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	22.1MHz
Flash Size	512KB
RAM Size	512KB
Connector Type	2 IDC Headers 2x20
Size / Dimension	2" x 3.5" (51mm x 89mm)
Operating Temperature	-40°C ~ 85°C
Purchase URL	https://www.e-xfl.com/product-detail/digi-international/101-0436

TABLE OF CONTENTS

2.1.2 Connect Programming Cable

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

Connect the 10-pin connector of the programming cable labeled **PROG** to header J5 on the RCM2100 module as shown in Figure 3 below. Be sure to orient the red edge of the cable towards pin 1 of the connector. (Do not use the **DIAG** connector, which is used for a normal serial connection.)

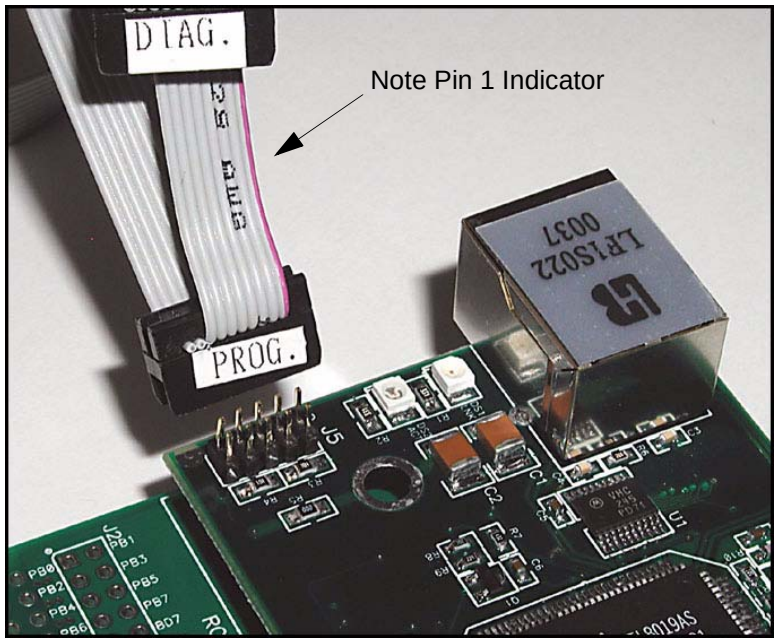


Figure 3. Attaching Programming Cable to the RCM2100

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

Connect the other end of the programming cable to a COM port on your PC. Make a note of the port to which you connect the cable, as Dynamic C needs to have this parameter configured when it is installed.

NOTE: COM 1 is the default port used by Dynamic C.

NOTE: Some PCs now come equipped only with a USB port. It may be possible to use an RS-232/USB converter (Part No. 540-0070) with the programming cable supplied with the RCM2100 Development Kit. Note that not all RS-232/USB converters work with Dynamic C.

3. RUNNING SAMPLE PROGRAMS

To develop and debug programs for the RCM2100 (and for all other Rabbit Semiconductor hardware), you must install and use Dynamic C. Dynamic C is an integrated development system for writing embedded software. It runs on an IBM-compatible PC and is designed for use with Rabbit Semiconductor single-board computers and other single-board computers based on the Rabbit microprocessor. This chapter takes you through the installation of Dynamic C, and then provides a tour of the sample programs for the RCM2100.

3.1 Sample Programs

To help familiarize you with the RCM2100 modules, several sample Dynamic C programs have been included. Loading, executing and studying these programs will give you a solid hands-on overview of the RCM2100's capabilities, as well as a quick start with Dynamic C as an application development tool. These programs are intended to serve as tutorials, but then can also be used as starting points or building blocks for your own applications.

> **NOTE:** It is assumed in this section that you have at least an elementary grasp of ANSI C. If you do not, see the introductory pages of the ***Dynamic C User's Manual*** for a suggested reading list.

Each sample program has comments that describe the purpose and function of the program.

Before running any of these sample programs, make sure that your RCM2100 is connected to the Prototyping Board and to your PC as described in Section 2.1, "Connections." To run a sample program, open it with the **File** menu (if it is not already open), then compile and run it by pressing **F9** or by selecting **Run** in the **Run** menu.

Sample programs are provided in the Dynamic C **SAMPLES** folder. Two folders contain sample programs that illustrate features unique to the RabbitCore RCM2100.

