RabbitCore RCM2100

C-Programmable Module with Ethernet

Getting Started Manual

019–0093 • 010915–D

RabbitCore RCM2100 Getting Started Manual

Part Number 019-0093 • 010915–C • Printed in U.S.A.

Z-World reserves the right to make changes and
improvements to its products without providing notice.

Notice to Users

Z-WORLD 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 Z-WORLD 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.

Trademarks

Rabbit 2000 is a trademark of Rabbit Semiconductor.

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

Z-World, Inc.

2900 Spafford Street
Davis, California 95616-6800
USA

Telephone: (530) 757-3737
Fax: (530) 753-5141

www.zworld.com

Rabbit Semiconductor

2932 Spafford Street
Davis, California 95616-6800
USA

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

www.rabbitsemiconductor.com

TABLE OF CONTENTS

Chapter 1: Introduction & Overview 1

Chapter 2: Hardware Setup 7

1. INTRODUCTION & OVERVIEW

The RCM2100 series is an advanced line of modules that incorporates the powerful Rabbit 2000 microprocessor, flash memory, static RAM and an RJ-45 Ethernet port, all on a PCB the size of a business card.

The RCM2100 series modules are designed for use on a motherboard that supplies power and interface to real-world I/O devices. Up to 40 pins of I/O and four serial ports are available for system interfacing.

To accommodate a variety of user and production needs, the RCM2100 family includes versions with varying amounts of onboard memory. Models with and without the Ethernet port are available, to permit simultaneous development of Ethernet-capable and cheaper non-Ethernet versions of production systems. All modules within the family are pin-for-pin compatible and may be installed or swapped in a matter of minutes.

1.1 RCM2100 Series Description

There are four production models in the RCM2100 module series, with an additional module that can be ordered in production quantities. If the standard models do not serve a user's needs, other variations can be specified and ordered in production quantities. (Contact your Z-World or Rabbit Semiconductor sales representative for details.)

Table 1 below provides a summary of all the models in the RCM2100 family.

Table 1. RCM2100 Versions

Feature	RCM2100	RCM2110	RCM2120	RCM2130
Microprocessor	Rabbit 2000 running at 22.1 MHz			
Flash Memory	512k	128k	512k	128k
Static RAM	512k	256k	512k	256k
General-Purpose I/O	34	34	40	40
Ethernet	RJ-45	RJ-45	None	None
Serial Ports	4, high-speed, CMOS-compatible; 2 configurable as clocked ports; 1 clocked port dedicated to programming port use.			

1.1.1 Standard Ethernet Versions

There are two RCM2100 series modules that incorporate an Ethernet port:

RCM2100. The RCM2100 is the most fully-equipped module in the family, with the Ethernet port, 512k flash memory and 512k static RAM. The Ethernet port uses portions of two of the Rabbit 2000 microprocessor's parallel ports, reducing the available number of I/O pins to 34. This is the version included in the Development Kit.

RCM2110. The RCM2110 is identical to the RCM2100 except that it is equipped with 128k of SRAM and 256k of flash memory.

1.1.2 Standard Non-Ethernet Versions

To accommodate developers and users who want the RCM2100's footprint and capabilities other than the integrated Ethernet port, two standard versions of the module areavailable without the Ethernet hardware:

RCM2120. The RCM2120 is equipped with 512k flash memory and 512k static RAM, but does not include the Ethernet port hardware. In its place, ports D and E of the Rabbit 2000 microprocessor are enabled, giving this module 40 I/O pins.

RCM2130. The RCM2130 is identical to the RCM2120 except that it is equipped with 128k of SRAM and 256k of flash memory.

1.1.3 Other Factory Versions

To further accommodate developers with specific needs, an additional version of the RCM2100 module is available on special order:

RCM2115. The RCM2115 is equipped with 256k flash memory and 128k static RAM as well as the Ethernet port. This version does not include the RJ-45 jack and associated interface transformers. The Ethernet signals are brought to a header for use by the production system. (Users will have to provide the interface components within their design.)

In addition to these standard models, production quantities of modules with custom configurations of memory, I/O and Ethernet capability can be ordered.

