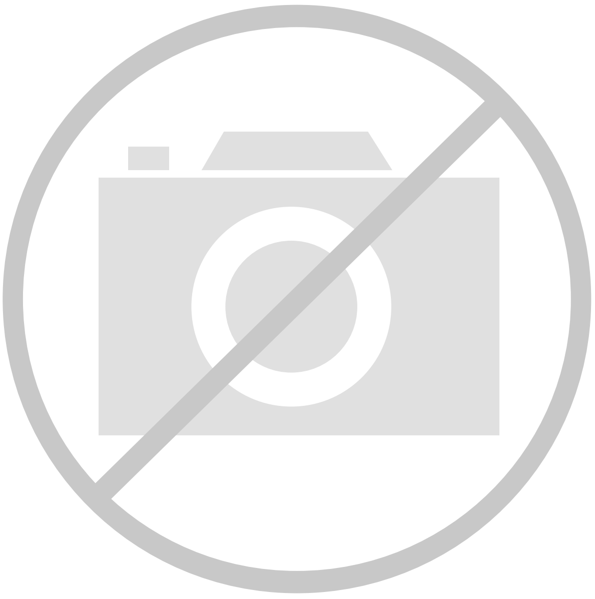
E·XFL

Digi - 101-0383 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	18.432MHz
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/101-0383

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.

