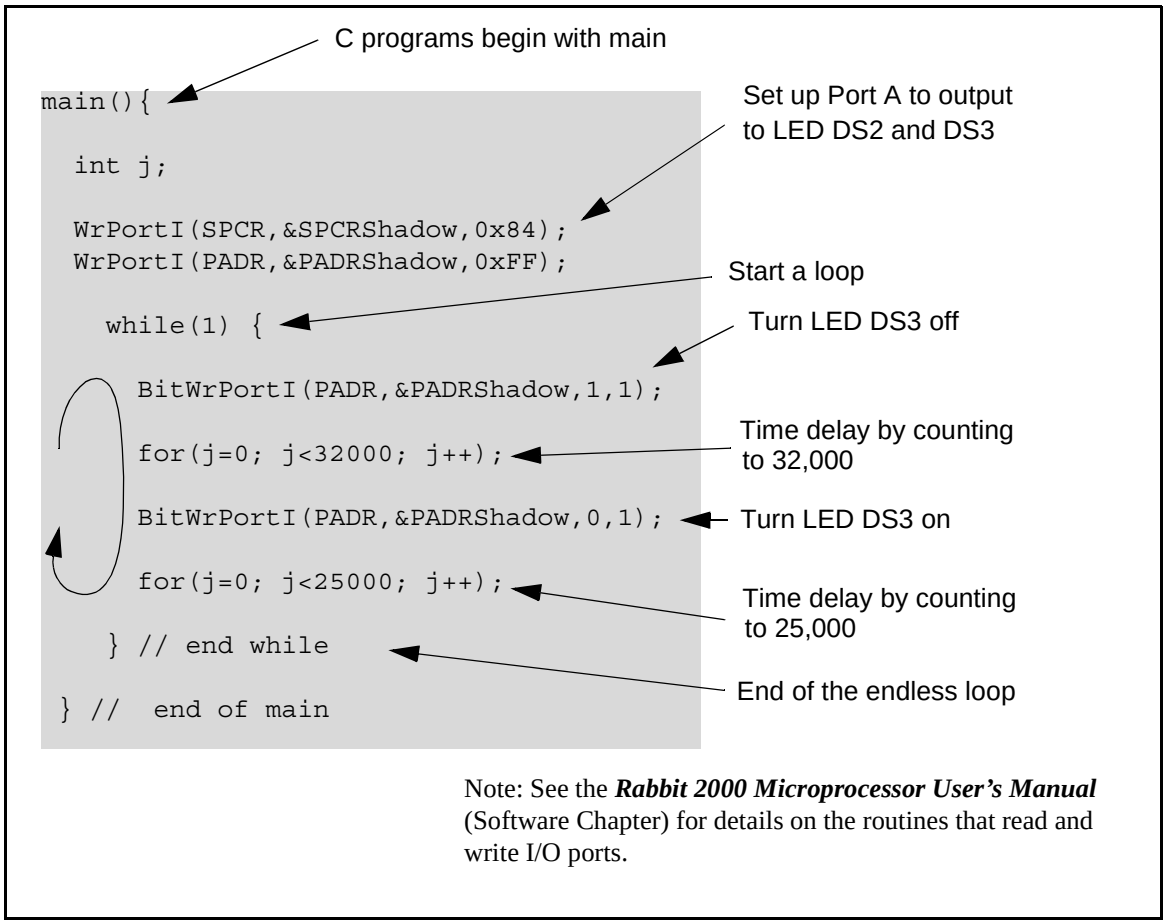

Figure 3. Sample Program FLASHLED.C

To run the program **FLASHLED.C**, open it with the **File** menu (if it is not already open), then compile and run it by pressing **F9** or by selecting **Run** in the **Run** menu. The LED on the Prototyping Board should start flashing if everything went well. If this doesn't work review the following points.

- The target should be ready, which is indicated by the message "BIOS successfully compiled..." If you did not receive this message or you get a communication error, recompile the BIOS by typing **<Ctrl-Y>** or select **Recompile BIOS** from the **Compile** menu.

You can set break points while the program is running by positioning the cursor to a statement and using the **F2** key. If the execution thread hits the break point, a break point will take place.You can toggle the break point off with the **F2** key and continue execution with the **F9** key. Try this a few times to get the feel of things.