1.1.4 Physical & Electrical Specifications

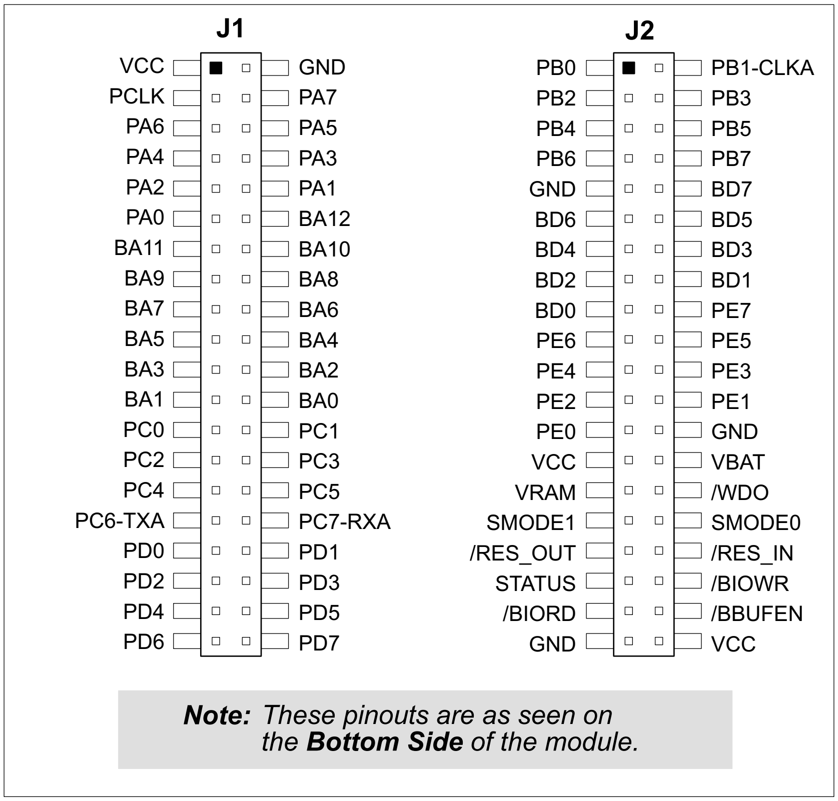
Table 2 lists the basic specifications for all models in the RCM2100 series.

Table 2. RCM2100 Specifications

Specification	Data
Power Supply	4.75–5.25 V DC (140 mA at 22.1 MHz clock speed)
Size	2.0" × 3.5" × 0.85" (51mm × 89 mm × 22 mm)
Environmental	0°C to 70°C, 5–95% humidity, noncondensing

NOTE: For complete product specifications, see Appendix A in the ***RabbitCore RCM2100 User's Manual***.

The RCM2100 modules have two 40-pin headers to which cables can be connected, or which can be plugged into matching sockets on a production device. The pinouts for these connectors are shown in Figure 1 below.



Note: *These pinouts are as seen on the **Bottom Side** of the module.*

Figure 1. RCM2100 Pinout

1.2 Development Software

The RCM2100 series of modules 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., “Software Installation & Overview,” for complete information on installing Dynamic C.

> **NOTE:** The RCM2100 series modules require Dynamic C v7.04 or later for development. A compatible version is included on the Development Kit CD-ROM.

1.3 How to Use This Manual

This *Getting Started* manual is intended to give users a quick but solid start with the RCM2100 series modules. It does not contain detailed information on the module hardware capabilities, the Dynamic C development environment, or the TCP/IP software support for the integrated Ethernet port. Most users will want more detailed information on some or all of these topics in order to put the RCM2100 module to effective use.

1.3.1 Additional Product Information

Detailed information about the RCM2100 series will be found in the *RabbitCore RCM2100 User's Manual*, provided on the accompanying CD-ROM in both HTML and Adobe PDF format.

Some advanced users may choose to skip the rest of this introductory manual and proceed directly with the detailed hardware and software information in the User's Manual.

> **NOTE:** We recommend that anyone not thoroughly familiar with Z-World controllers at least read through the rest of this manual to gain the necessary familiarity to make use of the more advanced information.

1.3.2 Additional Reference Information

In addition to the product-specific information contained in the ***RabbitCore RCM2100 User's Manual***, 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 RCM2100 series modules:

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

1.3.3 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 choose to provide our complete documentation and reference library in electronic form with every development kit and with our Dynamic C development environment.

> **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.

Providing this documentation in electronic form saves an enormous amount of paper by not printing copies of manuals that users don’t need. It reduces the number of outdated manuals we have to discard from stock as well, and it makes providing a complete library of manuals an almost cost-free option. For one-time or infrequent reference, electronic doc uments are more convenient than printed ones—after all, they aren’t taking up shelf or desk space!

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.
- Print only the sections you will need to refer to more than once.
- 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.
- 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 Setup

This chapter describes the RCM2100 series hardware in more detail, and explains how to set up and use the accompanying prototyping and development board.

NOTE: This chapter (and this manual) assume that you have the RabbitCore RCM2100 Development Kit. If you purchased an RCM2100 series module by itself, you will have to adapt the information in this chapter and elsewhere to your test and development setup.

2.1 Development Kit Contents

The RCM2100 Development Kit contains the following items:

- RCM2100 module with Ethernet port, 512K flash memory and 512K SRAM.
- RCM2100 Prototyping Board with accessory hardware and components.
- Wall transformer power supply, 12 V DC, 500 mA. (Included only with Development Kits sold for the North American market. Overseas users will have to substitute 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 Overview of the Prototyping Board

The Prototyping Board included in the Development Kit makes it easy to connect an RCM2100 module to a power supply and a PC workstation for development. It also provides an array of basic I/O peripherals (switches and LEDs), as well as a prototyping area for more advanced hardware development.

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

As you progress to more sophisticated experimentation and hardware development, modifications and additions can be made to the board without modifying or damaging the RCM2100 module itself.