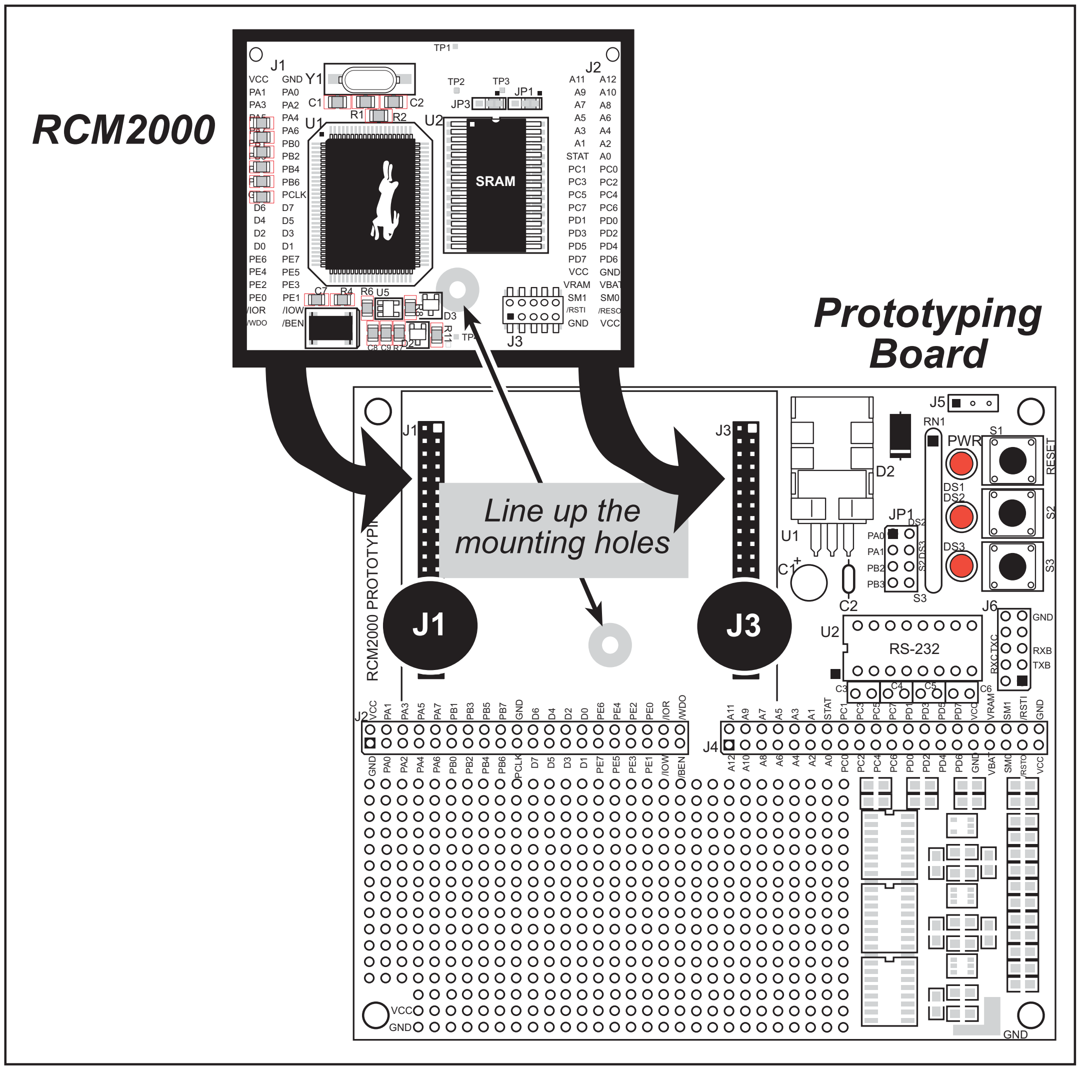


Figure 1. Attaching RCM2000 to Prototyping Board

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.

2.2 Run a Sample Program

Once the RCM2000 is connected as described in the preceding pages, start Dynamic C by double-clicking on the Dynamic C icon on your desktop or in your **Start** menu. Dynamic C uses the serial port specified during installation.

If you are using a USB port to connect your computer to the RCM2000 module, choose **Options > Project Options** and select "Use USB to Serial Converter" under the **Communications** tab, then click **OK**.

Find the file **`PONG.C`**, which is in the Dynamic C **`SAMPLES`** folder. To run the program, open it with the **File** menu (if it is not still open), then compile and run it by pressing **F9** or by selecting **Run** in the **Run** menu. The **STDIO** window will open and will display a small square bouncing around in a box.

2.2.1 Troubleshooting

If Dynamic C cannot find the target system (error message **`"No Rabbit Processor Detected."`**):

- Check that the RCM2000 is powered correctly — the red power LED on the Prototyping Board should be lit when the RCM2000 is mounted on the Prototyping Board and the AC adapter is plugged in.
- Check both ends of the programming cable to ensure that they are firmly plugged into the PC and the **PROG** connector, not the **DIAG** connector, is plugged in to the programming port on the RCM2000 with the marked (colored) edge of the programming cable towards pin 1 of the programming header.
- Ensure that the RCM2000 module is firmly and correctly installed in its connectors on the Prototyping Board.
- Dynamic C uses the COM port specified during installation. Select a different COM port within Dynamic C. From the **Options** menu, select **Project Options**, then select **Communications**. Select another COM port from the list, then click OK. Press **<Ctrl-Y>** to force Dynamic C to recompile the BIOS. If Dynamic C still reports it is unable to locate the target system, repeat the above steps until you locate the COM port used by the programming cable.

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

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

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

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

2.3 Where Do I Go From Here?

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

1. Run all of the sample programs described in Chapter 3 to get a basic familiarity with Dynamic C and the 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