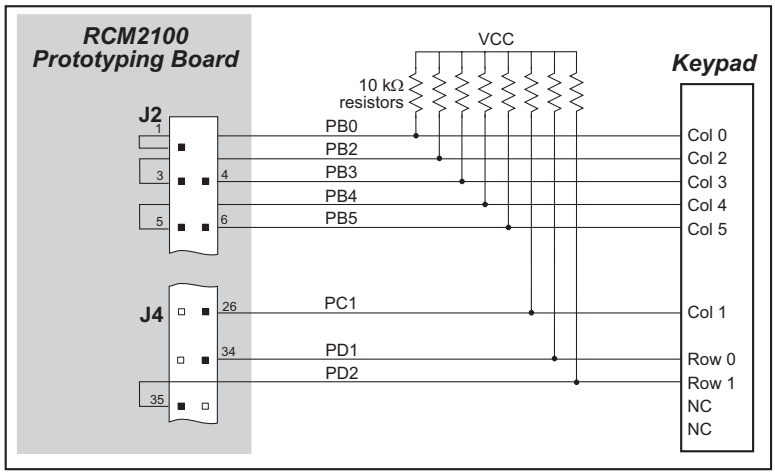
- **RCM2100**—Demonstrates the basic operation and the Ethernet functionality of the RabbitCore RCM2100.
- **TCPIP**—Demonstrates more advanced TCP/IP programming for Rabbit Semiconductor's Ethernet-enabled Rabbit-based boards.

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

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

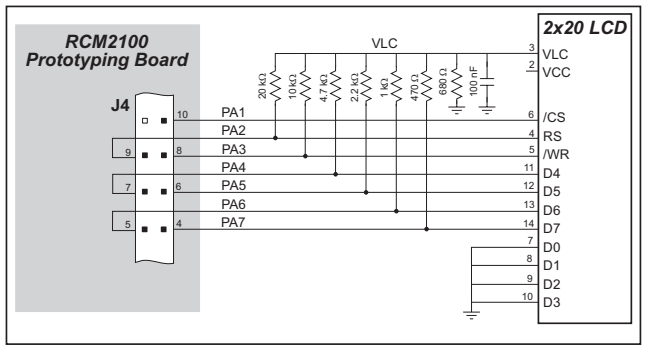
  Connect the keypad to Parallel Ports B, C, and D.

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



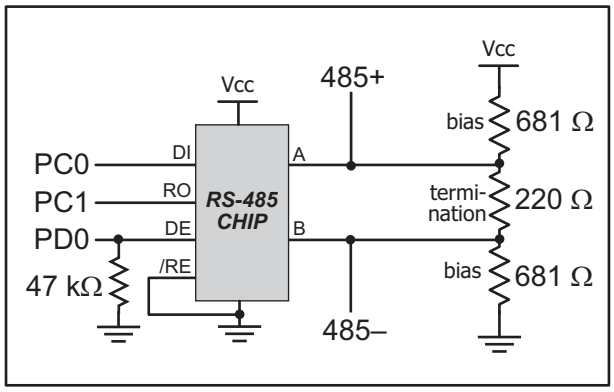
  Connect the LCD to Parallel Port A.

)PA0—backlight (if connected
PA1—LCD /CS
PA2—LCD RS (High = Control, Low = Data) / LCD Contrast 0
PA3—LCD /WR/ LCD Contrast 1
PA4—LCD D4 / LCD Contrast 2
PA5—LCD D5 / LCD Contrast 3
PA6—LCD D6 / LCD Contrast 4
PA7—LCD D7 / LCD Contrast 5



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

Two sample programs, **MASTER2.C** and **SLAVE2.C**, are available to illustrate RS-485 master/slave communication. To run these sample programs, you will need a second Rabbit-based system with RS-485, and you will also have to add an RS-485 transceiver such as the SP483E and bias resistors to the Prototyping Board.



The diagram shows the connections. You will have to connect PC0 and PC1 (Serial Port D) on the Prototyping Board to the RS-485 transceiver, and you will connect PD0 to the RS-485 transceiver to enable or disable the RS-485 transmitter.

The RS-485 connections between the slave and master devices are as follows.

- RS485+ to RS485+
- RS485– to RS485–
- GND to GND

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

3.1.3 Other Sample Programs

Section 6.7 covers how to run the TCP/IP sample programs, which are then described in detail.

3.1.4 Sample Program Descriptions

3.1.4.1 FLASHLED.C

This program is about as simple as a Dynamic C application can get—the equivalent of the traditional "Hello, world!" program found in most basic programming tutorials. If you are familiar with ANSI C, you should have no trouble reading through the source code and understanding it.