The Prototyping Board is shown in Figure 2 below, with its main features identified

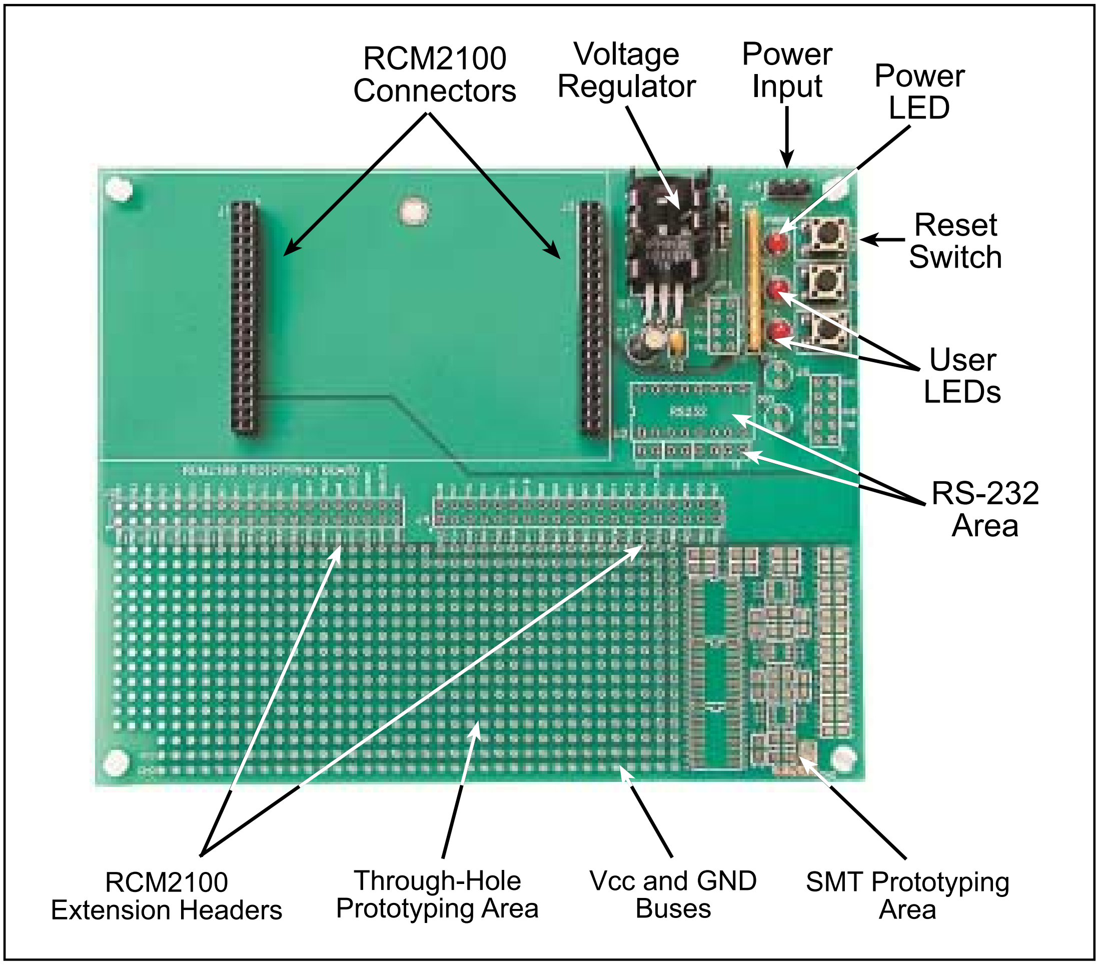


Figure 2. RCM2100 Prototyping Board

2.2.1 Prototyping Board Features

Power Connection. A 3-pin header is provided for connection of a power supply. Note that it is symmetrical, with both outer pins connected to ground and the center pin connected to the raw V+ input. The cable of the wall transformer provided with the North American version of the Development Kit ends in a connector that is correctly connected in either orientation.

Users providing their own power supply should ensure that it delivers 9–24 V DC at not less than 500 mA. The voltage regulator will get warm in use, but lower supply voltages will reduce thermal dissipation from the device.

Regulated Power Supply. The raw DC voltage provided at the POWER IN jack is routed to a 5 V linear voltage regulator, which provides stable power to the RCM2100 module and the Prototyping Board. A Shottky diode protects the power supply against damage from reversed raw power connections.

Power LED. The power LED lights whenever power is connected to the Prototyping Board.

Reset Switch. A momentary-contact, normally open switch is connected directly to the RCM2100's **/RES_IN** pin. Pressing the switch forces a hardware reset of the system.

I/O Switches & LEDs. Two momentary-contact, normally open switches are connected to the PB2 and PB3 pins of the RCM2100 module, and may be read as inputs by sample applications.

Two LEDs are connected to the PA0 and PA1 pins of the module, and may be driven as output indicators by sample applications. (Two more LEDs, driven by PA2 and PA3, may be added to the Prototyping Board for additional outputs.)

All the LEDs are connected through JP1, which has traces shorting adjacent pads together. These traces may be cut to disconnect the LEDs, and an 8-pin header soldered into JP1 to permit their selective reconnection with jumpers. See Figure 3 for details.

Expansion Areas. The Prototyping Board is provided with several unpopulated areas for expansion of I/O and interfacing capabilities. See the next section for details.

Prototyping Area. A generous prototyping area has been provided for the installation of through-hole components. Vcc (5 V DC) and Ground buses run around the edge of this area. An area for surface-mount devices is provided to the right of the through-hole area. (Note that there are SMT device pads on both top and bottom of the Prototyping Board.)