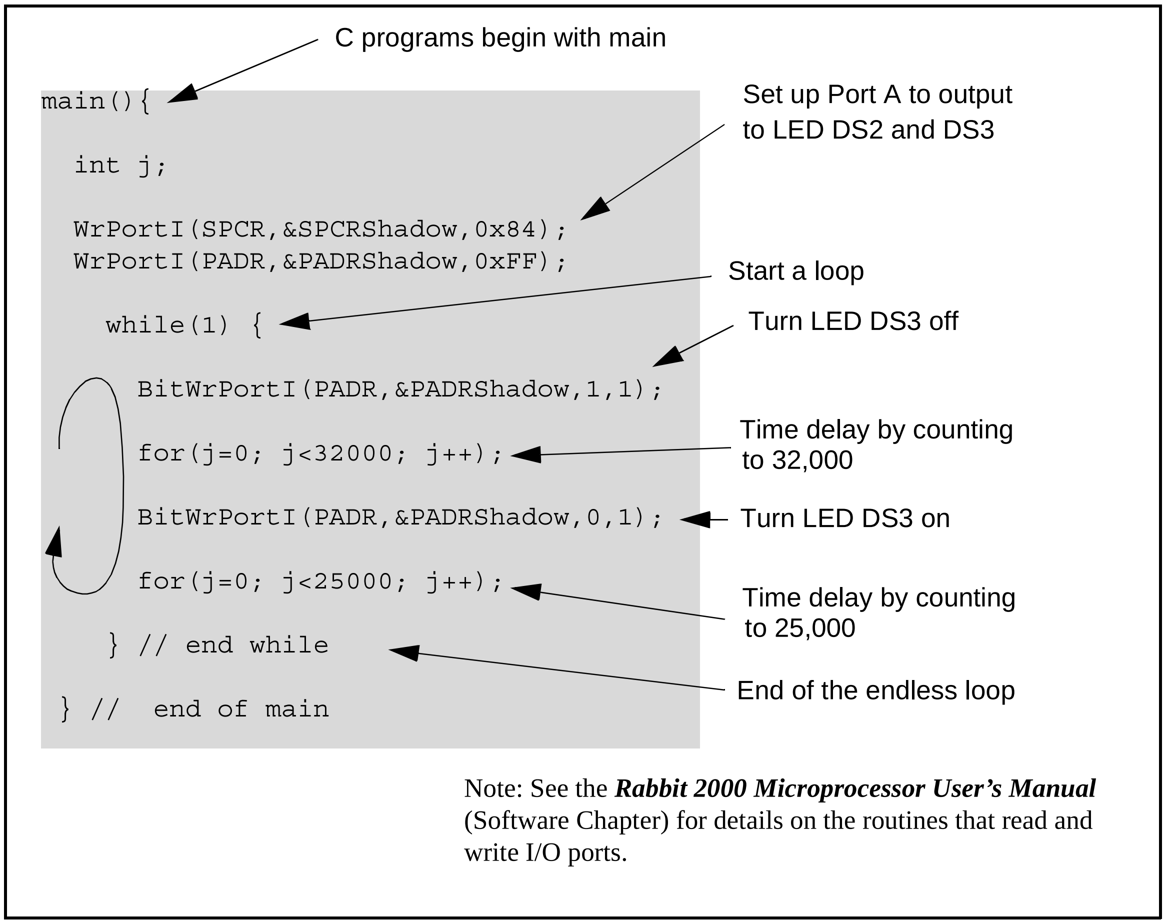
> **NOTE:** If you purchased your RCM2000 through a distributor or through a Rabbit partner, contact the distributor or partner first for technical support.

- 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/.

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.



```
main(){

  int j;

  WrPortI(SPCR,&SPCRShadow,0x84);
  WrPortI(PADR,&PADRShadow,0xFF);

    while(1) {

      BitWrPortI(PADR,&PADRShadow,1,1);

      for(j=0; j<32000; j++);

      BitWrPortI(PADR,&PADRShadow,0,1);

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

    } // end while

 } //  end of main
```

C programs begin with main
Set up Port A to output to LED DS2 and DS3
Start a loop
Turn LED DS3 off
Time delay by counting to 32,000
Turn LED DS3 on
Time delay by counting to 25,000
End of the endless loop

Note: See the ***Rabbit 2000 Microprocessor User's Manual*** (Software Chapter) for details on the routines that read and write I/O ports.

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.

- 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.

- Setting break points. The **F2** key is used to turn on or turn off (toggle) a break point at the cursor position if the program has already been compiled. You can set a break point if the program is paused at a break point. You can also set a break point in a program that is running at full speed. This will cause the program to break if the execution thread hits your break point.
- Watch expressions. A watch expression is a C expression that is evaluated on command in the watch window. An expression is basically any type of C formula that can include operators, variables and function calls, but not statements that require multiple lines such as *for* or *switch*. You can have a list of watch expressions in the watch window. If you are single-stepping, then they are all evaluated on each step. You can also command the watch expression to be evaluated by using the **<Ctrl-U>** command. When a watch expression is evaluated at a break point, it is evaluated as if the statement was at the beginning of the function where you are single-stepping. If your program is running you can also evaluate watch expressions with a **<Ctrl-U>** if your program has a **`runwatch()`** command that is frequently executed. In this case, only expressions involving global variables can be evaluated, and the expression is evaluated as if it were in a separate function with no local variables.

3.1.1.7 Cooperative Multitasking

Cooperative multitasking is a convenient way to perform several different tasks at the same time. An example would be to step a machine through a sequence of steps and at the same time independently carry on a dialog with the operator via a human interface. Cooperative multitasking differs from another approach called preemptive multitasking. Dynamic C supports both types of multitasking. In cooperative multitasking each separate task voluntarily surrenders its compute time when it does not need to perform any more activity immediately. In preemptive multitasking control is forcibly removed from the task via an interrupt.

