TCP/IP Development Kit

Integrated C Development System

Getting Started Manual

019–0079 • 020331–E

TCP/IP Development Kit Getting Started Manual

Part Number 019-0079 • 020331–E • Printed in U.S.A.

Trademarks

Rabbit 2000 is a trademark of Rabbit Semiconductor.

Dynamic C is a registered trademark of Z-World Inc.

Rabbit Semiconductor

2932 Spafford Street
Davis, California 95616-6800
USA

Telephone: (530) 757-8400
Fax: (530) 757-8402

www.rabbitsemiconductor.com

TABLE OF CONTENTS

1. OVERVIEW

The TCP/IP Development Kit provides a hardware platform based on the Rabbit 2000™ microprocessor, Dynamic C® SE, and the tools necessary to develop a robust 10Base-T Ethernet application.

1.1 Description

The TCP/IP Development Kit includes a TCP/IP Development Board (with a Rabbit 2000™ microprocessor, flash memory, SRAM, Ethernet hardware, serial ports, digital I/O), Dynamic C SE development software with TCP/IP stack and documentation on CD-ROM (not a trial version!), a demonstration board, a power supply, and a serial programming cable.

The TCP/IP Development Board included in the kit allows for immediate evaluation and development of TCP/IP applications using the Rabbit 2000 microprocessor. Executable code can be downloaded into flash memory or SRAM (an optional battery backup board for SRAM and the real-time clock is available). Two communication ports are available—an RS-232 port and a RS-485 port. Other features of the TCP/IP Development Board include four high-current outputs, four digital inputs, seven timers, a real-time battery-backable clock, and a 10Base-T Ethernet interface.

1.1.1 TCP/IP Development Board Features

- 18.432 MHz Rabbit 2000 Processor
- 10Base-T Ethernet interface
- 4 high-current outputs (200 mA @ 40 V DC)
- 4 digital input points (0–5 V DC nominal)
- RS-232 serial port
- RS-485 serial port (may be factory configured to second RS-232)
- 512K flash memory (2 × 256K)
- 128K SRAM
- 7 built-in timers
- Time/date real-time clock (requires battery-backup board, available separately)
- Watchdog timer

1.1.2 Key Benefits

- Ethernet ready—port to an Ethernet chip is done for the Rabbit 2000 chip.
- Cost-effective—no run-time royalties.
- Simplified development—a complete Dynamic C SE software package (with integrated editor, compiler and debugger) is provided. No in-circuit emulator is required.
- A head start—sample programs, including HTTP Web server and SMTP mail client, provide an advanced starting point for development.
- Quick development time—full hardware reference schematics help reduce development efforts.

1.1.3 TCP/IP Capabilities

- Socket-Level TCP (Transmission Control Protocol) provides reliable full-duplex data transmission.
- Socket-Level UDP(User Datagram Protocol)—simple protocol exchanges datagrams without acknowledgements or guaranteed delivery.
- ICMP (Internet Control Message Protocol)—network-layer Internet protocol that reports errors and provides other information relevant to IP packet processing.
- DNS (Domain Name System) client—a distributed Internet directory service that is used mostly to translate between domain names and IP addresses, and to control Internet e-mail delivery.
- DHCP (Dynamic Host Configuration Protocol) client—provides a framework for passing configuration information to hosts on a TCP/IP network. DHCP is based on the Bootstrap Protocol (BOOTP), adding the capability of automatic allocation of reusable network addresses and additional configuration options.
- HTTP (Hypertext Transfer Protocol) server—the protocol used by Web browsers and Web servers to transfer files, such as text and graphic files. Includes facilities for Server Side Includes (SSI) and CGI routines.
- SMTP (Simple Mail Transfer Protocol) client—Internet protocol providing e-mail services.
- FTP (File Transfer Protocol) server and client—application protocol, part of the TCP/IP protocol stack, used for transferring files between network nodes. Server with password support for file transfers between network nodes available on the Rabbit 2000.
- TFTP (Trivial File Transfer Protocol) server and client—simplified version of FTP that allows files to be transferred from one computer to another over a network.
- POP3 (Post Office Protocol) client.
- Serial-to-Telnet gateway.

Additional TCP/IP capabilities are added on an ongoing basis.

1.2 Physical and Electrical Specifications

Table 1 lists the basic specifications for the TCP/IP Development Board.

Table 1. TCP/IP Development Board Specifications

Specification	Data
Power Supply	9 V to 40 V DC
Board Size (with optional backup-battery board)	4.30" × 4.71" × 0.79" (109 mm × 120 mm × 20 mm)
Environmental	–40°C to 70°C, 5–95% humidity, noncondensing

The TCP/IP Development Board has 15 pins on header J7, one RJ-12 jack for RS-232 or RS-485 serial communication, and one RJ-45 Ethernet jack. The pinouts are shown in Figure 1.

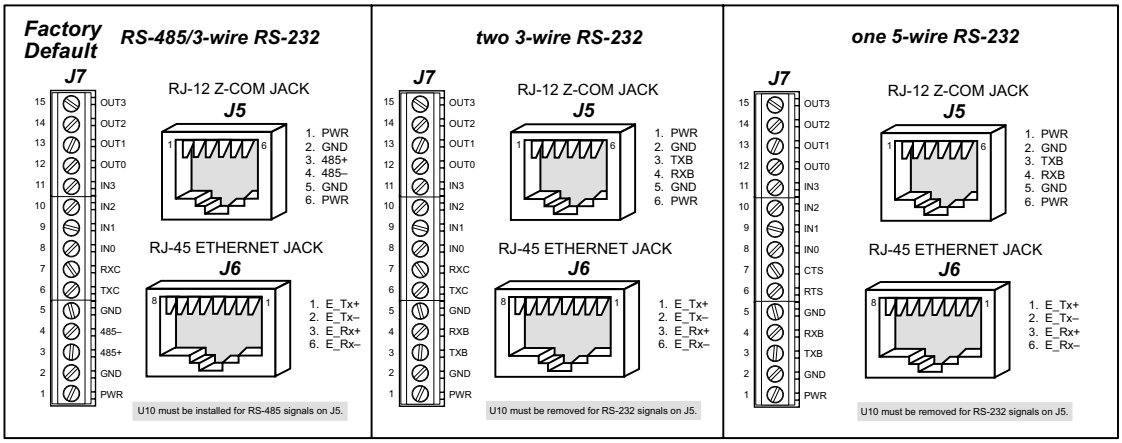


Figure 1. TCP/IP Development Board I/O Pinout

RJ-45 pinouts are sometimes numbered opposite to the way shown in Figure 1. Regardless of the numbering convention followed, the pin positions relative to the spring tab position (located at the bottom of the RJ-45 jack in Figure 1) are always absolute, and the RJ-45 connector will work properly with off-the-shelf Ethernet cables.

1.3 Development Software

The TCP/IP Development Board uses the Dynamic C development environment for rapid creation and debugging of runtime applications. Dynamic C provides a complete development environment with integrated editor, compiler and source-level debugger. It interfaces directly with the target system, eliminating the need for complex and unreliable in-circuit emulators.

Dynamic C must be installed on a Windows workstation with at least one free serial (COM) port for communication with the target system. See Chapter 3., “Installing Dynamic C,” for complete information on installing Dynamic C.

TCP/IP source code is provided in addition to the Dynamic C software on CD-ROM. ICMP, HTTP (includes facilities for SSI, CGI routines, cookies, and basic authentication), SMTP, FTP, and TFTP (client and server) capabilities are provided. Ethernet drivers for the RealTek Ethernet chip are also included. Users can directly write to TCP or UDP sockets to develop custom applications. In addition, extensive sample programs are provided to assist with development. No run-time royalties are required, leading to significant cost savings for OEMs over the life of their application.

1.4 How to Use This Manual

This ***Getting Started*** manual is intended to give users a quick but solid start with the TCP/IP Development Board. It does not contain detailed information on the hardware capabilities or the Dynamic C development environment. Most users will want more detailed information on some or all of these topics in order to put the TCP/IP Development Board to effective use.

> **TIP:** We recommend that anyone not thoroughly familiar with single-board computers at least read through the rest of this manual to gain the necessary familiarity to make use of the more advanced information.

1.4.1 Additional Reference Information

Several higher level reference manuals are provided in HTML and PDF form on the accompanying CD-ROM. Advanced users will find these references valuable in developing systems based on the TCP/IP Development Board:

- ***Dynamic C Premier User's Manual***
- ***Dynamic C TCP/IP User's Manual***
- ***Rabbit 2000 Microprocessor User's Manual***
- ***An Introduction to TCP/IP***

1.4.2 Using Online Documentation

We provide the bulk of our user and reference documentation in two electronic formats, HTML and Adobe PDF. We do this for several reasons.

We believe that providing all users with our complete library of product and reference manuals is a useful convenience. However, printed manuals are expensive to print, stock, and ship. Rather than include and charge for manuals that every user may not want, or provide only product-specific manuals, we chose to provide our complete documentation and reference library in electronic form with every Development Kit and with our Dynamic C development environment.

Finding Online Documents

The online documentation is installed along with Dynamic C, and an icon for the documentation menu is placed on the workstation's desktop. Double-click this icon to reach the menu. If the icon is missing, create a new desktop icon that points to **default.htm** in the **docs** folder, found in the Dynamic C installation folder.

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

Printing Electronic Manuals

We recognize that many users prefer printed manuals for some uses. Users can easily print all or parts of those manuals provided in electronic form. The following guidelines may be helpful:

- Print from the Adobe PDF versions of the files, not the HTML versions.

  NOTE: The most current version of Adobe Acrobat Reader can always be downloaded from Adobe's web site at **http://www.adobe.com**. We recommend that you use version 4.0 or later.

- Print only the sections you will need to refer to often.
- Print manuals overnight, when appropriate, to keep from tying up shared resources during the work day.
- If your printer supports duplex printing, print pages double-sided to save paper and increase convenience.

  NOTE: If you do not have a suitable printer or do not want to print the manual yourself, most retail copy shops (e.g., Kinkos, AlphaGraphics, etc.) will print the manual from the PDF file and bind it for a reasonable charge—about what we would have to charge for a printed and bound manual.

2. HARDWARE CONNECTIONS

Chapter 2 explains how to connect the power supply to the TCP/IP Development Board and how to connect the programming cable to your PC. Once you run a sample program to demonstrate that you have connected everything correctly, you will be ready to go on and finish developing your system.

2.1 Development Kit Contents

The TCP/IP Development Kit contains the following items:

- TCP/IP Development Board with 512K flash memory and 128K SRAM.
- Demonstration Board with pushbutton switches and LEDs, used to demonstrate I/O and TCP/IP capabilities.
- Wire assembly to connect Demonstration Board to TCP/IP Development Board.
- Set of 4 rubber foot pads to position TCP/IP Development Board.
- Wall transformer power supply, 12 V DC, 500 mA (included only with Development Kits sold for the North American market—overseas users will need a power supply compatible with their local mains power).
- 10-pin header to DE9 programming cable with integrated level-matching circuitry.
- *Dynamic C SE* CD-ROM, with complete product documentation on disk.
- This ***Getting Started*** manual.
- Registration card.

2.2 Connections

1. Attach the rubber feet to the bottom corners of the TCP/IP Development Board.

2. Connect the Programming Cable to the TCP/IP Development Board

Turn the Rabbit 2000 TCP/IP Development Board so that the Rabbit 2000 microprocessor is facing up as shown below. Connect the 10-pin **PROG** connector of the programming cable to header J4 onthe TCP/IP Development Board as shown in Figure 2. 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.) Connect the other end of the programming cable to a COM port on your PC.

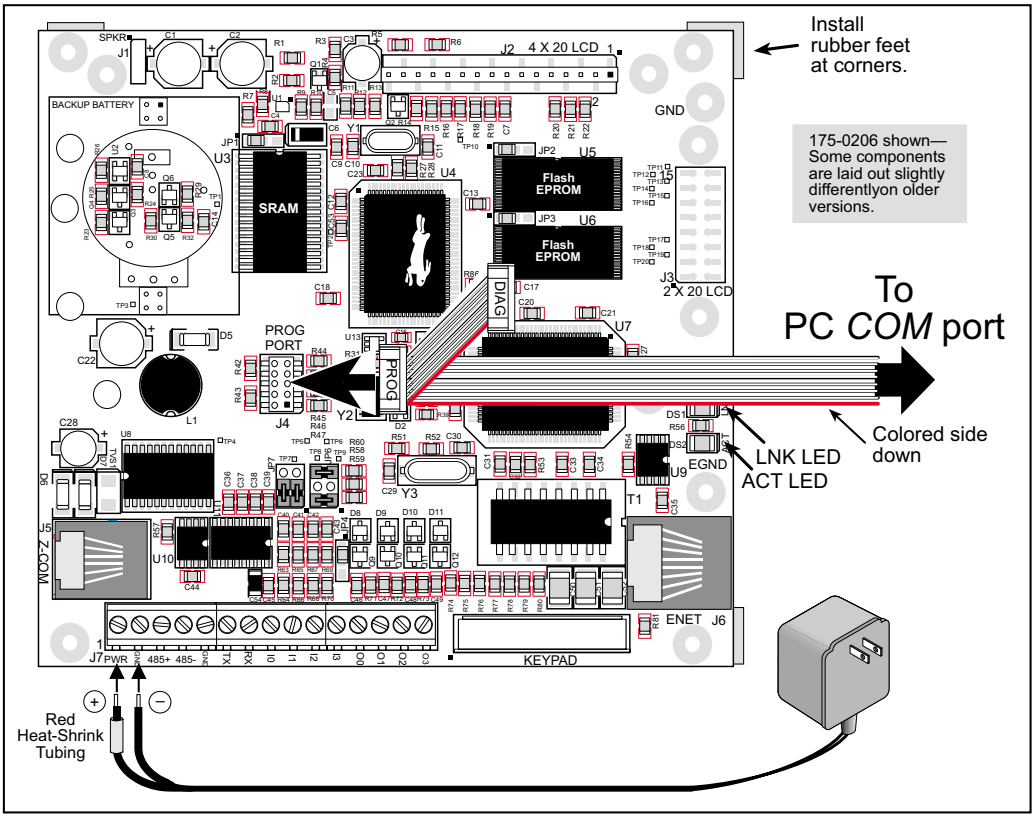


Figure 2. Connecting Power and PC to TCP/IP Development Board

3. Connect Power Supply to TCP/IP Development Board

Connect the positive lead (indicated with red heat-shrink tubing) to the PWR connector on header J7 on the TCP/IP Development Board and connect the negative lead to GND on header J7 as shown here.