3.1.1.4 Editing the Program

Click on the **Edit** box on the task bar. This will set Dynamic C into the edit mode so that you can change the program. Use the **Save as** choice on the **File** menu to save the file with a new name so as not to change the demo program. Save the file as **MYTEST.C**. Now change the number 25000 in the **for (..** statement to 10000. Then use the **F9** key to recompile and run the program. The LED will start flashing, but it will flash much faster than before because you have changed the loop counter terminal value from 25000 to 10000.

3.1.1.5 Watching Variables Dynamically

Go back to edit mode (select edit) and load the program **FLASHLED2.C** using the **File** menu **Open** command. This program is the same as the first program, except that a variable **k** has been added along with a statement to increment **k** each time around the endless loop. The statement:

```
runwatch();
```

has been added. This is a debugging statement that makes it possible to view variables while the program is running.

Use the **F9** key to compile and run **FLASHLED2.C**. Now type **<Ctrl-W>** to open the watch window and add the watch expression **k** to the top of the list of watch expressions. Now type **<Ctrl-U>**. Each time you type **<Ctrl-U>**, you will see the current value of **k**, which is incrementing about 5 times a second.

As an experiment, add another expression to the watch window:

```
k*5
```

Then type **<ctrl-U>** several times to observe the watch expressions **k** and **k*5**.

3.1.1.6 Summary of Features

So far you have practiced using the following features of Dynamic C.

- Loading, compiling and running a program. When you load a program it appears in an edit window. You can compile by selecting **Compile** on the task bar or from the **Compile** menu. When you compile the program, it is compiled into machine language and downloaded to the target over the serial port. The execution proceeds to the first statement of main where it pauses, waiting for you to command the program to run, which you can do with the **F9** key or by selecting **Run** on the **Run** menu. If want to compile and start the program running with one keystroke, use **F9**, the run command. If the program is not already compiled, the run command will compile it first.
- Single-stepping. This is done with the **F8** key. The **F7** key can also be used for single-stepping. If the **F7** key is used, then descent into subroutines will take place. With the **F8** key the subroutine is executed at full speed when the statement that calls it is stepped over.

4.4 Serial Programming Cable

The programming cable is used to connect the RCM2000's programming port to a PC serial COM port. The programming cable converts the RS-232 voltage levels used by the PC serial port to the TTL voltage levels used by the Rabbit 2000.

When the **PROG** connector on the programming cable is connected to the RCM2000's programming header, programs can be downloaded and debugged over the serial interface.

The **DIAG** connector of the programming cable may be used on the RCM2000's programming header with the RCM2000 operating in the Run Mode. This allows the programming port to be used as a regular serial port.

4.4.1 Changing Between Program Mode and Run Mode

The RCM2000 is automatically in Program Mode when the **PROG** connector on the programming cable is attached to the RCM2000, and is automatically in Run Mode when no programming cable is attached. When the Rabbit 2000 is reset, the operating mode is determined by the status of the SMODE pins. When the programming cable's **PROG** connector is attached, the SMODE pins are pulled high, placing the Rabbit 2000 in the Program Mode. When the programming cable's **PROG** connector is not attached, the SMODE pins are pulled low, causing the Rabbit 2000 to operate in the Run Mode.