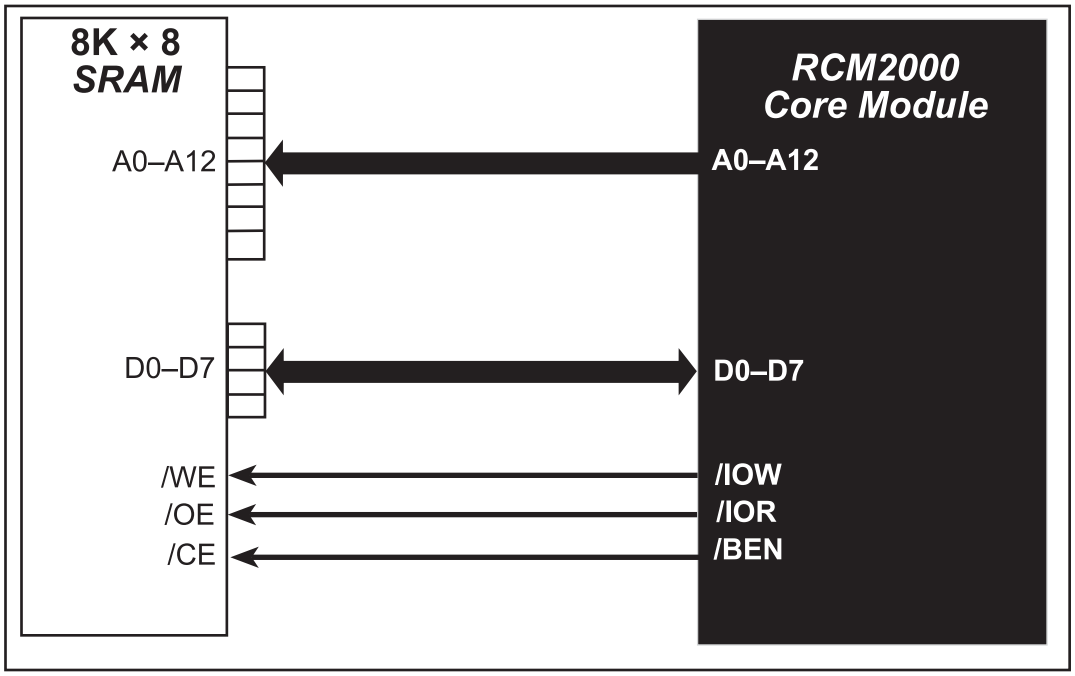
Dynamic C has language extensions to support multitasking. The major C constructs are called *costatements, cofunctions,* and *slicing*. These are described more completely in the ***Dynamic C User's Manual***. The example below, sample program **`FLASHLEDS2.C`**, uses costatements. A costatement is a way to perform a sequence of operations that involve pauses or waits for some external event to take place. A complete description of costatements is in the ***Dynamic C User's Manual***. The **`FLASHLEDS2.C`** sample program has two independent tasks. The first task flashes LED DS2 2.5 times a second. The second task flashes DS3 every 1.5 seconds.

3.1.2 Getting to Know the RCM2000

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

- **`EXTSRAM.C`**—demonstrates the setup and simple addressing to an external SRAM. This program first maps the external SRAM to the I/O Bank 0 register with a maximum of 15 wait states, chip select strobe (which is ignored because of the circuitry), and allows writes. The first 256 bytes of SRAM are cleared and read back. Values are then written to the same area and are read back. The Dynamic C **STDIO** window will indicate if writes and reads did not occur

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



- **`FLASHLED.C`**—repeatedly flashes LED DS3 on the Prototyping Board on and off. LED DS3 is controlled by Parallel Port A bit 1 (PA1).
- **`FLASHLED2.C`**—repeatedly flashes LED DS3 on the Prototyping Board on and off. LED DS3 is controlled by Parallel Port A bit 1 (PA1).

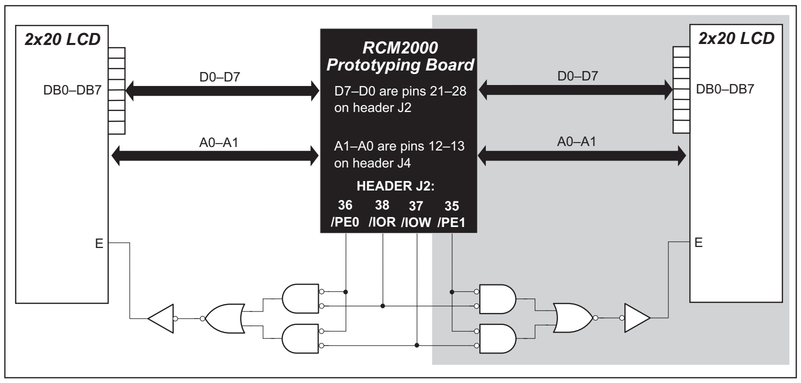
  This sample program also shows the use of the **`runwatch()`** function to allow Dynamic C to update watch expressions while running. The following steps explain how to do this.