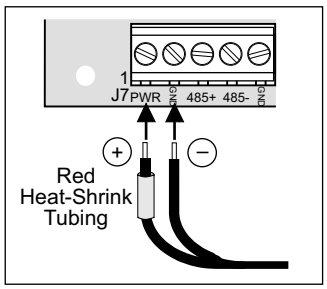


Figure 3. Power Supply Connections

NOTE: Be careful to hook up the positive and negative power leads ***exactly*** as described. Otherwise, the TCP/IP Development board will not function.

4. Apply Power

Plug in the wall transformer. The TCP/IP Development Board is now ready to be used.

NOTE: A hardware RESET is accomplished by unplugging the AC adapter, then plugging it back in.

2.3 Where Do I Go From Here?

We recommend that you proceed to the next chapter and install Dynamic C (if you do not already have it installed), then run the first sample program to verify that the TCP/IP Development Board is set up and functioning correctly.

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

1. Run all of the sample programs described in Chapter 4 to get a basic familiarity with Dynamic C and the TCP/IP Development Board's capabilities.
2. 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 Premier User's Manual***, also in the online documentation set.

2.3.1 Technical Support

> **NOTE:** If you purchased your TCP/IP Development Board through a distributor or through a Z-World or Rabbit Semiconductor partner, contact the distributor or Z-World partner first for technical support.

If there are any problems at this point:

- Check the Z-World/Rabbit Semiconductor Technical Bulletin Board at www.zworld.com/support/bb/.
- Use the Technical Support e-mail form at www.zworld.com/support/support_submit.html.
- Call our Technical Support center:

  Z-World Technical Support, (530) 757-3737.

  Rabbit Semiconductor Technical Support, (530) 757-8400.

3. INSTALLING DYNAMIC C

To develop and debug programs for the TCP/IP Development Board (and for all other Z-World and Rabbit Semiconductor hardware), you must install and use Dynamic C. This chapter takes you through the installation of Dynamic C, and then provides a tour of its major features with respect to the TCP/IP Development Board.

3.1 An Overview of Dynamic C

Dynamic C integrates the following development functions into one program:

- Editing
- Compiling
- Linking
- Loading
- Debugging

In fact, compiling, linking and loading are one function. Dynamic C does not use an In-Circuit Emulator; programs being developed are downloaded to and executed from the "target" system via an enhanced serial-port connection. Program development and debugging take place seamlessly across this connection, greatly speeding system development.

Other features of Dynamic C include:

- Dynamic C has an easy-to-use built-in text editor. Programs can be executed and debugged interactively at the source-code or machine-code level. Pull-down menus and keyboard shortcuts for most commands make Dynamic C easy to use.
- Dynamic C also supports assembly language programming. It is not necessary to leave C or the development system to write assembly language code. C and assembly language may be mixed together.
- Debugging under Dynamic C includes the ability to use **`printf`** commands, watch expressions, breakpoints and other advanced debugging features. Watch expressions can be used to compute C expressions involving the target's program variables or functions. Watch expressions can be evaluated while stopped at a breakpoint or while the target is running its program.
- Dynamic C provides extensions to the C language (such as shared and protected variables, costatements and cofunctions) that support real-world embedded system development. Interrupt service routines may be written in C. Dynamic C supports cooperative and preemptive multi-tasking.
- Dynamic C comes with many function libraries, all in source code. These libraries support real-time programming, machine level I/O, and provide standard string and math functions.
- Dynamic C compiles directly to memory. Functions and libraries are compiled and linked and downloaded on-the-fly. On a fast PC, Dynamic C can load 30,000 bytes of code in 5 seconds at a baud rate of 115,200 bps.

3.2 System Requirements

To install and run Dynamic C, your system must be running one of the following operating systems:

- Windows 95
- Windows 98
- Windows NT
- Windows Me
- Windows 2000

3.2.1 Hardware Requirements

The PC on which you install Dynamic C should have the following hardware:

- A Pentium or later microprocessor
- 32 MB of RAM
- At least 40 MB of free hard drive space
- At least one free COM (serial) port for communication with the target systems
- A CD-ROM drive (for software installation)

3.3 Installing Dynamic C

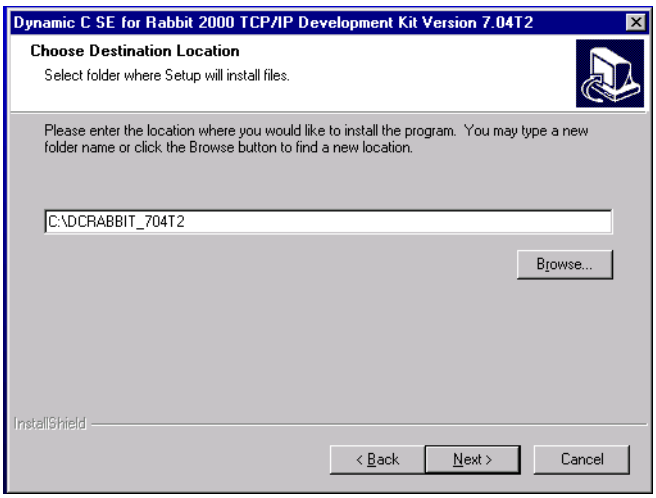
Insert the Dynamic C CD-ROM in the drive on your PC. If autorun is enabled, the CD installation will begin automatically.

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

The installation program will guide you through the installation process. Most steps of the process are self-explanatory and not covered in this section. Selected steps that may be confusing to some users are outlined below. (Some of the installation utility screens may vary slightly from those shown.)

3.3.1 Program & Documentation File Location

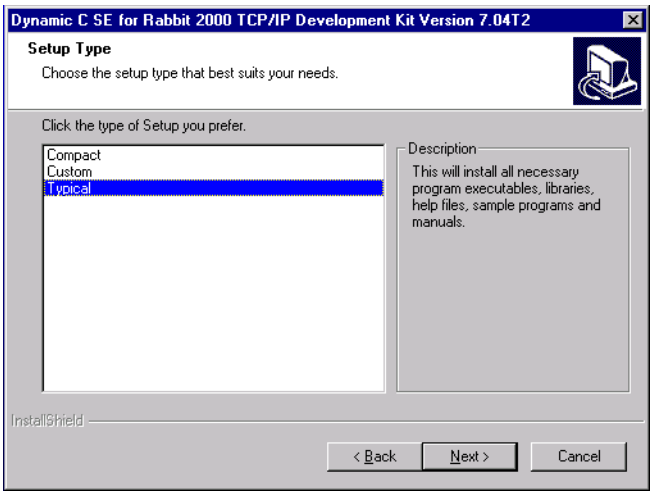
Dynamic C's application, library and documentation files can be installed in any convenient location on your workstation's hard drives.



The default location, as shown in the example above, is in a folder named for the version of Dynamic C, placed in the root folder of the C: drive. If this location is not suitable, enter a different root path before clicking **Next >**. Files are placed in the specified folder, so do not set this location to a drive's root directory.

3.3.2 Installation Type

Dynamic C has two components that can be installed together or separately. One component is Dynamic C itself, with the development environment, support files and libraries. The other component is the documentation library in HTML and PDF formats, which may be left uninstalled to save hard drive space or installed elsewhere (on a separate or network drive, for example).

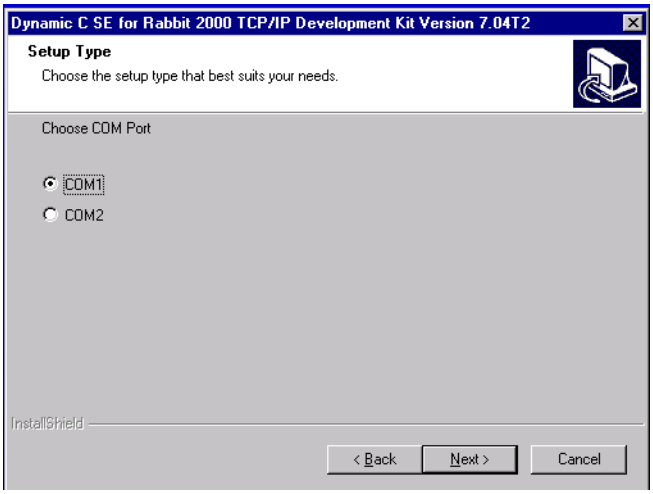


The installation type is selected in the installation menu shown above. The options are:

- **Typical Installation** — Both Dynamic C and the documentation library will be installed in the specified folder (default).
- **Compact Installation** — Only Dynamic C will be installed.
- **Custom Installation** — You will be allowed to choose which components are installed. This choice is useful to install or reinstall just the documentation.

3.3.3 Select COM Port

Dynamic C uses a COM (serial) port to communicate with the target development system. The installation allows you to choose the COM port that will be used.



The default selection, as shown in the example above, is COM1. You may select any available port for Dynamic C's use. If you are not certain which port is available, select COM1. This selection can be changed later within Dynamic C.

NOTE: The installation utility does not check the selected COM port in any way. Specifying a port in use by another device (mouse, modem, etc.) may cause temporary problems when Dynamic C is started.

3.3.4 Desktop Icons

Once your installation is complete, you will have up to three icons on your PC desktop, as shown below.

One icon is for Dynamic C, one opens the documentation menu, and the third is for the Rabbit Field Utility, a tool used to download precompiled software to a target system.

3.4 Starting Dynamic C

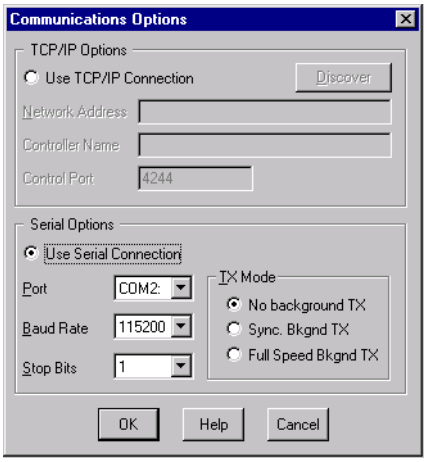
Once the TCP/IP Development Board is set up and connected as described in Chapter 2 and Dynamic C has been installed, start Dynamic C by double-clicking on the Dynamic C icon. Dynamic C should start, then look for the target system on the COM port you specified during installation (by default, COM1). Once detected, Dynamic C should go through a sequence of steps to cold-boot the module and compile the BIOS.

If you receive the message beginning "BIOS successfully compiled …" you are ready to continue with the sample programs in the next chapter.

3.4.1 Communication Error Messages

If you receive the message "No Rabbit Processor Detected," the programming cable may be connected to a different COM port, a connection may be faulty, or the target system may not be powered up. First, check to see that the power supply is connected correctly. If it is, check both ends of the programming cable to ensure that it is firmly plugged into the PC and the TCP/IP Development Board's programming port.

If there are no faults with the hardware, select a different COM port within Dynamic C. From the **Options** menu, select **Communications**. The dialog shown should appear.



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 active COM port.

If Dynamic C appears to compile the BIOS successfully, but you then receive a communication error message, it is possible that your PC cannot handle the 115,200 bps baud rate. Try changing the baud rate to 57,600 bps as follows.

- Locate the **Serial Options** dialog in the Dynamic C **Options > Communications** menu. Change the baud rate to 57,600 bps.

If you are using Dynamic C version 7.04 or earlier, modify the BIOS source code as follows. Skip these three steps if your version of Dynamic C is 7.05 or later.

1. Open the BIOS source code file named **RABBITBIOS.C**, which can be found in the **BIOS** directory.
2. Change the line

```
#define USE115KBAUD 1   // set to 0 to use 57600 baud
```

   to read as follows.

```
#define USE115KBAUD 0   // set to 0 to use 57600 baud
```

3. Save the changes using **File > Save**.

Now press **<Ctrl-Y>**. You should receive the "BIOS successfully compiled …" message indicating that the target is now ready to compile a program. You should then continue with the sample programs in the next chapter.

3.5 PONG.C

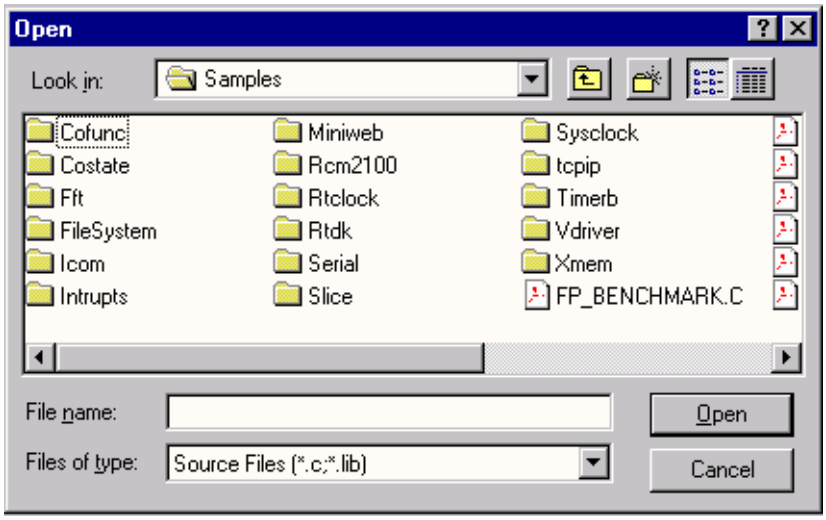
You are now ready to test your set-up by running a sample program.

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), compile it using the **Compile** menu, and then run it by selecting **Run** in the **Run** menu. The **STDIO** window will open and will display a small square bouncing around in a box.

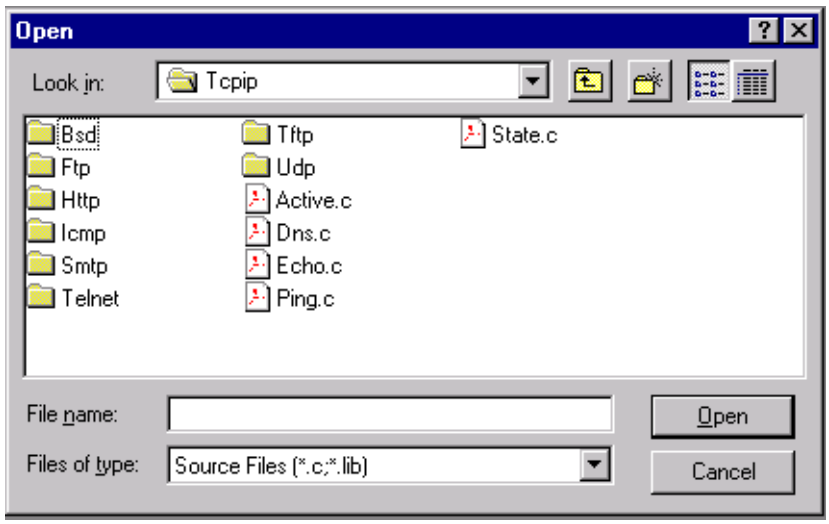
This program does not test the TCP/IP part of the board, but does ensure that the board is functional. The sample program in the next chapter tests the TCP/IP portion of the board.