2.2.2 Prototyping Board Expansion

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

Module Extension Headers. The complete pin set of the RCM2100 module is duplicated at these two headers. Developers can solder wires directly into the appropriate holes, or, for more flexible development, two 40-pin header strips can be soldered into place. See Figure 1 on page 3 for the header pinouts.

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

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

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

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

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

To disconnect these devices and permit the pins to be used for other purposes, cut the traces between the pin rows. Use an exacto knife or similar tool to cut or break the traces crossing JP1, in the area indicated in Figure 3.

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

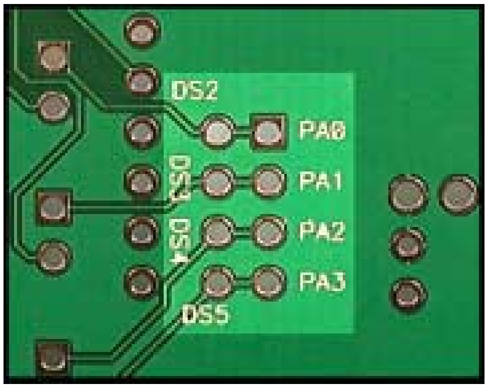


Figure 3. Where to Cut Traces to Permanently Disable Demonstration Hardware on Prototyping Board

2.3 Development Hardware Connections

There are four steps to connecting the prototyping board for use with Dynamic C and the sample programs:

1. Attach the RCM2100 module to the Prototyping Board.
2. Connect the programming cable between the RCM2100 module and the workstation PC.
3. Connect the module’s Ethernet port to a PC’s Ethernet port, or to an Ethernet network.
4. Connect the power supply to the Prototyping Board.

2.3.1 Attach Module to Prototyping Board

Turn the RCM2100 module so that the Ethernet connector is on the left, as shown in Figure 4 below. Align the module headers J1 and J2 into sockets J1 and J3 on the Prototyping Board.

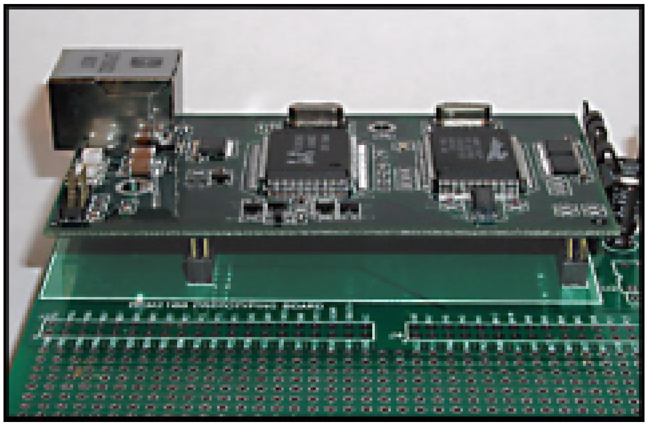


Figure 4. Installing the RCM2100 Module on the Prototyping Board. Note the orientation of the module.

NOTE: It is important that you line up the RCM2100 pins on headers J1 and J2 exactly with the corresponding pins of headers J1 and J3 on the Prototyping Board. The header pins may become bent or damaged if the pin alignment is offset, and the module will not work.

Press the module's pins firmly into the Prototyping Board headers. The installed module is shown in Figure 5 below.

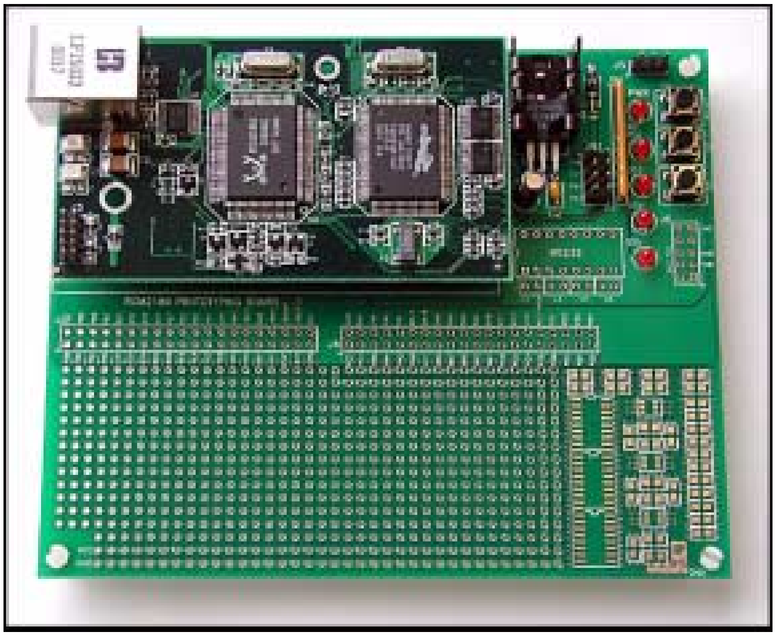


Figure 5. RCM2100 Installed and Seated on the Prototyping Board

2.3.2 Connect Programming Cable

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

Connect the 10-pin connector of the programming cable labeled **PROG** to header J3 on the RCM2100 module as shown in Figure 6 below. Be sure to orient the red edge of the cable towards pin 1 of the connector. (Do not use the **DIAG** connector, which is used for a normal serial connection.)