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

- **`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


2x20 LCD
DB0–DB7
E
D0–D7
A0–A1
RCM2000 Prototyping Board
D7–D0 are pins 21–28 on header J2
A1–A0 are pins 12–13 on header J4
HEADER J2:
36 /PE0
38 /IOR
37 /IOW
35 /PE1
D0–D7
A0–A1
2x20 LCD
DB0–DB7
E

  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.

Table 2. RCM2000 Pinout Configurations (continued)

Pin		Pin Name	Default Use	Alternate Use	Notes
Header J2	1–13	A[12:0]	Output		Rabbit 2000 address bus
	14	STAT	Output (Status)	Output	
	15	PC0	Output	TXD	
	16	PC1	Input	RXD	
	17	PC2	Output	TXC	
	18	PC3	Input	RXC	
	19	PC4	Output	TXB	
	20	PC5	Input	RXB	
	21	PC6	Output	TXA	
	22	PC7	Input	RXA	
	21	PC6	Output	TXA	Connected to programming port
	22	PC7	Input	RXA	
	23–26	PD[0:3]	Bitwise or parallel programmable I/O, can be driven or open-drain output		16 mA sourcing and sinking current at full AC switching speed
	27	PD4		ATXB output	
	28	PD5		ARXB input	
	29	PD6		ATXA output	
	30	PD7		ARXA input	
	31, 40	GND			
	32, 39	VCC			
	33	VBATR	3 V battery input		
	34	VRAM	2.1 V output		100 kΩ minimum load
	35–36	SMODE0, SMODE1	(0,0)—start executing at address zero		No programming cable attached
			SMODE0 =1, SMODE1 = 1 Cold boot from asynchronous serial port A at 2400 bps (programming cable connected)	(0,1)—cold boot from slave port (1,0)—cold boot from clocked serial port A	With programming cable attached
	37	/RES_OUT	Reset Output		
	38	/RES_IN	Reset Input		

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.

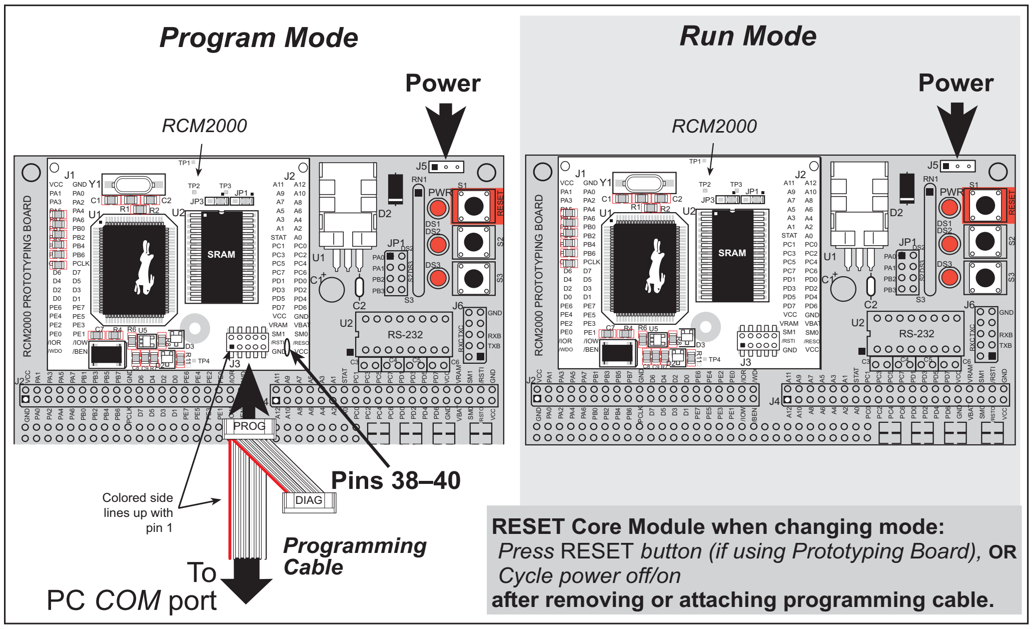

Program Mode
Run Mode
Power
Power
RCM2000
RCM2000
PROG
DIAG
Pins 38–40
Colored side lines up with pin 1
Programming Cable
To PC COM port
SRAM
RS-232
RESET Core Module when changing mode:
Press RESET button (if using Prototyping Board), OR
Cycle power off/on
after removing or attaching programming cable.