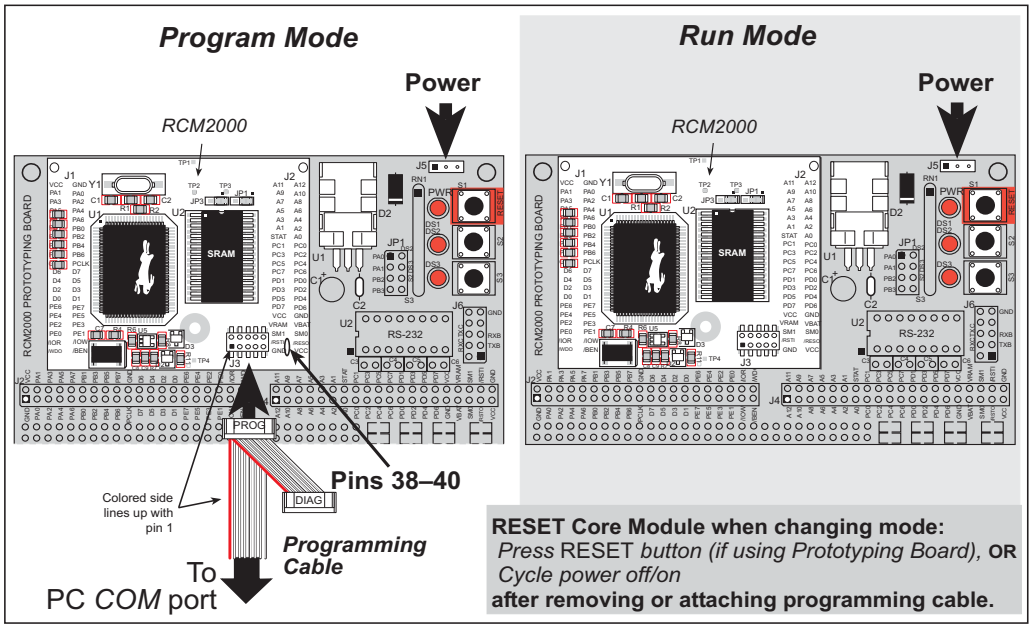

Figure 7. RCM2000 Program Mode and Run Mode Setup

A program "runs" in either mode, but can only be downloaded and debugged when the RCM2000 is in the program mode.

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

- 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 I/O

The RCM2000 was designed to interface with other systems, and so there are no drivers written specifically for this purpose. The general Dynamic C read and write functions allow you to customize the parallel I/O to meet your specific needs. For example, use

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

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

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

to set all the port E bits as outputs.

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

5.2.1 PCLK Output

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

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

PCLK output driven with peripheral clock:

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

PCLK output driven with peripheral clock ÷ 2:

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

PCLK output off (low):

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

PCLK output on (high):

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

APPENDIX A. SPECIFICATIONS

Appendix A provides the specifications for the RCM2000, and describes the conformal coating.

Table A-1 lists the electrical, mechanical, and environmental specifications for the RCM2000.

Table A-1. RCM2000 Specifications

Parameter	RCM2000	RCM2010	RCM2020
Microprocessor	Rabbit 2000® at 25.8 MHz		Rabbit 2000® at 18.432 MHz
Flash EPROM	256K (supports 128K–512K)		
SRAM	512K	128K	
Backup Battery	Connection for user-supplied backup battery (to support RTC and SRAM)		
General-Purpose I/O	40 parallel I/0 lines grouped in five 8-bit ports (and shared with serial ports): • 26 configurable I/O • 8 fixed inputs • 6 fixed outputs		
Additional Inputs	2 startup mode (for master/slave), reset in		
Additional Outputs	Status, clock, watchdog, reset out		
Memory, I/O Interface	13 address lines, 8 data lines, I/O read/write, buffer enable		
Serial Ports	Four 5 V CMOS-compatible ports. Two ports are configurable as clocked ports, one is a dedicated RS-232 programming port.		
Serial Rate	Maximum burst rate = CLK/32 Maximum sustained rate = CLK/64		
Slave Interface	A slave port allows the RCM2000 to be used as an intelligent peripheral device slaved to a master processor, which may either be another Rabbit 2000 or any other type of processor		
Real-Time Clock	Yes		
Timers	Five 8-bit timers cascadable in pairs, one 10-bit timer with 2 match registers that each have an interrupt		
Watchdog/Supervisor	Yes		
Power	4.75 V to 5.25 V DC, 130 mA		4.75 V to 5.25 V DC, 98 mA
Standby Current	10 µA (typical)		
Operating Temperature	–40°C to +85°C		
Humidity	5% to 95%, noncondensing		
Connectors	Two IDC headers 2 × 20, 2 mm pitch		
Board Size	1.90" × 2.30" × 0.55" (48.3 mm × 58.4 mm × 14 mm)		

A.6 Jumper Configurations

Figure A-6 shows the header locations used to configure the various RCM2000 options via jumpers.