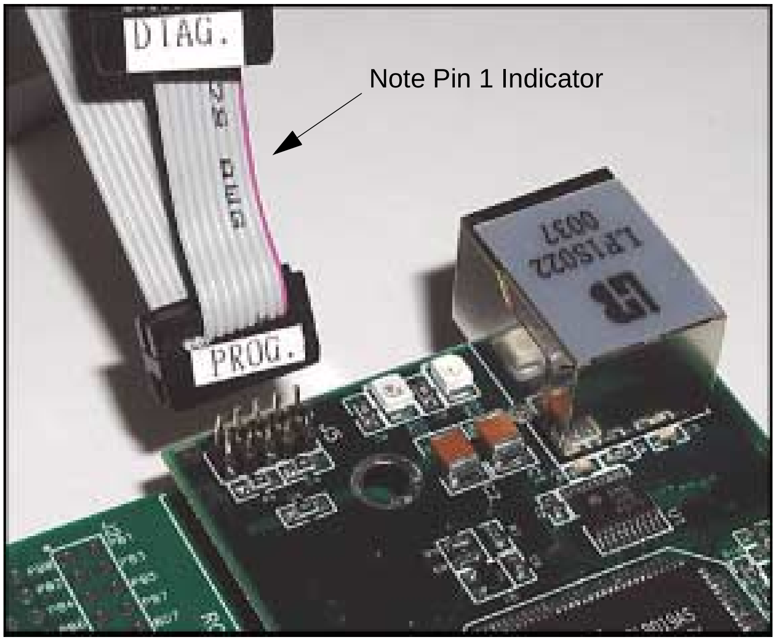


Figure 6. Attaching Programming Cable to the RCM2100

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

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

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

2.3.3 Connect Ethernet Network Cable

Programming and development can be done with the RCM2100 without connecting the Ethernet port to a network. However, if you will be running the sample programs that use the Ethernet capability or will be doing Ethernet-enabled development, you should connect the RCM2100's Ethernet port at this time.

There are four options for connecting the RCM2100 module to a network for development and runtime purposes. The first two options permit total freedom of action in selecting network addresses and use of the "network," as no action can interfere with other users. We recommend one of these options for initial development.

- **No LAN —** The simplest alternative for desktop development. Connect the RCM2100's Ethernet port directly to the workstation's network interface card using an RJ-45 *crossover cable*. A crossover cable is a special cable that flips some connections between the two connectors and permits direct connection of two client systems. A standard RJ-45 network cable will not work for this purpose.
- **Micro-LAN —** Another simple alternative for desktop development. Use a small Ethernet 10Base-T hub and connect both the workstation's network interface card and the RCM2100's Ethernet port to it, using standard network cables.

The following options require more care in address selection and testing actions, as conflicts with other users, servers and systems can occur:

- **LAN —** Connect the RCM2100's Ethernet port to an existing LAN, preferably one to which the development workstation is already connected. You will need to obtain IP addressing information from your network adminstrator.
- **WAN —** The RCM2100 series is capable of direct connection to the Internet and other Wide Area Networks, but exceptional care should be used with IP address settings and all network-related programming and development. We recommend that development and debugging be done on a local network before connecting a RabbitCore system to the Internet.

> **NOTE:** Checking and debugging the initial setup on a micro-LAN is recommended before connecting the system to a LAN or WAN.

2.3.4 Connect Power

When all other connections have been made, you can connect power to the RCM2100 Prototyping Board.

Hook the connector from the wall transformer to header J5 on the Prototyping Board as shown in Figure 7 below. The connector may be attached either way as long as it is not offset to one side.

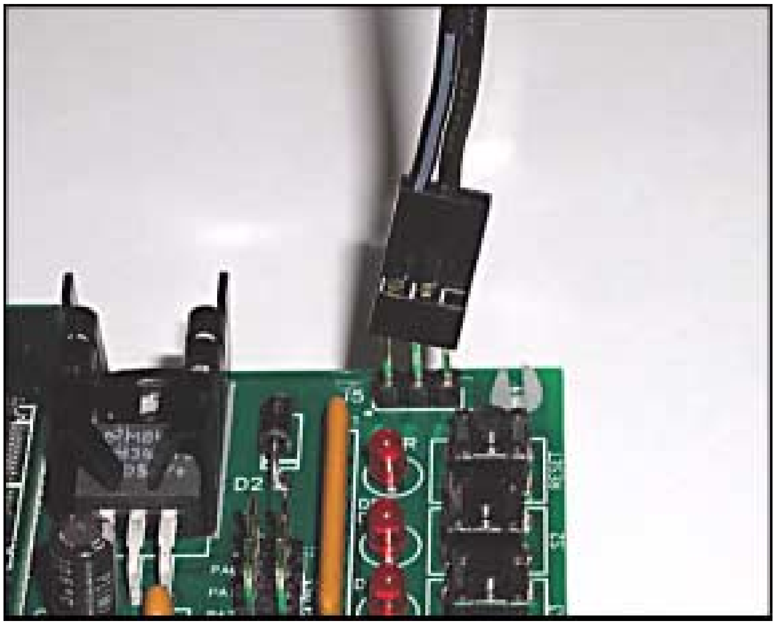


Figure 7. Power Supply Connections to Prototyping Board

Plug in the wall transformer. The power LED on the Prototyping Board should light up. The RCM2100 and the Prototyping Board are now ready to be used.