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.

5.4 Upgrading Dynamic C

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

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

5.4.1 Extras

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.

Starting with Dynamic C version 9.60, Dynamic C includes the popular µC/OS-II real-time operating system, point-to-point protocol (PPP), FAT file system, RabbitWeb, and other select libraries. Rabbit also offers for purchase the Rabbit Embedded Security Pack featuring the Secure Sockets Layer (SSL) and a specific Advanced Encryption Standard (AES) library.

In addition to the Web-based technical support included at no extra charge, a one-year telephone-based technical support subscription is also available for purchase.

Visit our Web site at www.rabbit.com for further information and complete documentation.

A.2 Bus Loading

You must pay careful attention to bus loading when designing an interface to the RCM2000. This section provides bus loading for external devices.

Table A-2 lists the capacitance for the various RCM2000 I/O ports.

Table A-2. Capacitance of RCM2000 I/O Ports

I/O Ports	Input Capacitance (pF)		Output Capacitance (pF)	
	Typ.	Max.	Typ.	Max.
Parallel Ports A to E	6 pF	12 pF	10 pF	14 pF
Data Lines D0–D7	16 pF	30 pF	24 pF	32 pF
Address Lines A0–A12	—	—	24 pF	32 pF

Table A-3 lists the external capacitive bus loading for the various Rabbit 2000 output ports. Be sure to add the loads for the devices you are using in your custom system and verify that they do not exceed the values in Table A-3.

Table A-3. External Capacitive Bus Loading -40°C to +85°C

Output Port	Clock Speed (MHz)	Maximum External Capacitive Loading (pF)
A[12:1] D[7:1]	25.8	50
A[12:1] D[7:1]	18.4	55 for 90 ns flash 100 for 55 ns flash*
A0 D0	25.8, 18.4	100
PD[3:0]	25.8, 18.4,	100
PA[7:0] PB[7,6] PC[6,4,2,0] PD[7:4] PE[7:0]	25.8, 18.4	90
All data, address, and I/O lines with clock doubler disabled	12.9, 9.2	100

* The RCM2020 operating at 18.4 MHz will typically come with a flash EPROM whose access time is 55 ns. Because of the volatility of the memory market, a 90 ns flash EPROM could be used on the RCM2020.

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.

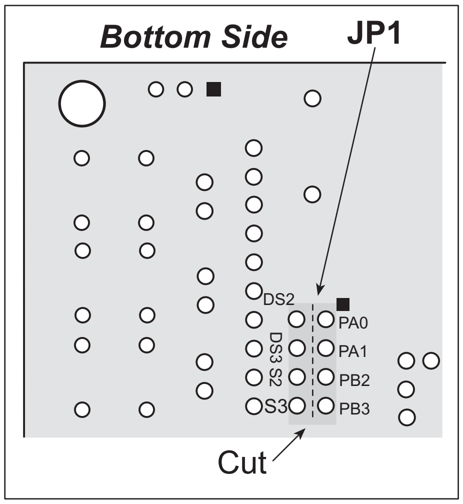


Figure B-4. Where to Cut Traces to Permanently Disable Demonstration Hardware on Prototyping Board

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.

APPENDIX C. POWER MANAGEMENT

Appendix C describes the RCM2000 power circuitry.

C.1 Power Supplies

The RCM2000 requires a regulated 5 V ± 0.25 V DC power source.

An RCM2000 with no loading at the outputs operating at 18.432 MHz typically draws 88 mA, and an RCM2000 operating at 25.8048 MHz typically draws 120 mA. The RCM2000 will consume 13 mA to 15 mA of additional current when the programming cable is used to connect J3 to a PC.

C.1.1 Batteries and External Battery Connections

The RCM2000 does not have a battery, but there is provision for a customer-supplied battery to back up SRAM and keep the internal Rabbit 2000 real-time clock running.