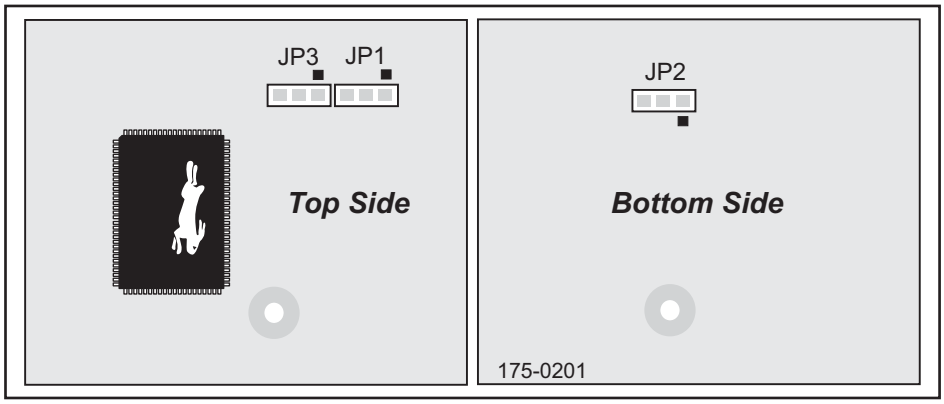

Figure A-6. Location of RCM2000 Configurable Positions

Table A-6 lists the configuration options.

Table A-6. RCM2000 Jumper Configurations

Header	Description	Pins Connected		Factory Default
JP1	SRAM Size	1–2	128K/256K	RCM2010 RCM2020
		2–3	512K	RCM2000
JP2	Flash Memory Size	1–2	128K/256K	×
		2–3	512K	
JP3	Flash Memory Bank Select	1–2	Normal Mode	×
		2–3	Bank Mode	

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

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

Table B-2. Prototyping Board Jumper Settings

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

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

The Prototyping Board provides the user with RCM2000 connection points brought out conveniently to labeled points at headers J2 and J4 on the Prototyping Board. Small to medium circuits can be prototyped using point-to-point wiring with 20 to 30 AWG wire between the prototyping area and the holes at locations J2 and J4. The holes are spaced at 0.1" (2.5 mm), and 40-pin headers or sockets may be installed at J2 and J4. The pinouts for locations J1 and J3, which correspond to J2 and J4, are shown in Figure B-5.

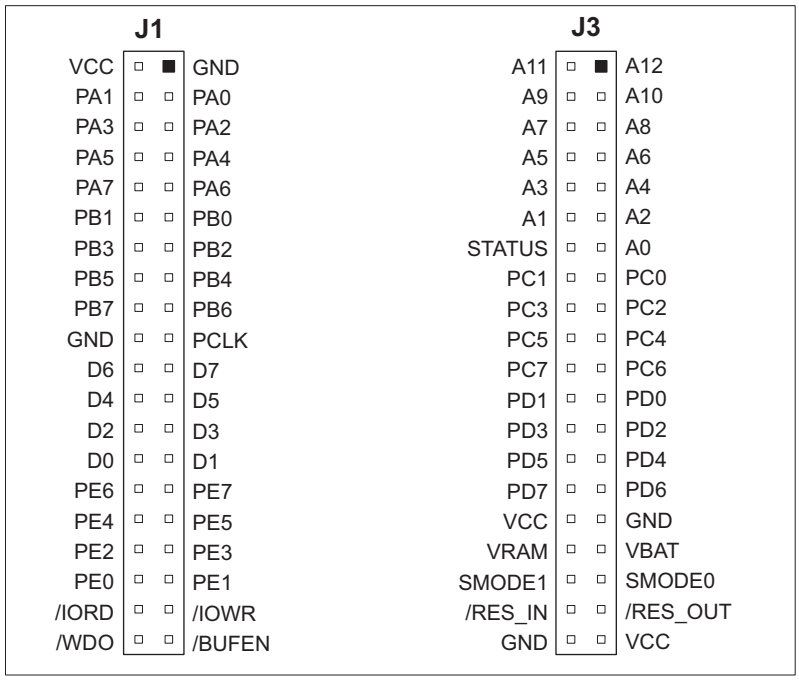

Figure B-5. RCM2000 Prototyping Board Pinout (Top View)