The only new element in this sample application should be Dynamic C's handling of the Rabbit microprocessor's parallel ports. The program:

4. Initializes the pins of Port A as outputs.
5. Sets all of the pins of Port A high, turning off the attached LEDs.
6. Starts an endless loop with a **`for(;;)`** expression, and within that loop:
    - Writes a bit to turn bit 1 off, lighting LED DS3;
    - Waits through a delay loop;
    - Writes a bit to turn bit 1 on, turning off the LED;
    - Waits through a second delay loop;

    These steps repeat as long as the program is allowed to run.

You can change the flash rate of the LED by adjusting the loop values in the two **`for`** expressions. The first loop controls the LED's "off" time; the second loop controls its "on" time.

> **NOTE:** Since the variable **`j`** is defined as type **`int`**, the range for **`j`** must be between 0 and 32767. To permit larger values and thus longer delays, change the declaration of **`j`** to **`unsigned int`** or **`long`**.

More Information

See the section on primitive data types, and the entries for the library functions **`WrPortI( )`** and **`BitWrPortI( )`** in the ***Dynamic C User's Manual***.

3.1.4.2 FLASHLEDS.C

In addition to Dynamic C's implementation of C-language programming for embedded systems, it supports assembly-language programming for very efficient processor-level control of the module hardware and program flow. This application is similar to **FLASHLED.C** and **TOGGLELEDS.C**, but uses assembly language for the low-level port control within *cofunctions*, another powerful multitasking tool.

Dynamic C permits the use of assembly language statements within C code. This program creates three functions using assembly language statements, then creates a C cofunction to call two of them. That cofunction is then called within **main()**.

Within each of the C-like functions, the **#asm** and **#endasm** directives are used to indicate the beginning and end of the assembly language statements.

In the function **initialize_ports( )**, port A is initialized to be all outputs while bit 0 of port E is initialized to be an output.

In the function **ledon()**, a 0 is written to the port A bit corresponding to the desired LED (0, which equals DS3, or 1 which equals DS4), turning that LED on. The **ledoff( )** function works exactly the same way except that a 1 is written to the bit, turning the selected LED off.

Finally, in the cofunction **flashled()**, the LED to be flashed, the on time in milliseconds, and the off time in milliseconds are passed as arguments. This function uses an endless **for(;;)** loop to call the **ledon()** and **ledoff()** functions, separated by calls to the wait function **DelayMs()**. This sequence will make the indicated LED flash on and off.

As is proper in C program design, the contents of **main()** are almost trivial. The program first calls **initialize_ports()**, then begins an endless **for(;;)** loop. Within this loop, the program:

1. Calls the library function **hitwd()**, which resets the microprocessor's watchdog timer. (If the watchdog timer is not reset every so often, it will force a hard reset of the system. The purpose is to keep an intermittent program or hardware fault from locking up the system. Normally, this function is taken care of by the virtual driver, but it is called explicitly here).
2. Sets up a costatement which calls two instances of the **flashled()** function, one for each LED. Note that one LED is flashed one second on, one-half second (500 ms) off, while the other is flashed in the reverse pattern.

Note also the **wfd** keyword in the costatement. This keyword (an abbreviation for **waitfordone**, which can also be used) must be used when calling cofunctions. For a complete explanation, see Section 5 and 6 in the ***Dynamic C User's Manual***.

More Information

See the entries for the **hitwd()** and **DelayMs()** functions in the ***Dynamic C User's Manual***, as well as those for the directives **#asm** and **#endasm**. For a complete explana-