Header J2, shown in Figure C-1, allows access to the external battery. This header makes it possible to connect an external 3 V power supply. This allows the SRAM and the internal Rabbit 2000 real-time clock to retain data with the RCM2000 powered down.

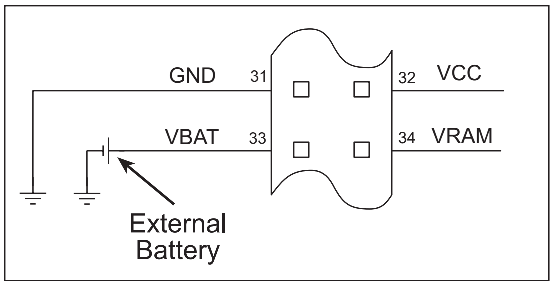


Figure C-1. External Battery Connections at Header J2

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 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:

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

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.

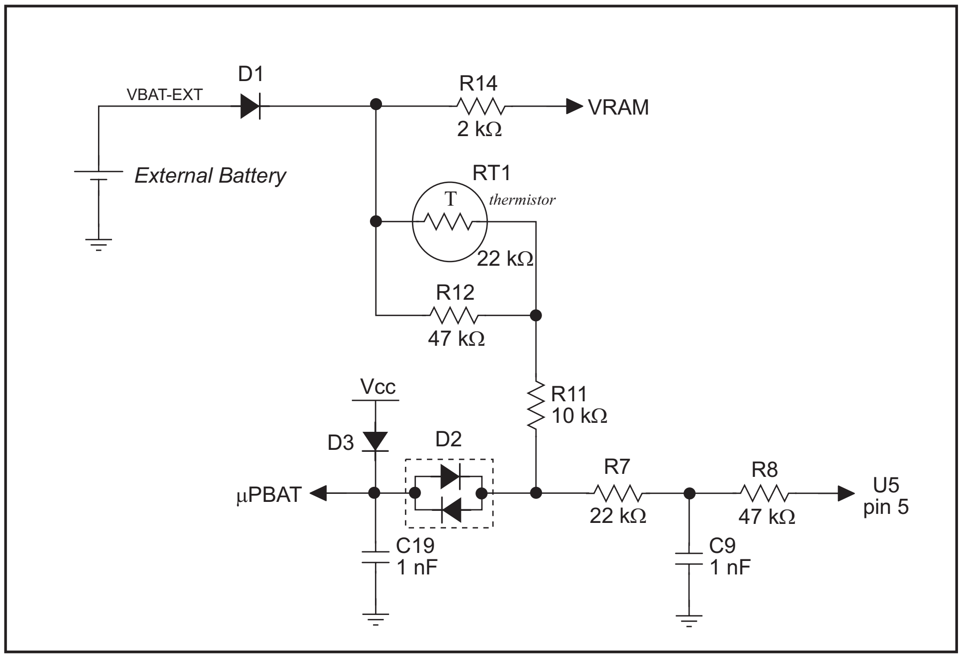


Figure C-2. RCM2000 Battery-Backup Circuit

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.

C.2 Chip Select Circuit

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

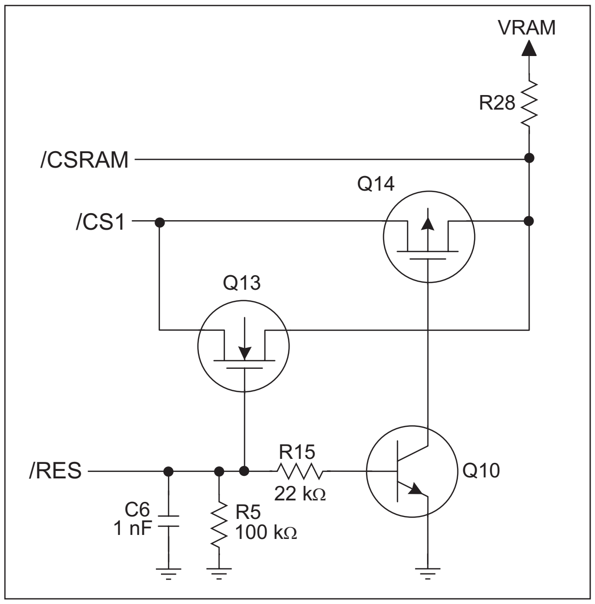


Figure C-4. Chip Select Circuit

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.