3.6 Sample Programs

Other sample programs are provided in the Dynamic C **Samples** folder, which is shown below.



The various folders contain specific sample programs that illustrate the use of the corresponding Dynamic C libraries. The sample program **PONG.C** demonstrates the output to the **STDIO** window. The **ICOM** and **TCPIP** folders provide sample programs specific to the TCP/IP Development Board. Let's take a look at the **TCPIP** folder.



The various folders contain sample programs to illustrate the topics associated with the TCP/IP Development Board. See ***An Introduction to TCP/IP*** for more information on these topics.

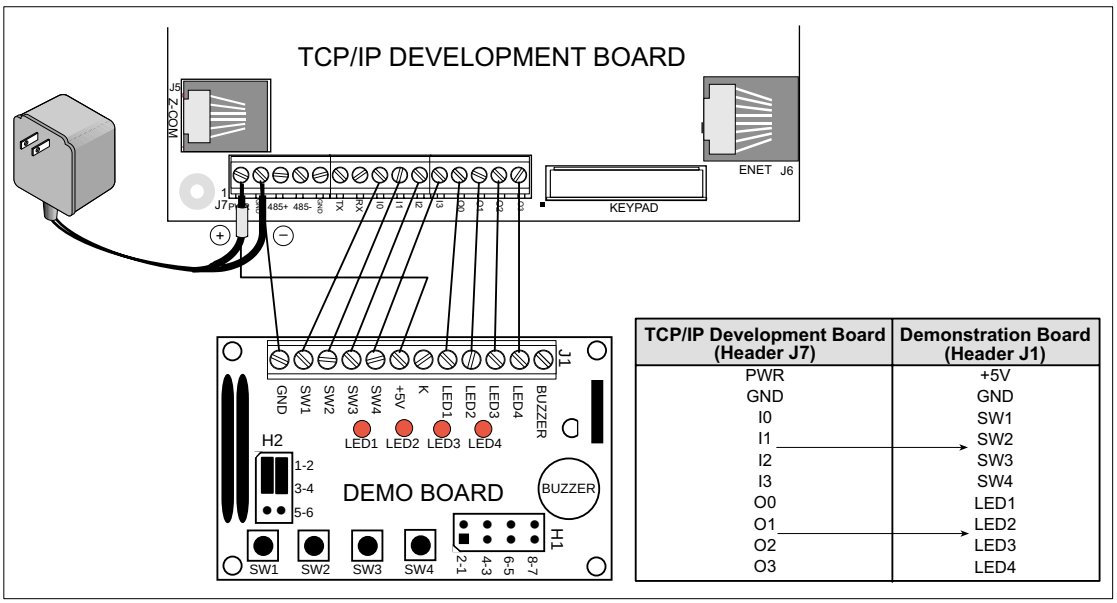
Each sample program has comments that describe the purpose and function of the program.

3.6.1 Running Sample Program DEMOBRD1.C

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

Before running this sample program, you will have to connect the Demonstration Board from the TCP/IP Development Kit to the TCP/IP Development Board. Proceed as follows.

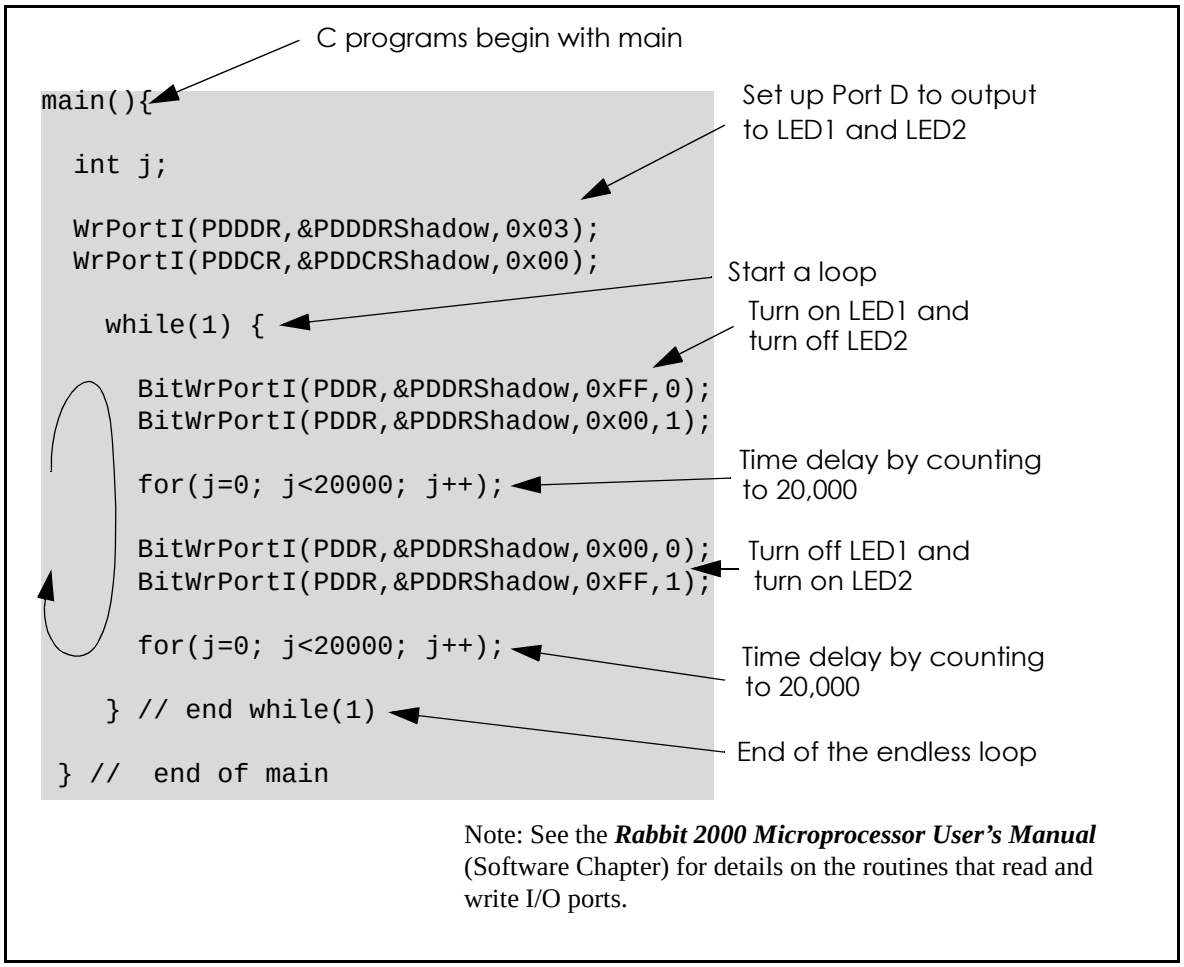
1. Use the wires included in the TCP/IP Development Kit to connect header J1 on the Demonstration Board to header J7 on the TCP/IP Development Board. The connections are shown in Figure 4.
2. Make sure that your TCP/IP Development Board is connected to your PC and that the power supply is connected to the TCP/IP Development Board and plugged in as described in Section 2.2.



TCP/IP Development Board (Header J7)	Demonstration Board (Header J1)
PWR	+5V
GND	GND
I0	SW1
I1	SW2
I2	SW3
I3	SW4
O0	LED1
O1	LED2
O2	LED3
O3	LED4

Figure 4. Connections Between TCP/IP Development Board and Demonstration Board

Now, open the file **DEMOBRD1.C**, which is in the **Samples/ICOM** folder. The program will appear in a window, as shown in Figure 5 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.



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

Figure 5. Sample Program DEMOBRD1.C

To run the program **DEMOBRD1.C**, load it with the **File** menu, compile it using the **Compile** menu, and then run it by selecting **Run** in the **Run** menu. LED1 and LED2 on the Demonstration Board should start going on and off 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 wall transformer is either not connected or is not plugged in.

- The programming cable must be connected to the TCP/IP Development Board. (The colored wire on the programming cable is closest to pin 1 on header J4 on the TCP/IP Development Board, as shown in Figure 1.) 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.

3.6.2 Single-Stepping

Compile or re-compile **DEMOBRD1.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.

3.6.2.1 Watch Expression

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.

3.6.2.2 Break Point

Move the cursor to the start of the statement:

```
for(j=0; j<20000; j++);
```

To set a break point on this statement, type **F2** or select **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. The break point will start flashing both red and green colors. Note that LED1 on the Demonstration Board is now solidly turned on. This is because we have passed the statement turning on LED1.

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 LED1 should be flashing again because the program is running at full speed.

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.

3.6.2.3 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 20000 in the **for (..** statement to 10000. Then use the **F9** key to recompile and run the program. The LEDs will start flashing, but it will flash much faster than before because you have changed the loop counter terminal value from 20000 to 10000.

3.6.2.4 Watching Variables Dynamically

Go back to edit mode (select edit) and load the program **DEMOBRD2.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 **DEMOBRD2.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**.

3.6.2.5 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.
- 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 **`rundwatch()`** 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.

3.6.3 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 a different 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 Premier User's Manual***. The example below, sample program **`DEMOBRD3.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 Premier User's Manual***. The **`DEMOBRD3.C`** sample program has two independent tasks. The first task flashes LED2 once a second. The second task uses button SW1 on the Demonstration Board to toggle the logical value of a virtual switch, **`vswitch`**, and flash LED1 each time the button is pressed. This task also debounces button SW1.

Note that the Demonstration Board has to be connected to the TCP/IP Development Board as described in Section 3.6.1 to be able to run **`DEMOBRD3.C`**.

```
main() {
    int vswitch;        // state of virtual switch controlled by button S1

    WrPortI(PDDDR, &PDDDRShadow, 0x03); // set port D bits 0-1 as outputs
    WrPortI(PDDCR, &PDDCRShadow, 0x00); // set port D to not open drain mode
    vswitch = 0;                        // initialize virtual switch as off

(1) while (1) {                         // endless loop

        // First task will flash LED4 for 200 ms once per second.

(2)     costate {
            BitWrPortI(PDDR, &PDDRShadow, 0xFF, 1);    // turn LED on
(3)         waitfor(DelayMs(200));                     // wait 200 ms
            BitWrPortI(PDDR, &PDDRShadow, 0x00, 1);    // turn LED off
            waitfor(DelayMs(800));                     // wait 800 ms
(4)     }

        // Second task - debounce SW1 and toggle vswitch

        costate {
(5)         if (!BitRdPortI(PDDR, 2)) abort; // if button not down skip out
            waitfor(DelayMs(50));            // wait 50 ms
            if(!BitRdPortI(PDDR, 2))  abort; // if button not still down exit

            vswitch = !vswitch;       // toggle since button was down 50 ms

            while (1) {
                waitfor(!BitRdPortI(PDDR, 2));  // wait for button to go up
                waitfor(DelayMs(200));          // wait additional 200 ms
                if (!BitRdPortI(PDDR, 2))
                    break;    // if button still up break out of while loop
            }
        }                     // end of costate

        // make LED1 agree with vswitch

(6)     BitWrPortI(PDDR, &PDDRShadow, vswitch, 0);

(7) }   // end of while loop
}       // end of main
```

The numbers in the left margin are reference indicators, and are not a part of the code. Load and run the program. Note that LED2 flashes once per second. Push button SW1 several times and note how LED1 is toggled.

The flashing of LED2 is performed by the costatement starting at the line marked (2). Costatements need to be executed regularly, often at least every 25 ms. To accomplish this, the costatements are enclosed in a **while** 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 while loop starts at (1) and ends at (7).

The statement at (3) waits for a time delay, in this case 200 ms. The costatement is being executed on each pass through the big loop. When a **waitfor** 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 (4), 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. Once 200 ms has passed, the statement after the waitfor is executed. 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 Premier User's Manual*** for more details.

The second costatement in the program debounces the switch and maintains the variable **vswitch**. Debouncing is performed by making sure that the switch is either on or off for a long enough period of time to ensure that high-frequency electrical hash generated when the switch contacts open or close does not affect the state of the switch. The **abort** statement is illustrated at (5). If executed, the internal statement pointer is set back to the first statement within the costatement, and a jump to the closing brace of the costatement is made.

At (6) a use for a shadow register is illustrated. 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. In this case a test is made to see the state of the LED and make it agree with the state of vswitch. This test is not strictly necessary, the output register could be set every time to agree with vswitch, but it is placed here to illustrate the concept of a shadow register.

To illustrate the use of snooping, use the watch window to observe **vswitch** while the program is running. Add the variable **vswitch** to the list of watch expressions. Then toggle **vswitch** and the LED. Then type **<Ctrl-U>** to observe **vswitch** again.

3.6.4 Advantages of Cooperative Multitasking

Cooperative multitasking, as implemented with language extensions, has the advantage of being intuitive. Unlike preemptive multitasking, variables can be shared between different tasks without having to take elaborate precautions. Sharing variables between tasks is the greatest cause of bugs in programs that use preemptive multitasking. It might seem that the biggest problem would be response time because of the big loop time becoming long as the program grows. Our solution for that is a device caused slicing that is further described in the ***Dynamic C Premier User's Manual***.

4. USING THE TCP/IP FEATURES

Chapter 4 provides an introduction to using the TCP/IP features on your TCP/IP Development Board.

4.1 TCP/IP Connections

Before proceeding you will need to have the following items.

- If you don't have Ethernet access, you will need at least a 10Base-T Ethernet card (available from your favorite computer supplier) installed in a PC.
- Two RJ-45 straight through Ethernet cables and a hub, or an RJ-45 crossover Ethernet cable.

The Ethernet cables and Ethernet hub are available from Rabbit Semiconductor in a TCP/IP tool kit. More information is available at www.rabbitsemiconductor.com.

1. Connect the AC adapter and the programming cable as shown in Section 2.2, "Connections."
2. Ethernet Connections

If you do not have access to an Ethernet network, use a crossover Ethernet cable to connect the TCP/IP Development Board to a PC that at least has a 10Base-T Ethernet card.

If you have access to an Ethernet network, use a straight through Ethernet cable to establish an Ethernet connection to the TCP/IP Development Board from an Ethernet hub or switch. These connections are shown in Figure 6.

The PC running Dynamic C through the serial programming port on the TCP/IP Development Board does not need to be the PC with the Ethernet card.

3. Apply Power

Plug in the AC adapter. The TCP/IP Development Board is now ready to be used.

> **NOTE:** A hardware RESET is accomplished by unplugging the AC adapter, then plugging it back in.

When working with the TCP/IP Development Board, the green **LNK** light is on when a program is running and the board is properly connected either to an Ethernet hub or to an active Ethernet card. The red **ACT** light flashes each time a packet is received.