> **NOTE:** A **RESET** button is provided on the Prototyping Board to allow hardware reset without disconnecting power.

To power down the Prototyping Board, unplug the power connector from J5. You should disconnect power before making any circuit adjustments in the prototyping area, changing any connections to the board, or removing the RCM2100 module from the Prototyping Board.

2.4 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 RabbitCore module and prototyping board are 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 RabbitCore module's capabilities.
2. For further development, refer to the ***RabbitCore RCM2100 User's Manual*** for details of the module's hardware and software components.

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

2.4.1 Technical Support

If you encounter any problems, call our Technical Support center:

- Z-World Technical Support, (530) 757-3737
- Rabbit Semiconductor Technical Support, (530) 757-8400

3. Software Installation & Overview

To develop and debug programs for the RCM2100 series (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 RCM2100 series.

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 for development of RCM2100-based systems 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 10Base-T Ethernet network interface port
  (optional if you will not be developing Ethernet-based systems)
- A CD-ROM drive (for software installation)

3.3 Installing Dynamic C

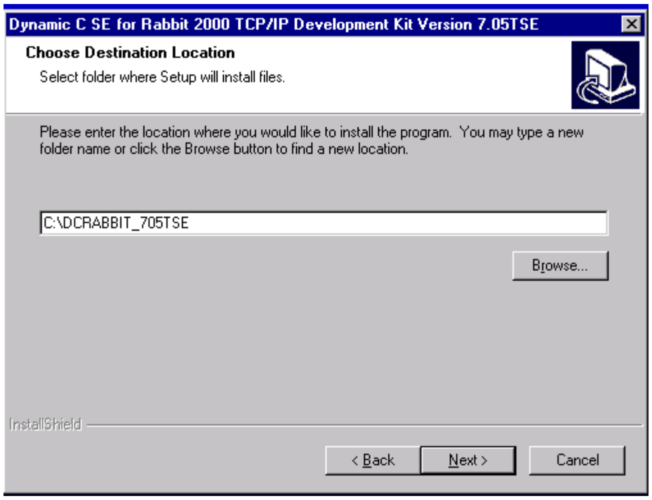
Insert the Dynamic C CD-ROM in the drive on your PC. If autorun is enabled, the CD installation will begin automatically.

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.



Dynamic C SE for Rabbit 2000 TCP/IP Development Kit Version 7.05TSE

Choose Destination Location

Select folder where Setup will install files.

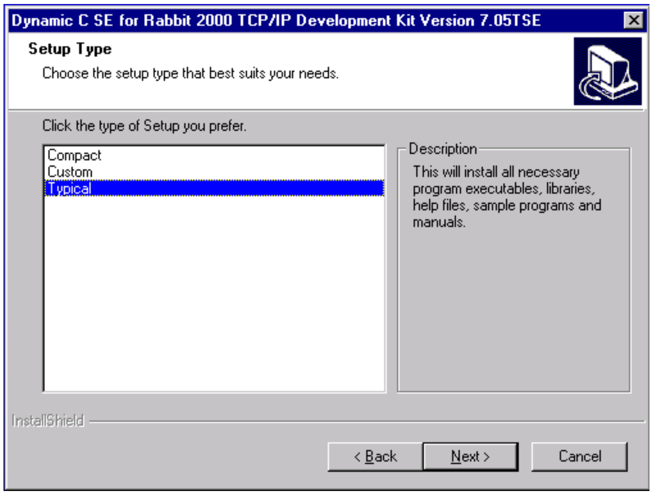
Please enter the location where you would like to install the program. You may type a new folder name or click the Browse button to find a new location.

C:\DCRABBIT_705TSE

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).



Dynamic C SE for Rabbit 2000 TCP/IP Development Kit Version 7.05TSE

Setup Type

Choose the setup type that best suits your needs.

Click the type of Setup you prefer.

Compact
Custom
Typical

Description

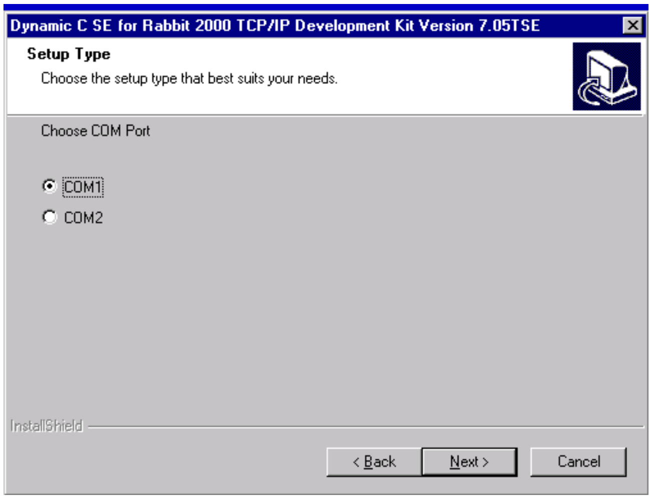
This will install all necessary program executables, libraries, help files, sample programs and manuals.