Table 2. RCM2100 Pinout Configurations

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

4.3 Serial Programming Cable

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

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

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

4.3.1 Changing Between Program Mode and Run Mode

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

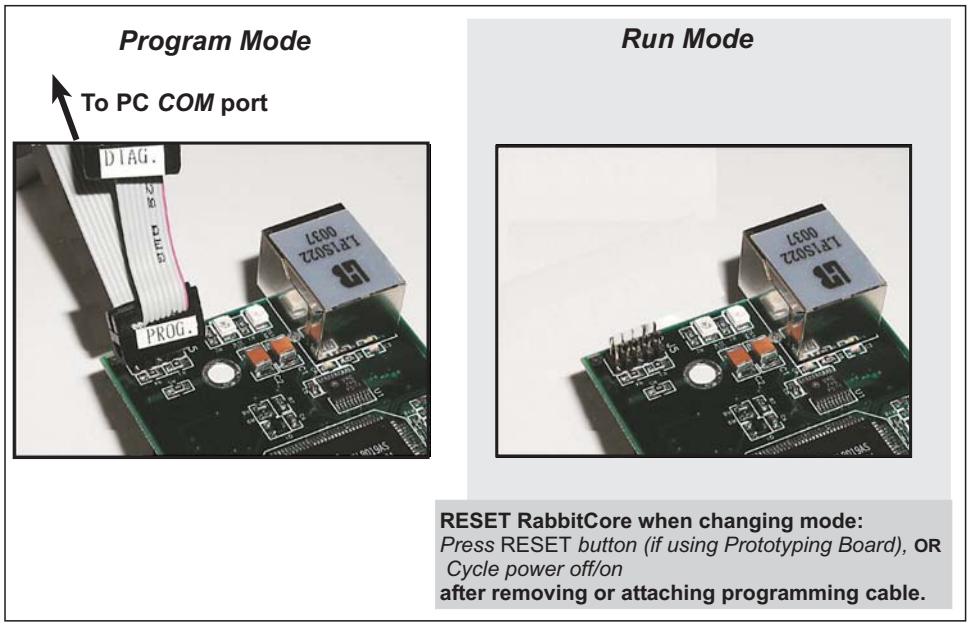


Figure 10. Switching Between Program Mode and Run Mode

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

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

5.1.1 Using Dynamic C

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

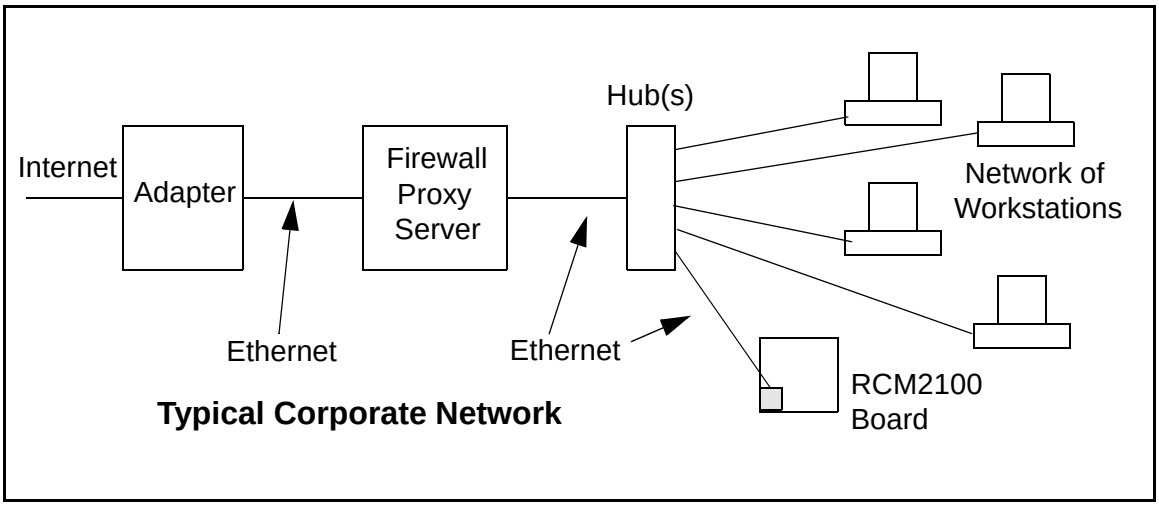
The advantage of working in RAM is to save wear on the flash memory, which is limited to about 100,000 write cycles. The disadvantage is that the code and data might not both fit in RAM.

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

Developing software with Dynamic C is simple. Users can write, compile, and test C and assembly code without leaving the Dynamic C development environment. Debugging occurs while the application runs on the target. Alternatively, users can compile a program to an image file for later loading. Dynamic C runs on PCs under Windows 95, 98, 2000, NT, Me, and XP. Programs can be downloaded at baud rates of up to 460,800 bps after the program compiles.

Dynamic C has a number of standard features.