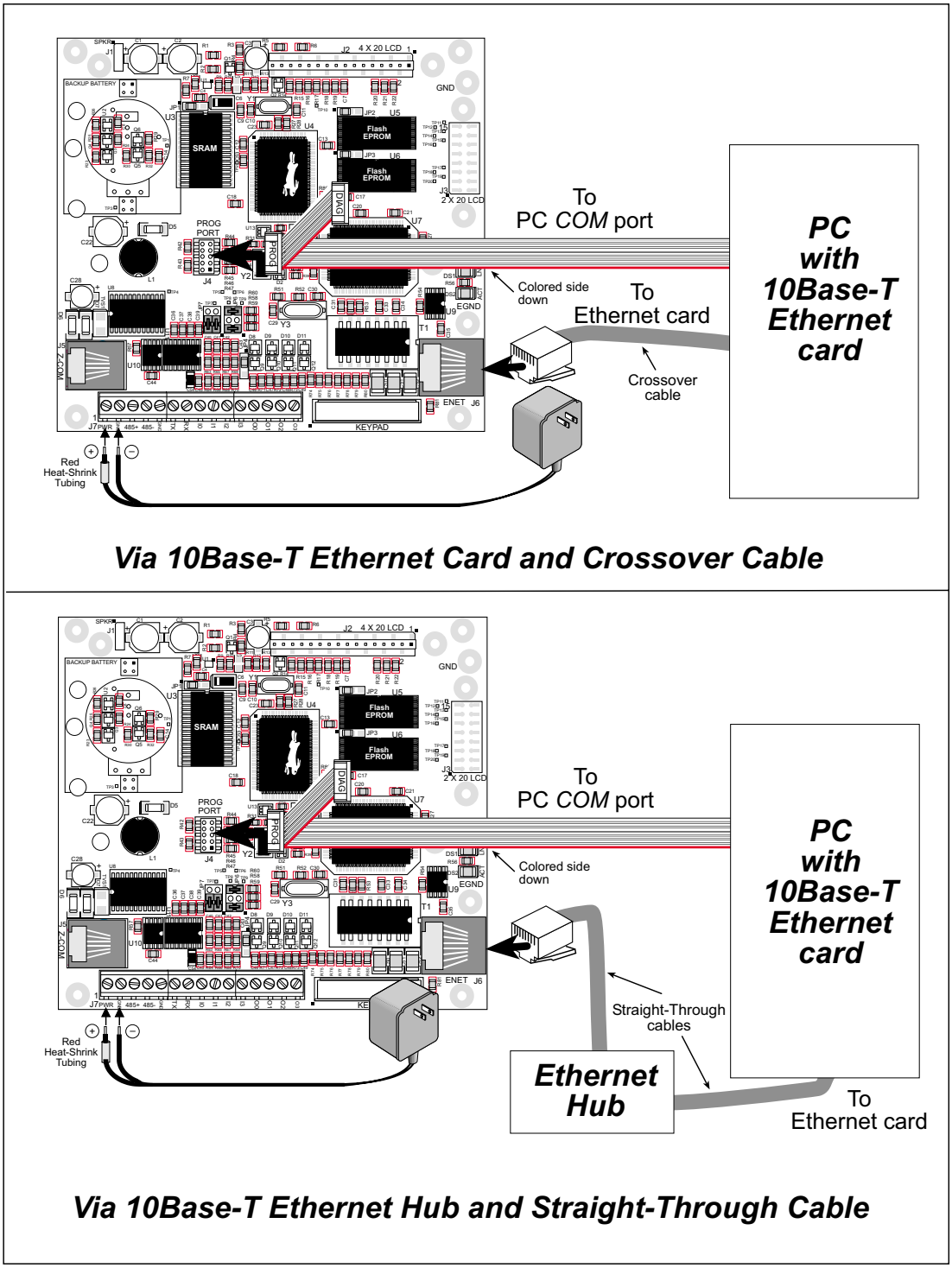


Figure 6. Ethernet Connections

4.2 Running TCP/IP Sample Programs

We have provided a number of sample programs demonstrating various uses of TCP/IP for networking embedded systems. These programs require that the user connect his PC and the TCP/IP Development Board together on the same network. This network can be a local private network (preferred for initial experimentation and debugging), or a connection via the Internet.

Obtaining IP addresses to interact over an existing, operating, network can involve a number of complications, and must usually be done with cooperation from your ISP and/or network systems administrator (if your company has one). For this reason, it is suggested that the user begin instead by using a direct connection between a PC and the TCP/IP Development Board using an Ethernet crossover cable or a simple arrangement with a hub. (A crossover cable should not be confused with regular straight through cables.) The hub and a wide variety of cables can also be purchased from a local computer store.

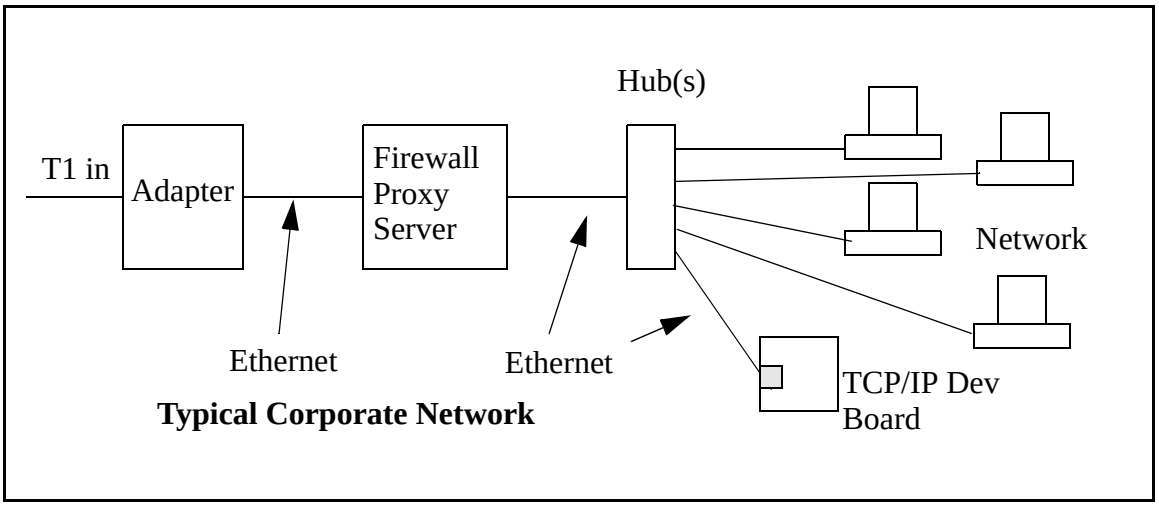
In order to set up this direct connection, the user will have to use a new PC (right out of the box), or disconnect a PC from the corporate network, or as yet another approach install a second Ethernet adapter and set up a separate private network attached to the second Ethernet adapter. Disconnecting your PC from the corporate network may be easy or nearly impossible, depending on how it is set up. Mobile PCs, such as laptops, are designed to be connected and disconnected, and will present the least problem. If your PC boots from the network or is dependent on the network for some or all of its disks, then it probably should not be disconnected. If a second Ethernet adapter is used, be aware that Windows TCP/IP will send messages to one adapter or the other, depending on the IP address and the binding order in Microsoft products. Thus you should have different ranges of IP addresses on your private network from those used on the corporate network. If both networks service the same IP address, then Windows may send a packet intended for your private network to the corporate network. A similar situation will take place if you use a dial-up line to send a packet to the Internet. Windows may try to send it via the local Ethernet network if it is also valid for that network.

The following IP addresses are set aside for local networks, and are not allowed on the Internet: 10.0.0.0 to 10.255.255.255, 172.16.0.0 to 172.31.255.255, and 192.168.0.0 to 192.168.255.255.

The TCP/IP Development Board uses a 10Base-T type of Ethernet connection, which is the most common scheme. The RJ-45 connectors are similar to U.S. style telephone connectors, are larger and have 8 contacts.

An alternative to the direct connection using a crossover cable is a direct connection using a hub. The hub relays packets received on any port to all of the ports on the hub. Hubs are low in cost and are readily available. The TCP/IP Development Board uses 10 Mbps Ethernet, so the hub or Ethernet adapter must be either a 10 Mbps unit or a 10/100 Mbps unit.

In a corporate setting where the Internet is brought in via a high-speed line, there are typically machines between the outside Internet and the internal network. These machines include a combination of proxy servers and firewalls that filter and multiplex Internet traffic. In the configuration below, the TCP/IP Development Board could be given a fixed address so any of the computers on the local network would be able to contact it. It may be possible to configure the firewall or proxy server to allow hosts on the Internet to contact the TCP/IP Development Board directly, but it would probably be easier to place the TCP/IP Development Board directly on the external network outside the firewall. This avoids some of the configuration complications by sacrificing some security.



Typical Corporate Network

If your system administrator can give you an Ethernet cable along with its IP address, the netmask and the gateway address, then you may be able to run the sample programs without having to setup a direct connection between your computer and the TCP/IP Development 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.

4.3 IP Addresses Explained

IP (Internet Protocol) addresses are expressed as 4 decimal numbers separated by periods, for example:

216.103.126.155

10.1.1.6

Each decimal number must be between 0 and 255. The total IP address is a 32-bit number consisting of the 4 bytes expressed as shown above. A local network uses a group of adjacent IP addresses. There are always 2^N IP addresses in a local network. The netmask (also called subnet mask) determines how many IP addresses belong to the local network. The netmask is also a 32-bit address expressed in the same form as the IP address. An example netmask is

255.255.255.0

This netmask has 8 zero bits in the least significant portion, and this means that 2^8 addresses are a part of the local network. Applied to the IP address above (216.103.126.155), this netmask would indicate that the following IP addresses belong to the local network:

216.103.126.0

216.103.126.1

216.103.126.2

etc.

216.103.126.254

216.103.126.255

The lowest and highest address are reserved for special purposes. The lowest address (216.102.126.0) is used to identify the local network. The highest address (216.102.126.255) is used as a broadcast address. Usually one other address is used for the address of the gateway out of the network. This leaves 256 - 3 = 253 available IP addresses for the example given.

4.4 How IP Addresses are Used

The actual hardware connection via an Ethernet uses Ethernet adapter addresses (also called MAC addresses). These are 48-bit addresses and are unique for every Ethernet adapter manufactured. In order to send a packet to another computer, given the IP address of the other computer, it is first determined if the packet needs to be sent directly to the other computer or to the gateway. In either case, there is an IP address on the local network to which the packet must be sent. A table is maintained to allow the protocol driver to determine the MAC address corresponding to a particular IP address. If the table is empty, the MAC address is determined by sending an Ethernet broadcast packet to all devices on the local network asking the device with the desired IP address to answer with its MAC address. In this way, the table entry can be filled in. If no device answers, then the device is nonexistent or inoperative, and the packet cannot be sent.

Private IP addresses are arbitrary and can be allocated as desired, provided that they don't conflict with other IP addresses. However, if IP addresses are to be used with the Internet, then the IP addresses must be assigned to your connection by the proper authorities, generally by delegation via your service provider.

4.5 Dynamically Assigned Internet Addresses

In many instances, IP addresses are assigned temporarily. This is the normal procedure when you use a dial-up Internet service provider (ISP). Your system will be provided with an IP address that it can use to send and receive packets. This IP address will only be valid for the duration of the call, and may not actually be a real IP address. Such an address works for browsing the Web, but cannot be used for transactions originating elsewhere since no other system has any way to know the Internet address except by first receiving a packet from you. (If you want to find the IP address assigned by a dial-up ISP, run the program **`winipcfg`** while connected and look at the address for the ppp adapter under Windows 98.)

In a typical corporate network that is isolated from the Internet by a firewall and/or proxy server using address translation, the IP addresses are not usually actual Internet addresses, and may be assigned statically or dynamically.

1. An external IP address may be assigned statically. All you need to do is get an unused IP address and assign it to the TCP/IP Development Board.
2. An external IP address may be assigned dynamically. You will have to get an IP address that is valid, but is outside the range of IP addresses that are assigned dynamically.
3. A combination of the external address and port number can be translated to the combination of IP address and port number being used by your TCP/IP Development Board.

These steps will enable you to communicate from a PC on the network to the TCP/IP Development Board. If you want to communicate to the TCP/IP Development Board from the external Internet, then an actual IP address must be assigned to the TCP/IP Development Board. It may be possible to set up the firewall to pass a real IP address, or it may be necessary to connect the TCP/IP Development Board in front of the firewall to accomplish this.

4.6 How to Set IP Addresses in the Sample Programs

Most of the sample programs use macros to define the IP address assigned to the board and the IP address of the gateway, if there is a gateway.

```
#define MY_IP_ADDRESS "216.112.116.155"
#define MY_NETMASK "255.255.255.248"
#define MY_GATEWAY "216.112.116.153"
```

In order to do a direct connection, the following IP addresses can be used for the BL2000:

```
#define MY_IP_ADDRESS "10.1.1.2"
#define MY_NETMASK "255.255.255.248"
// #define MY_GATEWAY "216.112.116.153"
```

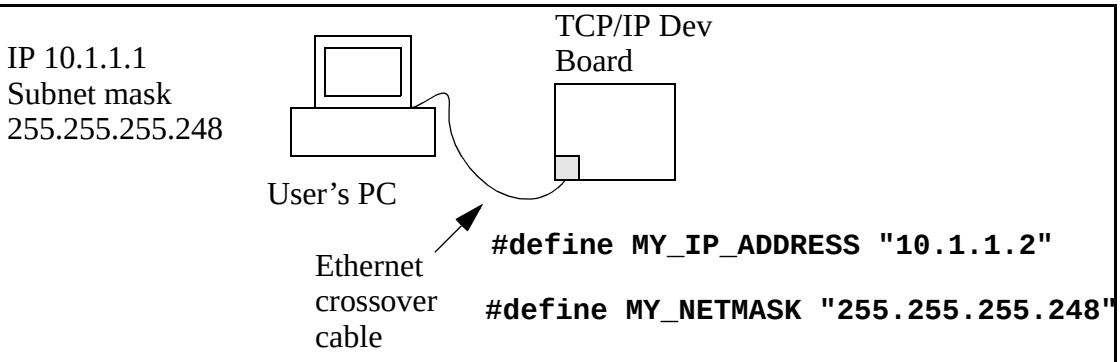
In this case, the gateway is not used and is commented out. The IP address of the board is defined to be 10.1.1.2. The IP address of your PC can be defined as 10.1.1.1.

4.7 How to Set Up your Computer's IP Address for a Direct Connection

When your computer is connected directly to the BL2000 via an Ethernet connection, you need to assign an IP address to your computer. To assign the PC the address 10.1.1.1 with the subnetmask 255.255.255.248 under Windows 98, do the following.