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.



Dynamic C SE for Rabbit 2000 TCP/IP Development Kit Version 7.05TSE

Setup Type

Choose the setup type that best suits your needs.

Choose COM Port

● COM1

○ COM2

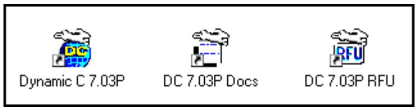
InstallShield
< Back
Next >
Cancel

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 RabbitCore module 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 and loaded…"`** you are ready to continue with the sample programs in the next chapter.

3.4.1 Communication Error Messages

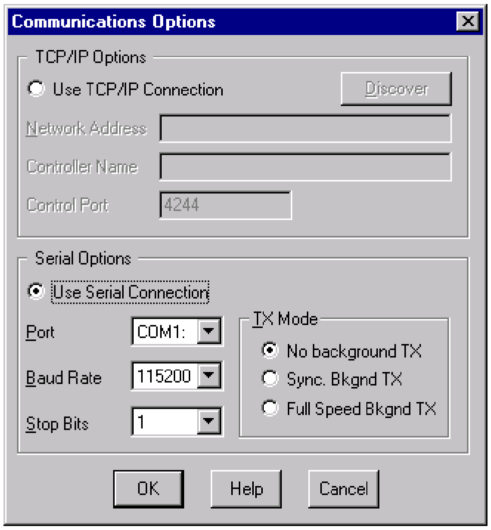
If you receive the message **`"No Rabbit Processor Detected,"`** the programming cable may be connected to a different COM port, a connection may be faulty, or the target system may not be powered up. First, check to see that the power LED on the prototyping board is lit. If it is, check both ends of the programming cable to ensure that it is firmly plugged into the PC and the RCM2100’s programming port. If you are using the Prototyping Board, ensure that the module is firmly and correctly installed in its connectors.

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.



Communications Options

TCP/IP Options

○ Use TCP/IP Connection

Network Address

Controller Name

Control Port 4244

Serial Options

● Use Serial Connection

Port COM1:

Baud Rate 115200

Stop Bits 1

TX Mode

● No background TX

○ Sync. Bkgnd TX

○ Full Speed Bkgnd TX

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.

4. Sample Programs

To help familiarize you with the RabbitCore RCM2100 modules, several sample Dynamic C programs have been included. Loading, executing and studying these programs will give you a solid hands-on overview of the RabbitCore's capabilities, as well as a quick start with Dynamic C as an application development tool.

4.1 Sample Program Overview

Dynamic C comes with a large number of sample programs that illustrate many of its features. These programs are intended to serve as tutorials, but then can also be used as starting points or building blocks for your own applications.

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

We have selected five of these sample programs to take you through a complete tour of the capabilities of the RCM2100 modules. They form a learning arc from basic I/O control to advanced TCP/IP issues, including Web serving:

- **`FlashLED.c`**
- **`ToggleLED.c`**
- **`FlashLEDs.c`**
- **`PingLED.c`**
- **`EthCore1.c`**

Once you have loaded and executed these five programs and have an understanding of how Dynamic C and the RCM2100 modules interact, you can move on and try the other sample programs, or begin building your own.

4.2 Loading and Compiling Programs in Dynamic C

This section gives you a short summary of how to connect the target system, start Dynamic C, and load a sample program. For more details on these topics, refer to Sections 2 and 3 of this manual, and the ***Dynamic C Premier User's Manual***.

4.2.1 Connect Prototyping Board

Section 2.3 provides detailed instructions for setting up the RCM2100 Prototyping Board and making its hardware connections. In summary:

- Install the RCM2100 module on the Prototyping Board;
- Connect the **PROG** connector of the programming cable to J5 of the RCM2100, and the other end of this cable to your PC's COM port;
- Connect an Ethernet cable to the module's RJ-45 jack, and to a LAN or micro-LAN hub;
- Connect the power supply to the Prototyping Board.

The **PWR** LED should light up. If it does not, see Section 2.3.

4.2.2 Start Dynamic C

Section 3.4 of this manual provides detailed instructions for installing and starting up Dynamic C.

- Double-click on the Dynamic C icon.

Dynamic C should start and compile the BIOS, completing with a message that reads **`"BIOS successfully compiled and loaded."`** If it does not, see Section 3.4.

4.2.3 Load Program

In Dynamic C, use the **File | Open** menu item to open the file explorer, and double-click on the **Samples** folder. In that folder, double-click on the **RCM2100** folder. In that folder, double-click on the file **`FlashLED.c`**.

4.2.4 Compile & Run Program

With **`FlashLED.c`** open in a Dynamic C window, press **F9** or click on the **Run | Run** menu item. Dynamic C should compile the program, and when it finishes, LED DS3 on the Prototyping Board should begin to flash.

Congratulations! You've just compiled and run your first Dynamic C program on the RCM2100 module!

> **NOTE:** For a quick tutorial on using Dynamic C's editing, compiling and debugging features, see Section 3 in the ***Dynamic C Premier User's Manual***.

4.3 Sample Program: FlashLED.c

If you did not load and compile **`FlashLED.c`** in the prior section, refer to Sections 4.2.3 and 4.2.4.