D.3 LCD Connections

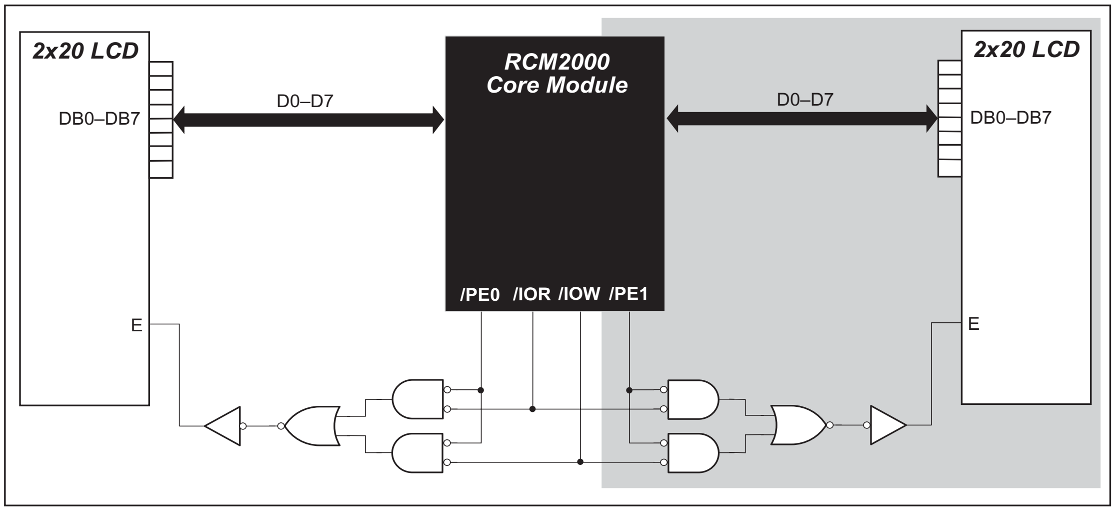


Figure D-4. Sample LCD Connections

Sample Program: **`LCD_DEMO.C`** in **`SAMPLES\COREMODULE`**.

The shaded part of the circuit in Figure D-4 can be used to drive a second LCD, but additional software not included in **`LCD_DEMO.C`** will have to be written.

INDEX

A

additional information
  online documentation 4

B

battery backup
  reset generator 65
battery life 64
bus loading 46

C

clock doubler 32
conformal coating 50

D

Development Kit 3
  RCM2000 3
digital I/O 23
  I/O buffer sourcing and sinking limits 49
  memory interface 28
  SMODE0 29
  SMODE1 29
digital inputs 27
digital outputs 27
dimensions
  Prototyping Board 55
  RCM2000 42
Dynamic C 3, 35
  add-on modules 40
  features 14
  multitasking 15
  Rabbit Embedded Security Pack 40
  sample programs 11
    break point 13
    editing a program 14
    single-stepping 13
    watch expressions 13
  standard features 36
    debugging 37
  telephone-based technical support 40
  upgrades and patches 40
  USB port settings 9

E

EMI
  spectrum spreader feature . 33
exclusion zone 43

F

features 1
  Prototyping Board 54
flash memory addresses
  user blocks 34

H

hardware connections 6
  install RCM2000 on Prototyping Board 6
  power supply 8
  programming cable 7
hardware reset 8

I

I/O buffer sourcing and sinking limits 49

J

jumper configurations 51
  JP1 (SRAM size) 51
  JP2 (flash memory size) 51
  JP3 (flash memory bank select) 34, 51
  jumper locations 51

M

manuals 4

P

PCLK output 38
pinout
  Prototyping Board 59
  RCM2000 24
    alternate configurations 25, 26
power supplies
  +5 V 63
  battery backup 63, 64
  battery life 64
power supply
  connections 8
programming cable
  PROG connector 30
  RCM2000 connections 7
programming port 29
Prototyping Board 53, 54
  adding components 61
  dimensions 55
  features 54
  mounting RCM2000 6
  pinout 59
  power supply 57
  prototyping area 59, 60
  specifications 56

R

Rabbit subsystems 23
RCM2000
  mounting on Prototyping Board 6
reset 8

S

sample circuits 67
  D/A converter 72
  external memory 71
  keypad and LCD connections 69
  LCD connections 70
  RS-232/RS-485 serial communication 68