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

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

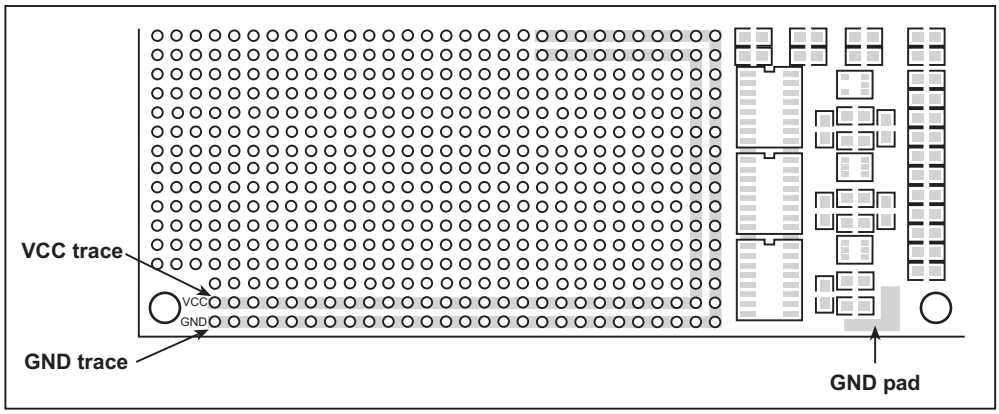

Figure B-6. VCC and GND Traces Along Edge of Prototyping Board

B.4.1 Adding Other Components

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

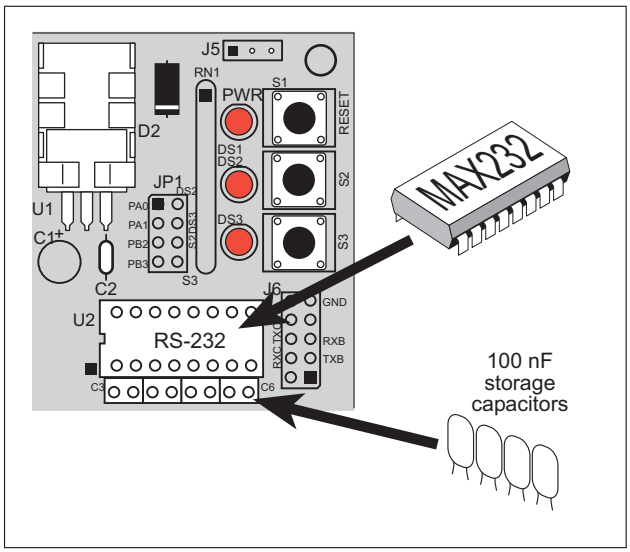

Figure B-7. Location for User-Supplied RS-232 Transceiver and Charge Storage Capacitors

There are two sets of pads that can be used for surface mount prototyping SOIC devices. The silk screen layout separates the rows into six 16-pin devices (three on each side). However, there are pads between the silk screen layouts giving the user two 52-pin (2x26) SOIC layouts with 50 mil pin spacing. There are six sets of pads that can be used for 3- to 6-pin SOT23 packages. There are also 60 sets of pads that can be used for SMT resistors and capacitors in an 0805 SMT package. Each component has every one of its pin pads connected to a hole in which a 30 AWG wire can be soldered (standard wire wrap wire can be soldered in for point-to-point wiring on the Prototyping Board). Because the traces are very thin, carefully determine which set of holes is connected to which surface mount pad.

C.1.3 Power to VRAM Switch

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

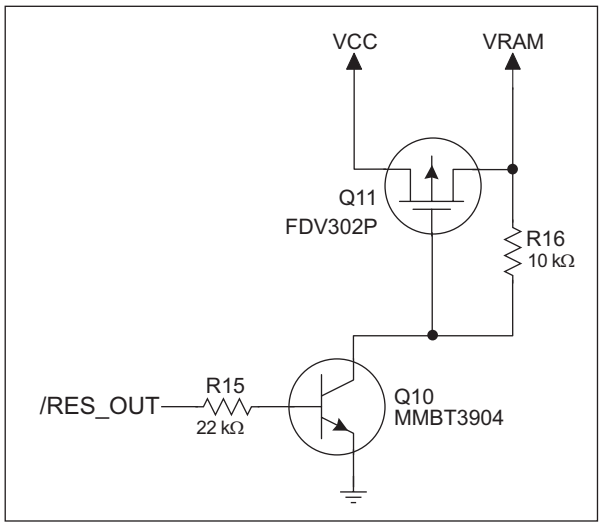

Figure C-3. VRAM Switch

Transistor Q11 is needed to provide a very small voltage drop between Vcc and VRAM (<100 mV, typically 10 mV) so that the processor lines powered by Vcc will not have a significantly different voltage than VRAM.