Click on **Start > Settings > Control Panel** to bring up the Control Panel, and then double-click the Network icon. In the window find the line of the form **TCP/IP > Ethernet adapter name**. Double-click on this line to bring up the TCP/IP properties dialog box. You can edit the IP address directly and the subnet mask. (Disable "obtain an IP address automatically.") You may want to write down the existing values in case you have to restore them later. It is not necessary to edit the gateway address since the gateway is not used with direct connect.

The method of setting the IP address may differ for different versions of Windows, such as 95, NT or 2000.



Direct Connection PC to TCP/IP Development Board

4.8 Run the PINGME.C Demo

In order to run this program, edit the IP address and netmask in the **PINGME.C** program (**SAMPLES\TCPIP\ICMP**) to the values given above (10.1.1.2 and 255.255.255.248). Compile the program and start it running under Dynamic C. The crossover cable is connected from your computer's Ethernet adapter to the TCP/IP Development Board's RJ-45 Ethernet connector. When the program starts running, the green **LNK** light on the TCP/IP Development 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 ping program:

```
ping 10.1.1.2
```

or by **Start > Run**

and typing the command

```
ping 10.1.1.2
```

Notice that the red **ACT** light flashes on the TCP/IP Development Board while the ping is taking place indicating the transfer of data. The ping routine will ping the board four times and write a summary message on the screen describing the operation.

4.9 Running More Demo Programs With a Direct Connection

The programs **STATIC.C** and **SSI.C** (**SAMPLES\TCPIP\HTTP**) demonstrate how to make the TCP/IP Development Board a Web server. This program allows you to turn the LEDs on an attached Demonstration Board from the Tool Kit on and off from a remote Web browser. In order to run these sample programs, edit the IP address as for the pingme program, compile the program and start it executing. Then bring up your Web browser and enter the following server address: http://10.1.1.2.

This should bring up the Web page served by the sample program. The demo program **STATIC.C** is a static Web page. The sample program **SSI.C** allows you to control the TCP/IP Development Board from the Web browser, turning the LED indicators on and off on the Demo Board attached to the main TCP/IP Development Board.

The sample program **RXSAMPLE.C** (**SAMPLES\TELNET**) allows you to communicate with the TCP/IP Development Board using the Telnet protocol. To run this program, edit the IP address and compile it and start it running. Run the telnet program on your PC (**Start > Run** `telnet 10.1.1.2`). Each character you type will be printed in Dynamic C's **STDIO** window, indicating that the board is receiving the characters typed via TCP/IP.

4.10 Where Do I Go From Here?

> **NOTE:** If you purchased your TCP/IP Development Board through a distributor or through a Z-World or Rabbit Semiconductor partner, contact the distributor or Z-World partner first for technical support.

If there are any problems at this point:

- Check the Z-World/Rabbit Semiconductor Technical Bulletin Board at www.zworld.com/support/bb/.
- Use the Technical Support e-mail form at www.zworld.com/support/support_submit.html.
- Call our Technical Support center:

  Z-World Technical Support, (530) 757-3737.

  Rabbit Semiconductor Technical Support, (530) 757-8400.

If the sample programs ran fine, you are now ready to go on.

Refer to the ***Dynamic C TCP/IP User's Manual*** to develop your own applications. ***An Introduction to TCP/IP*** provides background information on TCP/IP, and is included on the CD.

5. SERIAL PORTS AND DIGITAL I/O

Chapter 5 describes how to set up TCP/IP Development boards for serial communication, and how to use the digital I/O.

5.1 Serial Communication

In the factory-default configuration, the TCP/IP Development Board has one RS-232 (3-wire) serial channel, one RS-485 serial channel, and one synchronous CMOS serial channel. The TCP/IP Development Board may be configured for 5-wire RS-232 or two 3-wire RS-232 channels. The exact configuration instructions depend on the version of the TCP/IP Development Board you have. This information is etched on the bottom side of the printed circuit board, or you can readily determine your version by examining the diagrams below to find the one that matches your board.

Version 175-0188 Rev. A & B

The RS-232 transceiver may be used as a 5-wire RS-232 channel or as two 3-wire RS-232 channels at the expense of the RS-485 channel by adding 0 Ω surface-mounted resistors at R61 and R62 as shown in Figure 7(a). The RS-485 chip (U10) and the associated bias and termination resistors (R58, R59, and R60) shown in Figure 7 must be removed when configuring the TCP/IP Development Board for either one 5-wire RS-232 or two 3-wire RS-232.

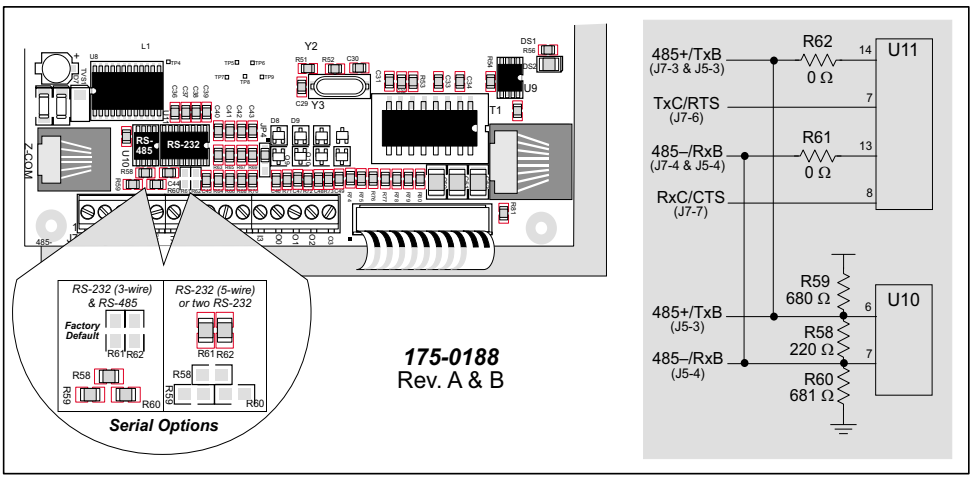


Figure 7(a). RS-232/RS-485 Serial Communication Options

Table 2(a) summarizes the options. Note that the parameters in the **`serMode`** software function call must also be set to match the hardware configuration being used.

Table 2(a). Serial Communication Configurations (Version 175-0188 Rev. A & B)

Item	*Factory Default* One 3-wire RS-232 & RS-485	Two 3-wire RS-232	One 5-wire RS-232
R58–R60	In	Out	Out
R61–R62	Out	In	In
U10	In	Out	Out
J7-3 & J5-3	RS-485+	TxB	TxB
J7-4 & J5-4	RS-485–	RxB	RxB
J7-6	TxC	TxC	RTS
J7-7	RxC	RxC	CTS

Version 175-0188 Rev. C

The RS-232 transceiver may be used as a 5-wire RS-232 channel or as two 3-wire RS-232 channels at the expense of the RS-485 channel, which is connected through 0 Ω surface-mounted resistors at R82 and R83 as shown in Figure 7(b). R82 and R83, shown in Figure 7(b), must be removed when configuring the TCP/IP Development Board for either one 5-wire RS-232 or two 3-wire RS-232. U10 and the associated bias and termination resistors (R58, R59, and R60) must also be removed, but R82 and R83 are left installed, if you wish the TxB and RxB RS-232 signals to be available on header J5.

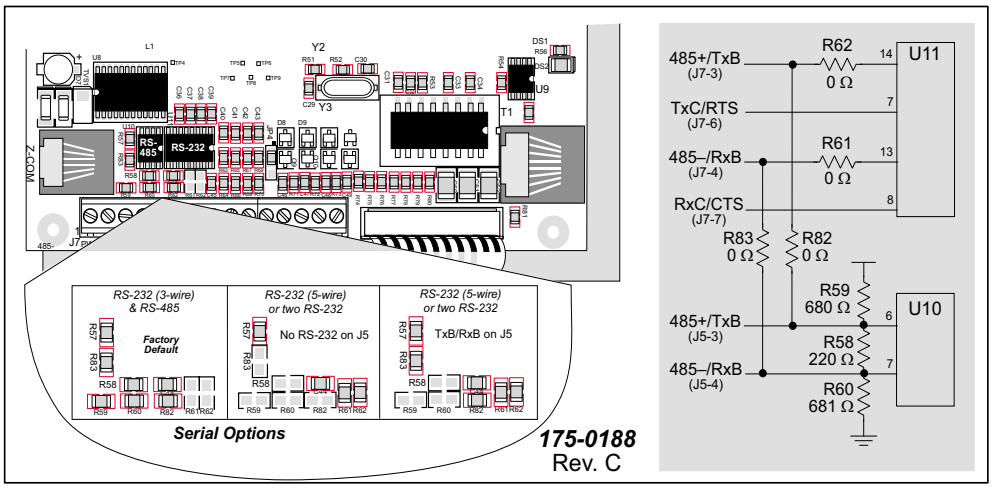


Figure 7(b). RS-232/RS-485 Serial Communication Options

Table 2(b) summarizes the options. Note that the parameters in the **`serMode`** software function call must also be set to match the hardware configuration being used.

Table 2(b). Serial Communication Configurations (Version 175-0188 Rev. C)

Item	*Factory Default* One 3-wire RS-232 & RS-485	Two 3-wire RS-232	One 5-wire RS-232	RS-232 on J5
R58–R60	In	—	—	Out
R61–R62	Out	In	In	In
R82–R83	In	Out	Out	In
U10	In	In	In	Out
J7-3	RS-485+	TxB	TxB	TxB
J7-4	RS-485–	RxB	RxB	RxB
J7-6	TxC	TxC	RTS	TxC or RTS
J7-7	RxC	RxC	CTS	RxC or CTS
J5-3	RS-485+	—	—	TxB
J5-4	RS-485–	—	—	RxB

Version 175-0206

The RS-232 transceiver may be used as a 5-wire RS-232 channel or as two 3-wire RS-232 channels at the expense of the RS-485 channel, which is connected through jumpers across header JP7 as shown in Figure 7(c). The jumper configurations are shown in Figure 7(c).

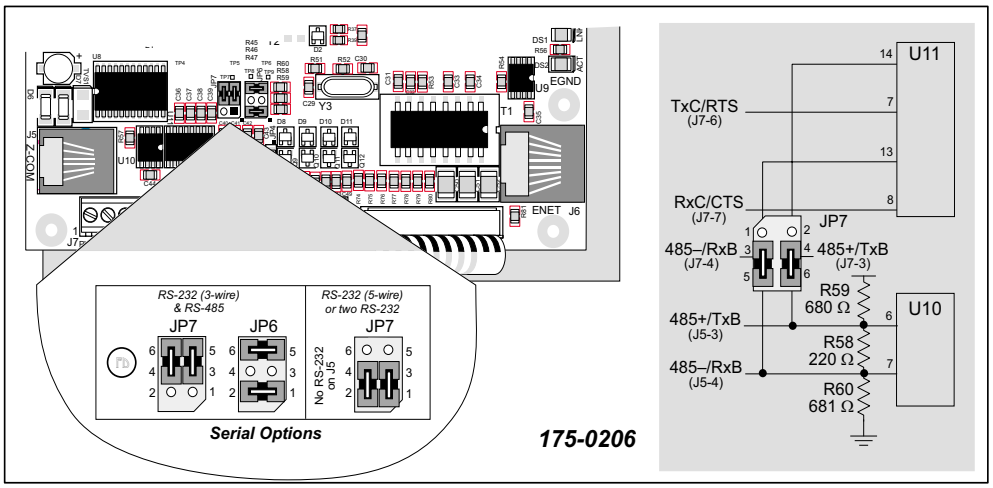


Figure 7(c). RS-232/RS-485 Serial Communication Options

Table 2(c) summarizes the options. Note that the parameters in the **`serMode`** software function call must also be set to match the hardware configuration being used.

Table 2(c). Serial Communication Configurations (Version 175-0206)

Item	*Factory Default* One 3-wire RS-232 & RS-485	Two 3-wire RS-232	One 5-wire RS-232	RS-232 on J5
Header JP7	3–5 4–6	1–3 2–4	1–3 2–4	1–5 2–6
Header JP6	1–2 5–6	—	—	No jumpers installed
U10	In	In	In	Out
J7-3	RS-485+	TxB	TxB	—
J7-4	RS-485–	RxB	RxB	—
J7-6	TxC	TxC	RTS	TxC or RTS
J7-7	RxC	RxC	CTS	RxC or CTS
J5-3	RS-485+	—	—	TxB
J5-4	RS-485–	—	—	RxB

5.1.1 RS-232

The TCP/IP Development Board's RS-232 serial channel is connected to an RS-232 transceiver, U11. U11 provides the voltage output, slew rate, and input voltage immunity required to meet the RS-232 serial communication protocol. Basically, the chip translates the Rabbit 2000's 0 V to +Vcc signals to RS-232 signal levels. Note that the polarity is reversed in an RS-232 circuit so that +5 V is output as approximately -10 V and 0 V is output as approximately +10 V. U11 also provides the proper line loading for reliable communication.

The maximum baud rate is 115,200 bps. RS-232 can be used effectively at this baud rate for distances up to 15 m.

5.1.2 RS-485

The TCP/IP Development Board has one RS-485 serial channel, which is connected to the Rabbit 2000 Serial Port B through U10, an RS-485 transceiver. The chip's slew rate limiters provide for a maximum baud rate of 250,000 bps, which allows for a network of up to 1200 m (or 4000 ft). The half-duplex communication uses the Rabbit 2000's PC0 pin to control the data enable on the communication line.

The RS-485 signals are available on pins 3 and 4 of header J7, and on J5, the RJ-12 jack.

The TCP/IP Development Board can be used in an RS-485 multidrop network. Connect the 485+ to 485+ and 485– to 485– using single twisted-pair wires (nonstranded, tinned).

Alternatively, the RS-485 multidrop network may be hooked up using cables with RJ-12 plugs. Note that the RJ-12 jack has +RAW_485 and GND, which means that only ***one*** TCP/IP Development Board needs to be connected to an external power source via an AC adapter. When doing so, ensure that the AC adapter has sufficient capacity for the network — each TCP/IP Development Board nominally draws 100 mA at 24 VDC.

NOTE: If you plan to connect a power supply to more than one TCP/IP Development Board in an RS-485 network using the RJ-12 jacks, rework the RS-485 cables so they do not connect +RAW_RS485 through the RJ-12 jack to the boards in the network.

NOTE: The RS-485 port is available only in the factory default configuration. The RS-485 port will not be available when you select the configuration option for both 3-wire RS-232 ports or one 5-wire RS-232 port.

The TCP/IP Development Board comes with a 220 Ω termination resistor and two 680 Ω bias resistors installed and enabled with jumpers across pins 1–2 and 5–6 on header JP6, as shown in Figure 8.