- Full-feature source and/or assembly-level debugger, no in-circuit emulator required.
- Royalty-free TCP/IP stack with source code and most common protocols.
- Hundreds of functions in source-code libraries and sample programs:
  - ▶ Exceptionally fast support for floating-point arithmetic and transcendental functions.
  - ▶ RS-232 and RS-485 serial communication.
  - ▶ Analog and digital I/O drivers.
  - ▶ I^2C, SPI, GPS, file system.
  - ▶ LCD display and keypad drivers.
- Powerful language extensions for cooperative or preemptive multitasking.
- Loader utility program to load binary images into Rabbit targets in the absence of Dynamic C.
- Provision for customers to create their own source code libraries and augment on-line help by creating “function description” block comments using a special format for library functions.



Typical Corporate Network

If your system administrator can give you an Ethernet cable along with its IP address, the netmask and the gateway address, then you may be able to run the sample programs without having to set up a direct connection between your computer and the RCM2100 board. You will also need the IP address of the nameserver, the name or IP address of your mail server, and your domain name for some of the sample programs.

6.5 Dynamically Assigned Internet Addresses

In many instances, there are no fixed IP addresses. This is the case when, for example, you are assigned an IP address dynamically by your dial-up Internet service provider (ISP) or when you have a device that provides your IP addresses using the Dynamic Host Configuration Protocol (DHCP). The RCM2100 RabbitCore modules can use such IP addresses to send and receive packets on the Internet, but you must take into account that this IP address may only be valid for the duration of the call or for a period of time, and could be a private IP address that is not directly accessible to others on the Internet. These private addresses can be used to perform some Internet tasks such as sending e-mail or browsing the Web, but usually cannot be used to participate in conversations that originate elsewhere on the Internet. If you want to find out what this dynamically assigned IP address is, under Windows XP you can run the **ipconfig** program while you are connected and look at the interface used to connect to the Internet.

Many networks use private IP addresses that are assigned using DHCP. When your computer comes up, and periodically after that, it requests its networking information from a DHCP server. The DHCP server may try to give you the same address each time, but a fixed IP address is usually not guaranteed.