When the RCM2000 is not resetting (pin 2 on U10 is high), the /RES_OUT line will be high. This turns on Q10, causing its collector to go low. This turns on Q11, allowing VRAM to nearly equal Vcc.

When the RCM2000 is resetting, the /RES_OUT line will go low. This turns off Q10 and Q11, providing an isolation between Vcc and VRAM.

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

C.1.4 Reset Generator

The RCM2000 uses a reset generator, U10, to reset the Rabbit 2000 microprocessor when the voltage drops below the voltage necessary for reliable operation. The reset occurs between 4.50 V and 4.75 V, typically 4.63 V. The RCM2000 has a reset output, pin 37 on header J3, presented to the headers. The reset generator has a reset input, pin 38 on header J3, that can be used to force the RCM2000 to reset.

C.2 Chip Select Circuit

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

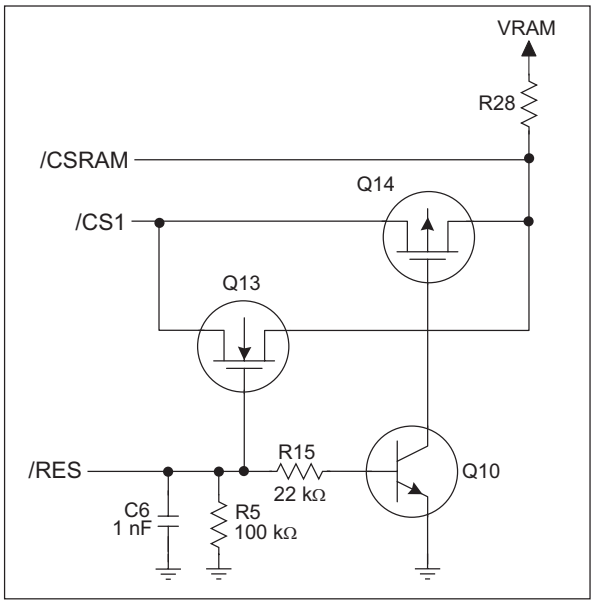

Figure C-4. Chip Select Circuit

The current drain on the battery in a battery-backed circuit must be kept to a minimum. When the RCM2000 is not powered, the battery keeps the SRAM memory contents and the real-time clock (RTC) going. The SRAM has a powerdown mode that greatly reduces power consumption. This powerdown mode is activated by raising the chip select (CS) signal line. Normally the SRAM requires Vcc to operate. However, only 2 V is required for data retention in powerdown mode. Thus, when power is removed from the circuit, the battery voltage needs to be provided to both the SRAM power pin and to the CS signal line. The CS control circuit accomplishes this task for the CS signal line.

In a powered-up condition, the CS control circuit must allow the processor's chip select signal /CS1 to control the SRAM's CS signal /CSRAM. So, with power applied, /CSRAM must be the same signal as /CS1, and with power removed, /CSRAM must be held high (but only needs to be battery voltage high). Q13 and Q14 are MOSFET transistors with opposing polarity. They are both turned on when power is applied to the circuit. They allow the CS signal to pass from the processor to the SRAM so that the processor can periodically access the SRAM. When power is removed from the circuit, the transistors will turn off and isolate /CSRAM from the processor. The isolated /CSRAM line has a 100 kΩ pullup resistor to VRAM (R28). This pullup resistor keeps /CSRAM at the VRAM voltage level (which under no-power conditions is the backup battery's regulated voltage at a little more than 2 V).

Transistors Q13 and Q14 are of opposite polarity so that a rail-to-rail voltages can be passed. When the /CS1 voltage is low, Q13 will conduct. When the /CS1 voltage is high, Q14 will conduct. It takes time for the transistors to turn on, creating a propagation delay. This delay is typically very small, about 10 ns to 15 ns.

T

U