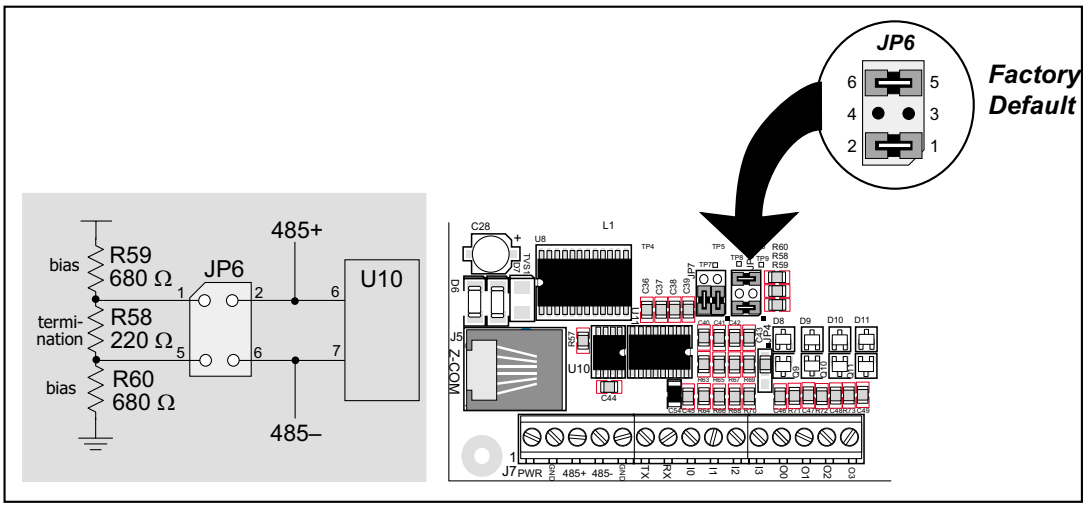


Figure 8. RS-485 Termination and Bias Resistors

The bias and termination resistors in a multidrop network should only be enabled on both end nodes of the network. Disable the termination and bias resistors on the intervening TCP/IP Development Boards in the network by removing both jumpers from header JP6. Note that older versions of the TCP/IP Development Board do not have this jumper feature, and the surface-mounted bias and termination resistors shown in Figure 8 have to be removed in networks containing more than 10 TCP/IP Development Boards.

5.1.3 Programming Port

The TCP/IP Development Board has a 10-pin programming header labeled J4. The programming port uses the Rabbit 2000's Serial Port A for communication. The Rabbit 2000 startup-mode pins (SMODE0, SMODE1) are presented to the programming port so that an externally connected device can force the TCP/IP Development Board to start up in an external bootstrap mode.

> **NOTE:** Refer to the ***Rabbit 2000 Microprocessor User's Manual*** for more information related to the bootstrap mode.

The programming port is used to start the TCP/IP Development Board in a mode where the TCP/IP Development Board will download a program from the port and then execute the program. The programming port transmits information to and from a PC while a program is being debugged.

The TCP/IP Development Board can be reset from the programming port.

The Rabbit 2000 status pin is also presented to the programming port. The status pin is an output that can be used to send a general digital signal.

5.1.4 Serial Communication Software

Library files included with Dynamic C provide a full range of serial communications support. The **RS232.LIB** library provides a set of circular-buffer-based serial functions. The **PACKET.LIB** library provides packet-based serial functions where packets can be delimited by the 9th bit, by transmission gaps, or with user-defined special characters. Both libraries provide blocking functions, which do not return until they are finished transmitting or receiving, and nonblocking functions, which must be called repeatedly until they are finished. For more information, see the ***Dynamic C Premier User's Manual*** and Technical Note 213, ***Rabbit 2000 Serial Port Software***.

The following function calls are specific to the TCP/IP Development Board.

int serMode (int mode);

User interface to set up up serial communication lines for the TCP/IP Development Board. Call this function after **serXOpen()**.

Parameters

mode is the defined serial port configuration of the devices installed.

Mode	Serial Port	
	B	C
0	RS-485	RS-232, 3-wire
1	RS-232, 3-wire	RS-232, 3-wire
2	RS-232, 5-wire	CTS/RTS

Return Value

0 if correct mode, 1 if not.

See Also

serB485Tx, serB485Rx

void serB485Tx();

Sets pin 3 (DE) high to disable Rx and enable Tx.

See Also

serMode, serB485Rx

void serB485Rx();

Resets pin 3 (DE) low to enable Rx and disable Tx.

See Also

serMode, serB485Tx

5.1.4.1 Sample Serial Communication Programs

RS-232

1. Connect RX to TX as shown in Figure 9 below.

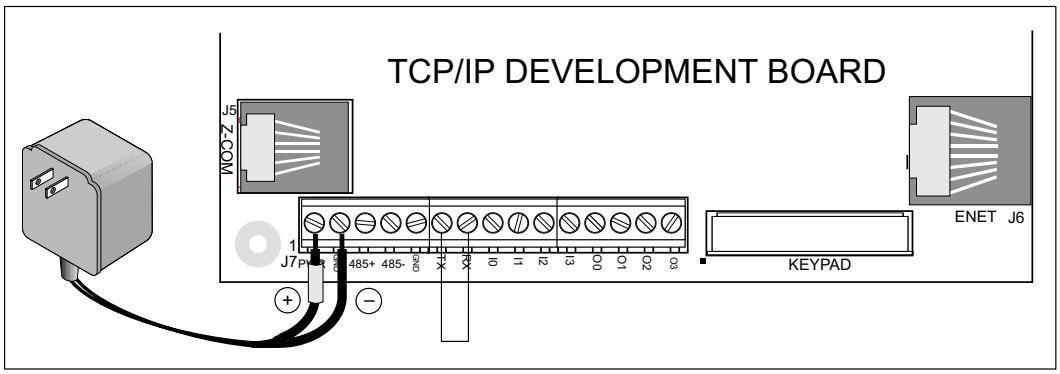


Figure 9. TCP/IP Development Board Setup for RS-232 Serial Communication Demonstration

2. Connect the programming cable to header J4 on the TCP/IP Development Board. Apply power to the TCP/IP Development Board.

3. Open the sample program `SAMPLES\ICOM\ICOM232.C` and press `F9`.

This program demonstrates a simple RS-232 loopback displayed in the Dynamic C **STDIO** window.

RS-485

1. **Connect 485+ to 485+, 485– to 485–, and GND to GND as shown in Figure 10 below. If you do not have a separate wall transformer for the other board, also connect PWR to PWR as shown in Figure 10.**

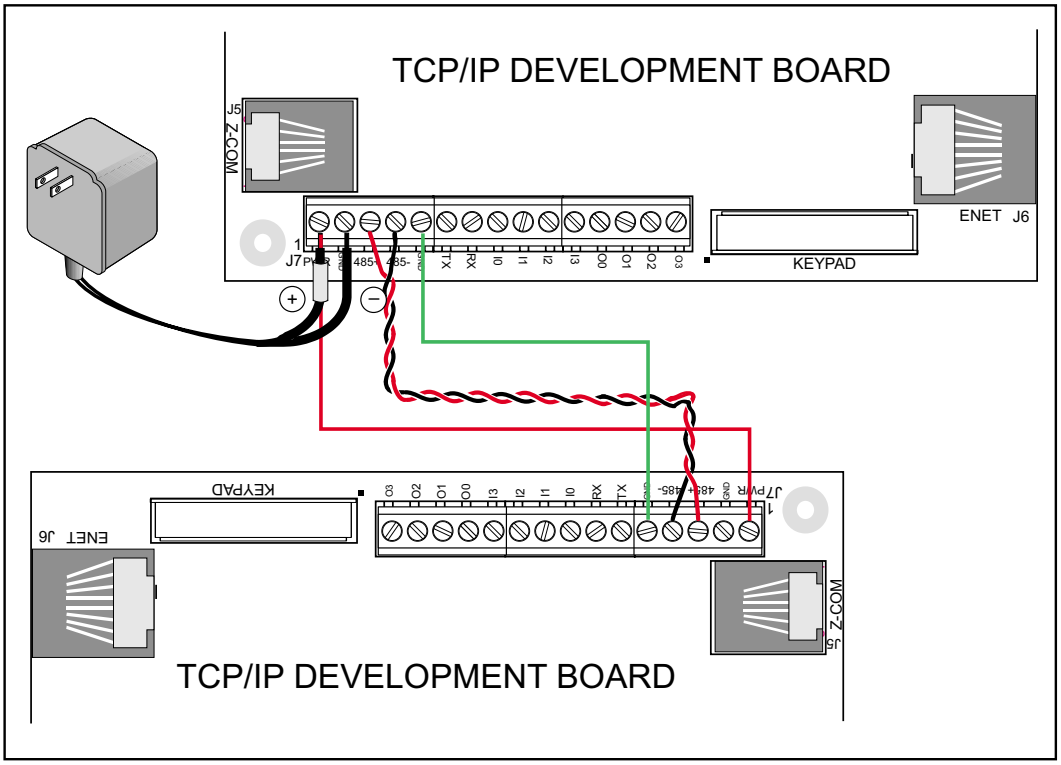


Figure 10. TCP/IP Development Board Setup for RS-485 Serial Communication Demonstration

2. **Connect the programming cable to header J4 on one TCP/IP Development Board. This will be the slave, the other board will be the master. Apply power to the TCP/IP Development Boards.**
3. **Open the sample program `SAMPLES\ICOM\ICOM485.C`. You will find some code for the master, and some code for the slave. Copy and save the master and slave versions separately.**
4. **Open the sample slave program and press F9.**
5. **Connect the programming cable to header J4 on the master TCP/IP Development Board.**
6. **Open the master program and press F9.**

This program demonstrates a simple RS-485 transmission of lower-case letters to a slave. The slave will send back converted upper case letters back to the master, which then displays them in the Dynamic C **STDIO** window.

5.2 Digital I/O

5.2.1 Digital Inputs

Pins 8–11 on header J7 have the four digital inputs IN0–IN3. Each of the four digital 0 V to 5 V inputs is protected over a range of –36 V to +36 V. The TCP/IP Development Board is factory-configured for the digital inputs to be pulled up to +5 V, but the digital inputs can also be pulled down by moving the surface-mounted jumper at JP4. The jumper settings and the location of JP4 are shown in Figure 11.

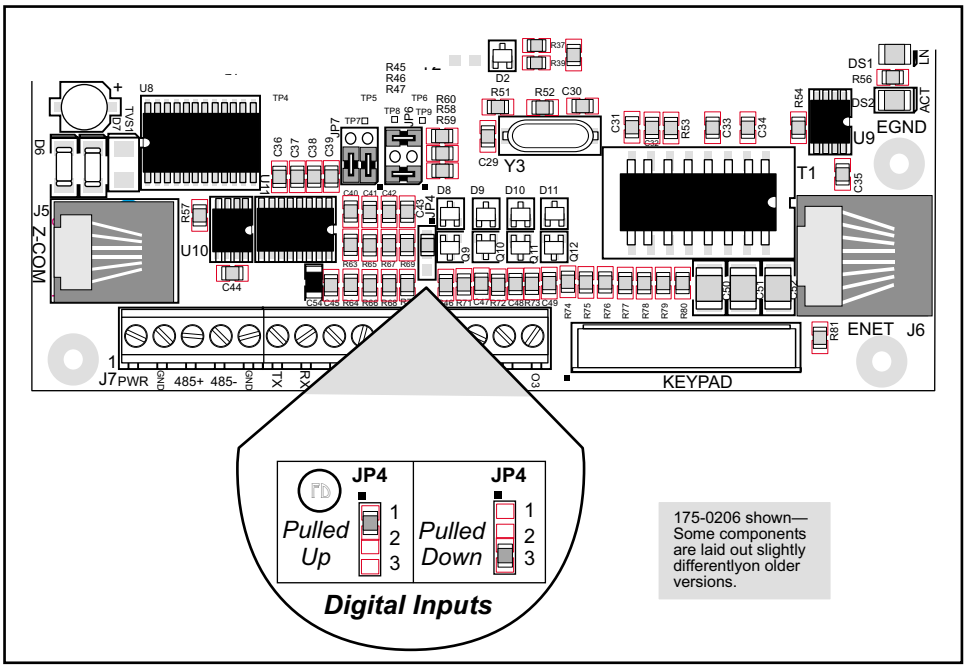


Figure 11. Surface-Mounted Jumper Configurations for Selecting Pullup/Pulldown on the Digital Inputs

5.2.2 Digital Outputs

Pins 12–15 on header J7 have the four digital outputs OUT0–OUT3. Each of the four open-collector digital outputs can sink up to 200 mA at 40 V DC.

5.2.3 Digital I/O Software

```
void digOut (int channel, int value);
```

Sets the state of a digital output.

Parameters

`channel` is the output channel number (0, 1, 2, or 3).

`value` is the output value (0 or 1).

Return Value

None.

See Also

`digIn`

```
int digIn (int channel);
```

Reads the state of a digital input.

Parameters

`channel` is the input channel number (0, 1, 2, or 3).

Return Value

The state of the input (0 or 1).

See Also

`digOut`

5.2.4 Sample Digital I/O Programs

1. **Connect the programming cable to header J4 on the TCP/IP Development Board. Apply power to the TCP/IP Development Board.**
2. **Open the sample program `SAMPLES\ICOM\ICOMIO.C` and press `F9`.**

This program demonstrates how to turn the I/O on and off.

LEGAL NOTICE

RABBIT SEMICONDUCTOR PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE-SUPPORT DEVICES OR SYSTEMS UNLESS A SPECIFIC WRITTEN AGREEMENT REGARDING SUCH INTENDED USE IS ENTERED INTO BETWEEN THE CUSTOMER AND RABBIT SEMICONDUCTOR PRIOR TO USE. Life-support devices or systems are devices or systems intended for surgical implantation into the body or to sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling and user's manual, can be reasonably expected to result in significant injury.

No complex software or hardware system is perfect. Bugs are always present in a system of any size. In order to prevent danger to life or property, it is the responsibility of the system designer to incorporate redundant protective mechanisms appropriate to the risk involved.

All Rabbit Semiconductor products are 100 percent functionally tested. Additional testing may include visual quality control inspections or mechanical defects analyzer inspections. Specifications are based on characterization of tested sample units rather than testing over temperature and voltage of each unit. Rabbit Semiconductor products may qualify components to operate within a range of parameters that is different from the manufacturer's recommended range. This strategy is believed to be more economical and effective. Additional testing or burn-in of an individual unit is available by special arrangement.

APPENDIX A. TCP/IP DEVELOPMENT BOARD SPECIFICATIONS

A.1 Electrical and Mechanical Specifications

Figure A-1 shows the mechanical dimensions for the TCP/IP Development Board.