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

6.9 Run the PINGME.C Sample Program

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

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

```
ping 10.10.6.100
```

or by **Start > Run**

and typing the entry

```
ping 10.10.6.100
```

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

6.10 Running More Sample Programs With Direct Connect

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

6.10.1 Sample Program: PINGLED.C

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

Compile & Run Program

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

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

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

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

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

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

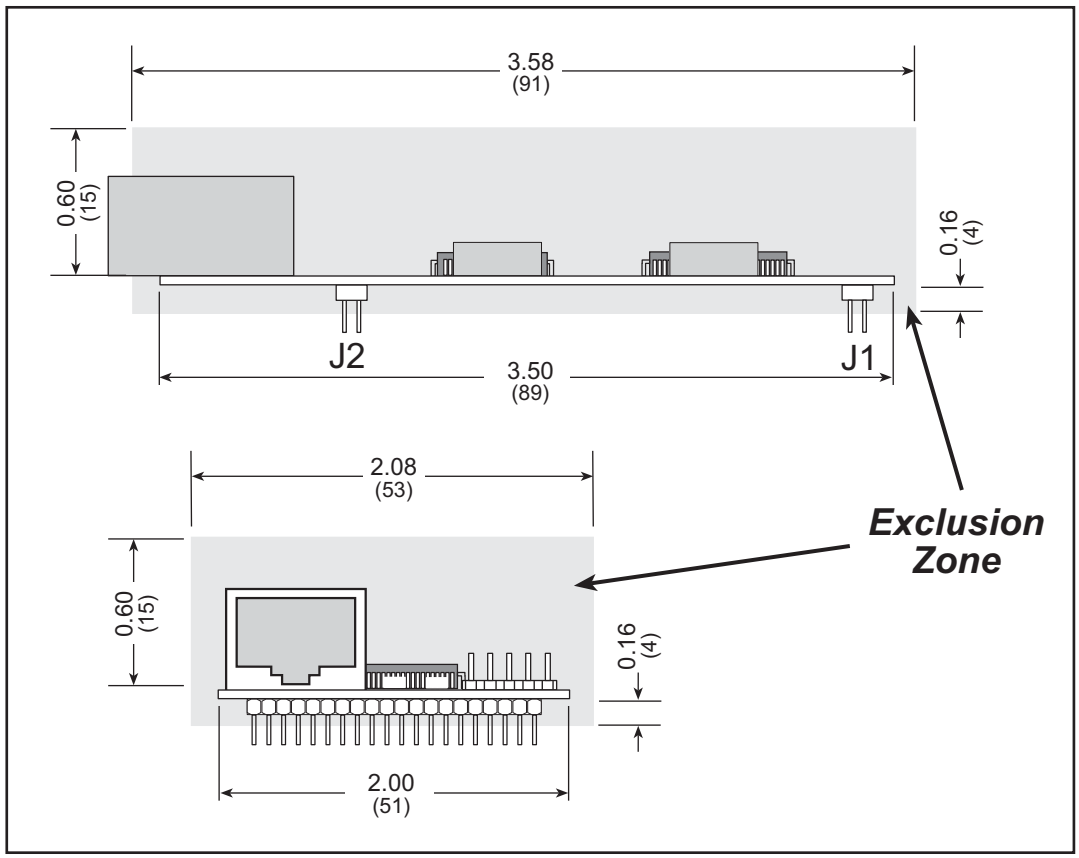


Figure A-2. RCM2100 “Exclusion Zone”

A.1.1 Headers

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

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

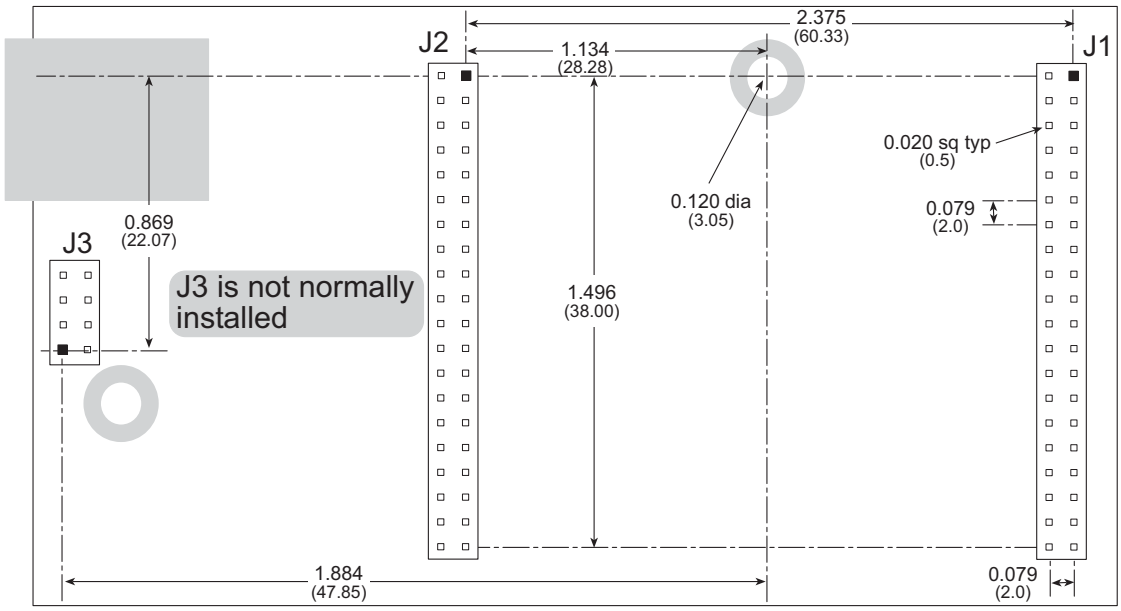


Figure A-3. User Board Footprint for the RCM2100

A.1.2 Physical Mounting

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

A.3 Rabbit 2000 DC Characteristics

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

Table A-4. 5.0 Volt DC Characteristics

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

B.1.2 Prototyping Board Expansion

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

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

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

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

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

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

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

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

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

B.3 Power Supply

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

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

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

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

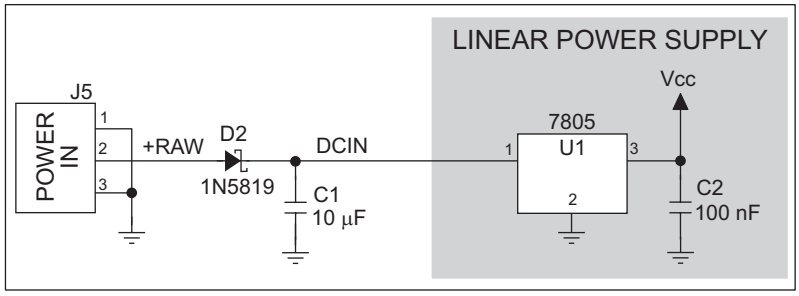


Figure B-3. Prototyping Board Power Supply

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

B.4 Using the Prototyping Board

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

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

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

INDEX