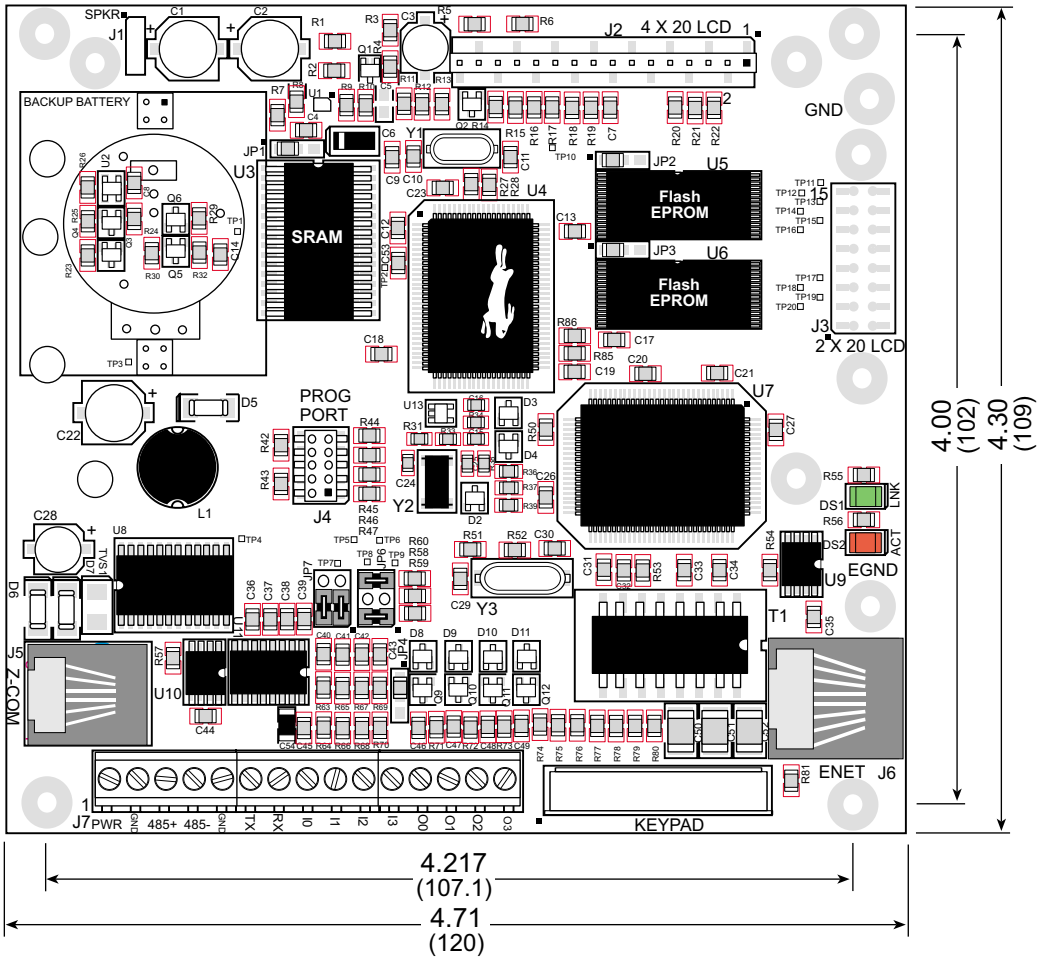


Figure A-1. TCP/IP Development Board Dimensions

Table A-1 lists the electrical, mechanical, and environmental specifications for the TCP/IP Development Board.

Table A-1. TCP/IP Development Board Specifications

Parameter	Specification
Board Size (with optional backup battery board)	4.30" × 4.71" × 0.79" (109 mm × 120 mm × 20 mm)
Connectors	15 screw terminals, 1 RJ-12, and 1 RJ-45
Operating Temperature	–20°C to +70°C
Humidity	5% to 95%, noncondensing
Input Voltage	9 V to 40 V DC
Current	100 mA @ 12 VDC
Ethernet Interface	Direct connection to 10BaseT Ethernet networks via RJ-45 connection
Digital Inputs	4 protected, 0 V to 5 V DC (protection from –36 V to + 36 VDC max.)
Digital Outputs	4 open collector, sinking (200 mA, 40 V DC max.)
Microprocessor	Rabbit 2000™
Clock	18.432 MHz
SRAM	128K, surface mount (supports 32K–512K)
Flash EPROM	256K for program and data plus 256K for file storage (supports 128K–512K)
Timers	7 eight-bit timers available
Serial Ports	• 1 RS-232 (3-wire), 1 RS-485, and 1 RS-232 programming port • RS-232 (3-wire) and RS-485 may be reconfigured for 1 RS-232 (5-wire) or 2 RS-232 (3-wire)
Serial Rate	Maximum asynchronous 115,200 bps for both serial ports
Watchdog/Supervisor	Yes
Time/Date Clock	Yes
Backup Battery	On backup battery board (not included)

A.2 Jumper Configurations

Figure A-2 shows the header locations used to configure the various TCP/IP Development Board options via jumpers.

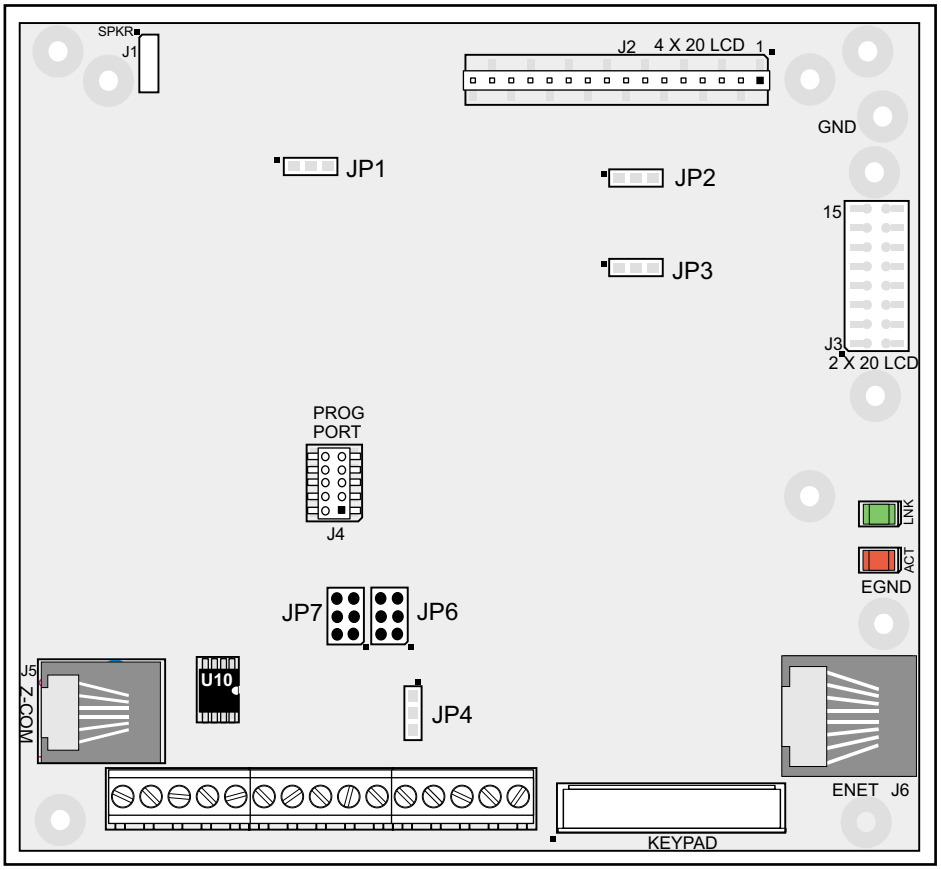


Figure A-2. Location of TCP/IP Development Board Configurable Positions

Table A-2 lists the configuration options.

Table A-2. TCP/IP Development Board Jumper Configurations

Header	Description	Pins Connected		Factory Default
JP1	SRAM Size	1–2	128K	✕
		2–3	512K	
		None	32K	
JP2	Flash 1 Memory Size (U5)	1–2	128K/256K	✕
		2–3	512K	
JP3	Flash 2 Memory Size (U6)	1–2	128K/256K	✕
		2–3	512K	
JP4	Digital Input Pull-Up/Pull-Down Resistors	1–2	Pulled up	✕
		2–3	Pulled down	
JP5	Flash Memory Bank Select	1–2	Normal Mode	✕
		2–3	Bank Mode	
JP6	RS-485 Bias and Termination Resistors	1–2 5–6	Bias and termination resistors connected	✕
		None	Bias and termination resistors ***not*** connected	
JP7	RS-232/RS-485 Select	1–3 2–4	RS-232 TxB/RxB (also TxC/RxC or RTS/CTS) signals on J7	
		3–5 4–6	RS-232 TxC/RxC and RS-485 signals on J7	✕
		1–5 2–6	RS-232 TxB/RxB signals on J5 (U10 must be removed)	

NOTE: Only headers JP6 and JP7 use actual jumpers. The other connections are made using 0 Ω surface-mounted resistors.

A.3 Conformal Coating

The areas around the crystal oscillator and the battery-backup circuit on the TCP/IP Development Board have had the Dow Corning silicone-based 1-2620 conformal coating applied. The conformally coated areas are shown in Figure A-3. The conformal coating protects these high-impedance circuits from the effects of moisture and contaiminants over time.

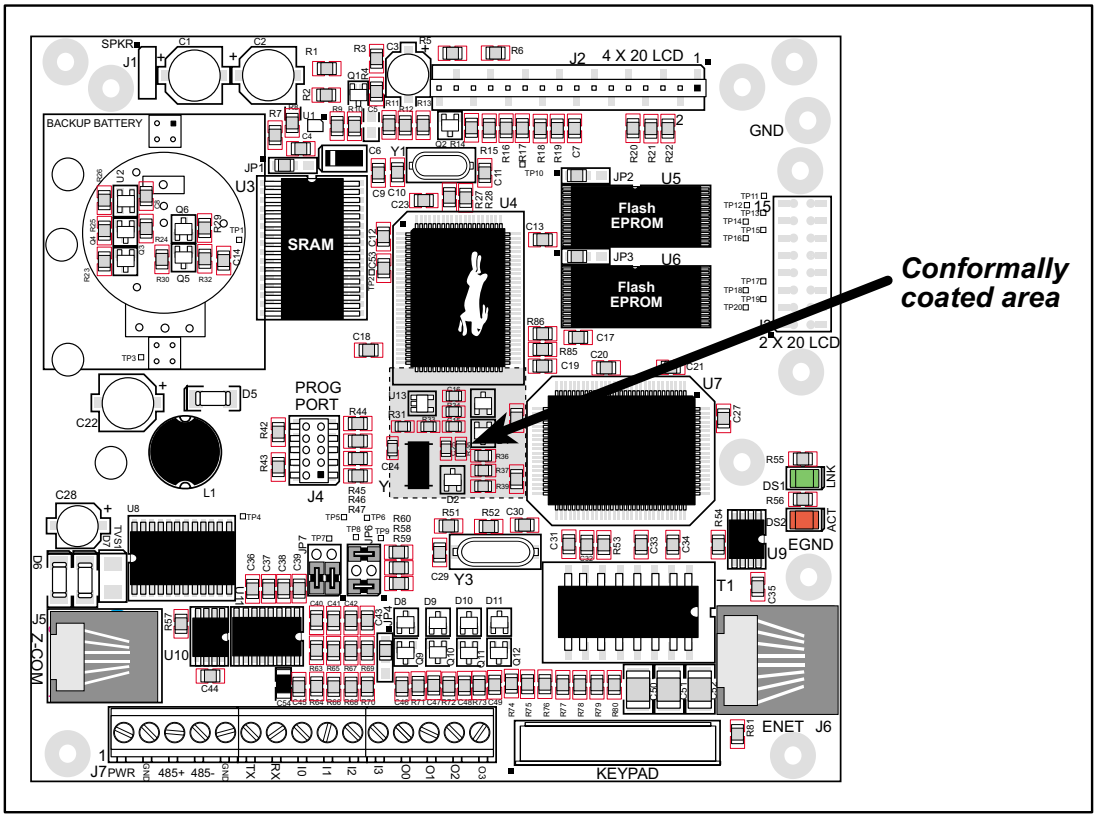


Figure A-3. TCP/IP Development Board Areas Receiving Conformal Coating

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

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

APPENDIX B. POWER MANAGEMENT

Appendix B describes the power circuitry distributed on the TCP/IP Development Board.

B.1 Power Supplies

Power is supplied to the TCP/IP Development Board from an external source either through header J7 or from another TCP/IP Development Board through header J5, the RJ-12 jack.

The TCP/IP Development Board itself is protected against reverse polarity by Shottky diodes at D6 and D7 as shown in Figure B-1. The Shottky diode has a low forward voltage drop, 0.3 V, which keeps the minimum DCIN required to power the TCP/IP Development Board lower than a normal silicon diode would allow.

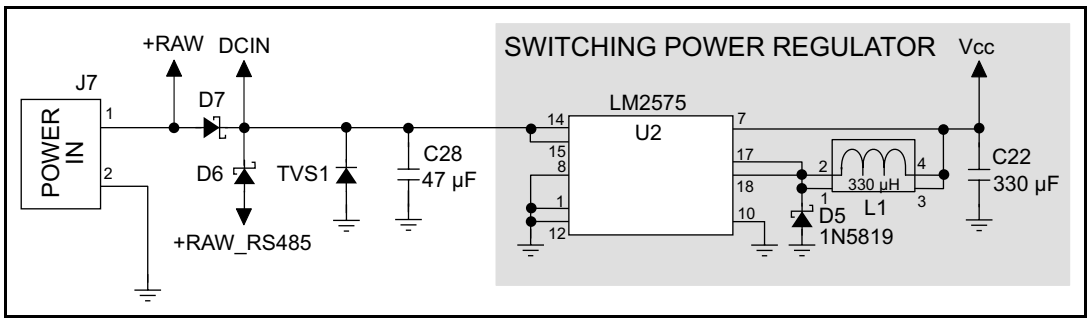


Figure B-1. TCP/IP Development Board Power Supply Schematic

Capacitor C28 provides surge current protection for the voltage regulator, and allows the external power supply to be located some distance away from the TCP/IP Development Board. A switching power regulator is used. The input voltage range is from 9 V to 40 V.

B.2 Batteries and External Battery Connections

A battery board with a 1000 mA·h lithium coin cell is available to provide power to the real-time clock and SRAM when external power is removed from the circuit. This allows the TCP/IP Development Board to continue to keep track of time and preserves the SRAM memory contents.

Figure B-2 shows the battery-board circuit.

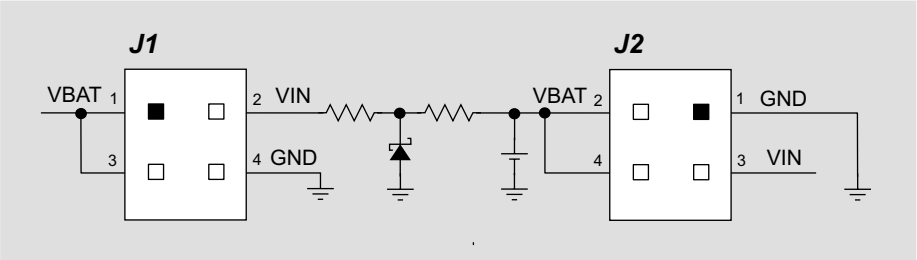


Figure B-2. Battery-Board Circuit

Alternatively, starting with the 175-0206 version of the TCP/IP Development Board, there is provision to add a soldered-in battery directly on the TCP/IP Development Board.

The drain on the battery is typically less than 20 µA when there is no external power applied. The battery can last more than 5 years:

$$\frac{1000\ \text{mA·h}}{20\ \mu\text{A}} = 5.7\ \text{years.}$$

The drain on the battery is typically less than 4 µA when external power ***is*** applied. The battery can last for its full shelf life:

$$\frac{1000\ \text{mA·h}}{4\ \mu\text{A}} = 28.5\ \text{years (shelf life = 10 years).}$$

Since the shelf life of the battery is 10 years, the battery can last for its full shelf life when external power is applied to the TCP/IP Development Board.

B.2.1 Battery-Backup Circuit

The battery-backup circuit serves two purposes:

- It reduces the battery voltage to the real-time clock, thereby reducing 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.

Figure B-3 shows the battery-backup circuitry on the TCP/IP Development Board.

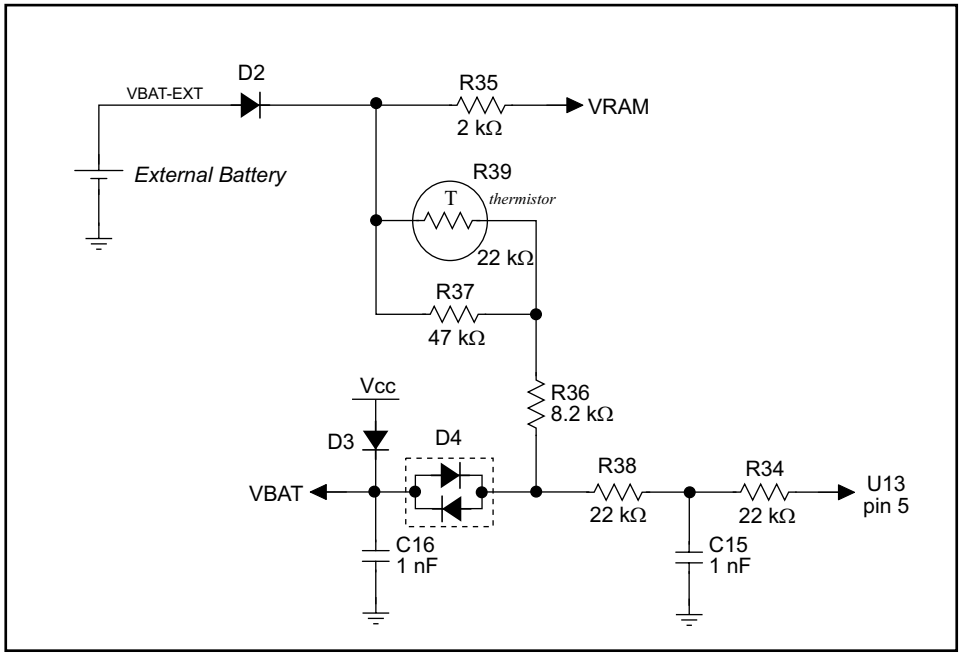


Figure B-3. TCP/IP Development Board Battery Backup Circuit

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 U13, 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 TCP/IP Development Board. 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.

B.2.2 Power to VRAM Switch

The VRAM switch, shown in Figure B-4, allows the battery backup 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.

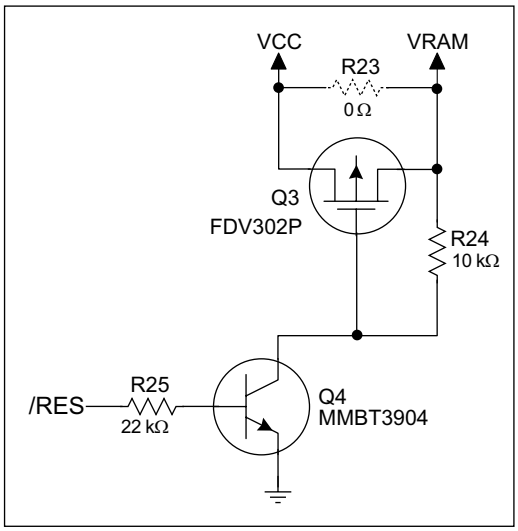


Figure B-4. VRAM Switch

Transistor Q3 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 TCP/IP Development Board is ***not*** resetting (pin 2 on U4 is high), the /RES line will be high. This turns on Q4, causing its collector to go low. This turns on Q3, allowing VRAM to nearly equal Vcc.

When the TCP/IP Development Board ***is*** resetting, the /RES line will go low. This turns off Q3 and Q4, providing an isolation between Vcc and VRAM.

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

B.2.3 Reset Generator

The TCP/IP Development Board uses a reset generator, U2, 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.

B.2.4 Installing/Replacing the Backup-Battery Board

An optional pluggable backup-battery board is available from Rabbit Semiconductor.

To install the backup-battery board, align the battery board over the outline as shown in Figure B-5, and plug it in. Be careful to align the connectors and the backup battery board. Fasten the backup board using a 4-40 × 3/16 screw and lockwasher.

> **NOTE:** Before replacing the backup-battery board, make sure that the TCP/IP Development Board is receiving power from the standard power supply. This makes sure that data in RAM are not lost when the battery backup board is removed temporarily.

To replace the backup-battery board, remove the screw and unplug the old battery board. Then install a replacement backup-battery board.

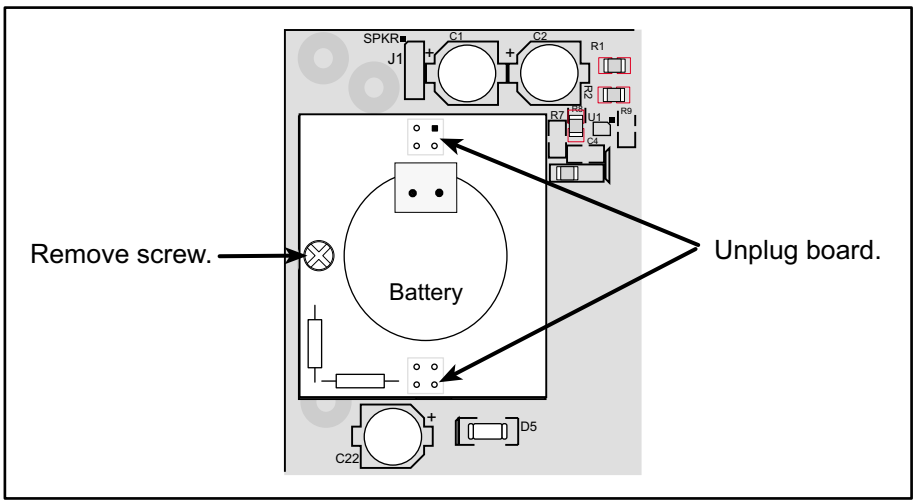


Figure B-5. Installing Backup Battery Board

Do ***not*** attempt to recharge the old battery and do ***not*** dispose of it in regular trash to avoid any risk of explosion or fire. You may either return the old backup battery board to Rabbit Semiconductor for recycling or send the battery yourself to an approved recycling facility.

B.3 Chip Select Circuit

Figure B-6 shows a schematic of the chip select circuit.

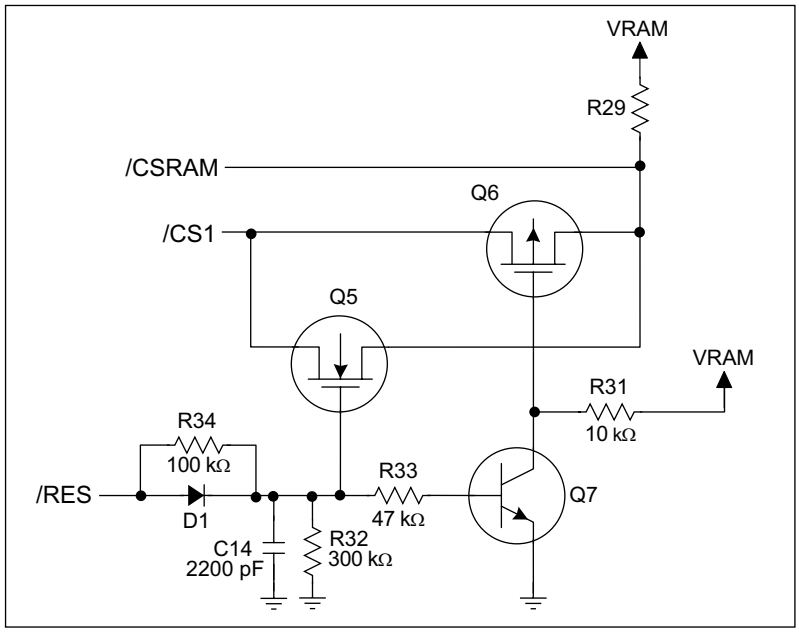


Figure B-6. Chip Select Circuit

The current drain on the battery in a battery-backed circuit must be kept at a minimum. When the TCP/IP Development Board 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). Q5 and Q6 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 (R29). This pullup resistor keeps /CSRAM at the VRAM voltage level (which under no power condition is the backup battery's regulated voltage at a little more than 2 V).

Transistors Q5 and Q6 are of opposite polarity so that a rail-to-rail voltage can be passed. When the /CS1 voltage is low, Q5 will conduct. When the /CS1 voltage is high, Q6 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.

The signal that turns the transistors on is a high on the processor's reset line, /RES. When the TCP/IP Development Board is not in reset, the reset line will be high, turning on n-channel Q5 and Q7. Q7 is a simple inverter needed to turn on Q6, a p-channel MOSFET. When a reset occurs, the /RES line will go low. This will cause C14 to discharge through R32 and R34. This small delay (about 160 µs) ensures that there is adequate time for the processor to write any last byte pending to the SRAM before the processor puts itself into a reset state. When coming out of reset, CS will be enabled very quickly because D1 conducts to charge capacitor C14.

APPENDIX C. PROGRAMMING CABLE

Appendix C provides additional information for the Rabbit 2000™ microprocessor when using the **DIAG** and **PROG** connectors on the programming cable. The **PROG** connector is used only when the programming cable is attached to the programming connector (header J4) while a new application is being developed. Otherwise, the **DIAG** connector on the programming cable allows the programming cable to be used as an RS-232 to CMOS level converter for serial communication, which is appropriate for monitoring or debugging an Intellicom system while it is running.

The programming port, which is shown in Figure C-1, can serve as a convenient communications port for field setup or other occasional communication need (for example, as a diagnostic port). There are several ways that the port can be automatically integrated into software. If the port is simply to perform a setup function, that is, write setup information to flash memory, then the controller can be reset through the programming port and a cold boot performed to start execution of a special program dedicated to this functionality.

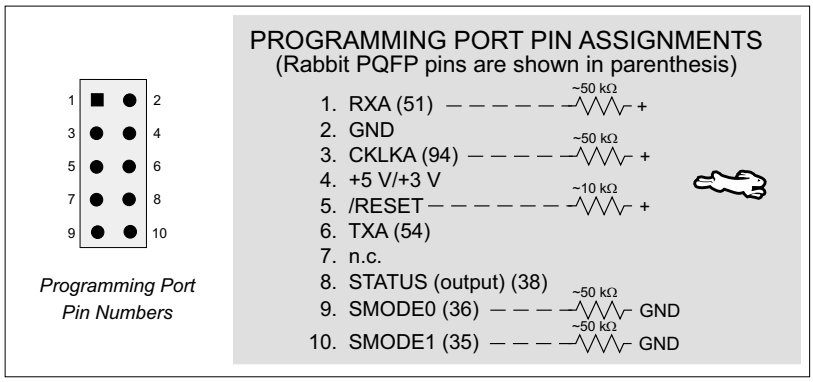


Figure C-1. Programming Port Pin Assignments

When the **PROG** connector is used, the /RESET line can be asserted by manipulating DTR and the STATUS line can be read as DSR on the serial port. The target can be restarted by pulsing reset and then, after a short delay, sending a special character string at 2400 bps. To simply restart the BIOS, the string 80h, 24h, 80h can be sent. When the BIOS is started, it can tell whether the programming cable is connected because the SMODE1 and SMODE0 pins are sensed as being high. This will cause the Rabbit 2000 to enter the bootstrap mode. The Dynamic C programming mode then can have an escape message that will enable the diagnostic serial port function.

Alternatively, the **DIAG** connector can be used to connect the programming port. The /RESET line and the SMODE1 and SMODE0 pins are not connected to this connector. The programming port is then enabled as a diagnostic port by polling the port periodically to see if communication needs to begin or to enable the port and wait for interrupts. The pull-up resistors on RXA and CLKA prevent spurious data reception that might take place if the pins floated.

If the clocked serial mode is used, the serial port can be driven by having two toggling lines that can be driven and one line that can be sensed. This allows a conversation with a device that does not have an asynchronous serial port but that has two output signal lines and one input signal line.

The line TXA (also called PC6) is zero after reset if the cold-boot mode is not enabled. A possible way to detect the presence of a cable on the programming port is for the cable to connect TXA to one of the SMODE pins and then test for the connection by raising PC6 (by configuring it as a general output bit) and reading the SMODE pin after the cold-boot mode has been disabled. The value of the SMODE pin is read from the SPCR register.

Once you establish that the programming port will never again be needed for programming, it is possible to use the programming port for additional I/O lines. Table C-1 lists the pins available for this alternate configuration.

Table C-1. TCP/IP Development Board Programming Port Pinout Configurations

Pin	Pin Name	Default Use	Alternate Use	Notes
1	RXA	Serial Port A	PC6—Input	
2	GND			
3	CLKA		PB1—Bitwise or parallel programmable input	
4	VCC			
5	RESET			Connected to reset generator U4
6	TXA	Serial Port A	PC7—Output	
8	STATUS		Output	
9	SMODE0		Input	Must be low when BL2000 boots up
10	SMODE1		Input	Must be low when BL2000 boots up

INDEX

S

T

W

SCHEMATICS

090-0095 TCP/IP Development Board Schematic

090-0042 Demonstration Board Schematic

090-0128 Programming Cable Schematic