Program Description

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

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

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

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

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

> **NOTE:** Since the variable **`j`** is defined as type **`int`**, the range for **`j`** must be between 0 and 32768. To permit larger values and thus longer delays, change the declaration of **`j`** to **`unsigned int`** or **`long`**.

More Information

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

4.4 Sample Program: ToggleLED.c

One of Dynamic C's unique and powerful aspects is its ability to efficiently multitask using *cofunctions* and *costatements*. This simple application demonstrates how these program elements work.

Compile & Run Program

Open the source file **`ToggleLED.c`**, located in the **Samples\RCM2100** folder. Press **F9** to compile and run the program.

The LED DS3 on the Prototyping Board will begin blinking. Press switch S2 to toggle LED DS2 on and off.

Program Description

This program uses two costatements to set up and manage the two tasks. Costatements must be contained in a loop that will "tap" each of them at regular intervals. This program:

1. Initializes the pins of Port A as outputs.
2. Sets all the pins of Port A high, turning off the attached LEDs.
3. Sets the toggled LED status variable **`vswitch`** to 0 (LED off).
4. Starts an endless loop using a **`while(1)`** expression, and within that loop:
    - Executes a costatement that flashes LED DS3;
    - Executes a costatement that checks the state of switch S2 and toggles the state of **`vswitch`** if it is pressed;
    - Turns LED DS2 on or off, according to the state of **`vswitch`**.

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

The first costatement is a compressed version of **`FlashLED.c`**, with slightly different flash timing. It also uses the library function **`DelayMs()`** to deliver more accurate timing than the simple delay loops of the previous program.

The second costatement does more than check the status of S2. Switch contacts often "bounce" open and closed several times when the switch is actuated, and each bounce can be interpreted by fast digital logic as an independent press. To clean up this input, the code in the second costatement "debounces" the switch signal by waiting 50 milliseconds and checking the state of the switch again. If it is detected as being closed both times, the program considers it a valid switch press and toggles **`vswitch`**.

Unlike most C statements, the two costatements are not executed in their entirety on each iteration of the **`while(1)`** loop. Instead, the list of statements within each costatement is initiated on the first loop, and then executed one "slice" at a time on each successive interation. This mode of operation is known as a *state machine*, a powerful concept that permits a single processor to efficiently handle a number of independent tasks.

The ability of Dynamic C to manage state machine programs enables you to create very powerful and efficient embedded systems with much greater ease than other programming methods.

More Information

See the entries for the **`DelayMs()`** function, as well as Section 5, “Multitasking with Dynamic C,” in the ***Dynamic C Premier User’s Manual***.

4.5 Sample Program: FlashLEDs.c

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

Compile & Run Program

Open the source file **`FlashLEDs.c`**, located in the **Samples\RCM2100** folder. Press **F9** to compile and run the program.

All the LEDs on the Prototyping Board (including DS4 and DS5, if you have installed them) will light. DS2 and DS3 will begin flashing at different rates, and will continue doing so until the program is interrupted.

Program Description

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

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

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

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

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

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

1. Calls the library function **`hitwd()`**, which resets the microprocessor's watchdog timer. (If the watchdog timer is not reset every so often, it will force a hard reset of the system. The purpose is to keep an intermittent program or hardware fault from locking up the system. Normally, this function is taken care of by the Virtual Driver, but it is called explicitly here).

2. Sets up a costatement which calls two instances of the **`flashled()`** function, one for each LED. Note that one LED is flashed one second on, one-half second (500 ms) off, while the other is flashed in the reverse pattern.

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

More Information

See the entries for the **`hitwd()`** and **`DelayMs()`** functions in the ***Dynamic C Premier User's Manual***, as well as those for the directives **`#asm`** and **`#endasm`**. For a complete explanation of how Dynamic C handles multitasking with costatements and cofunctions, see Chapter 5, "Multitasking with Dynamic C," and Chapter 6, "The Virtual Driver," in the ***Dynamic C Premier User's Manual***.

4.6 Sample Program: PingLED.c

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

Compile & Run Program

Open the source file **PingLED.c**, located in the **Samples\RCM2100** folder. Edit the six **#define** values near the beginning of the file to represent valid network addresses for your setup.

Press **F9** to compile and run the program.

> **NOTE:** The RCM2100 must be connected to a network as described in Section 2.3.3, "Connect Ethernet Network Cable," in order for this program to work.

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

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

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

Program Description

For operation, network addresses must be correctly defined at the start of this program. The most important address to set correctly is **MY_IP_ADDRESS**, which is the address of the RCM2100 module. (The **MY_NETMASK** address must be correct as well, but the default of 255.255.255.0 is almost universally used.)

If you wish to ping systems outside the local network, you will have to correctly define the **MY_GATEWAY** address as well. If you wish to ping systems using domain names instead of IP addresses, a valid DNS server address must be defined for **MY_NAMESERVER**.

The IP address to be pinged is defined by **PING_WHO**. You will have to change this address and recompile the program to ping different addresses. (In most real-world applications, there should be some mechanism by which to dynamically define or select addresses.) This address may be defined as a numeric IP address. If a gateway to the Internet and a valid DNS server are specified, this definition may also be a fully-qualified domain name (such as "www.zworld.com").

The program first defines three functions to control the LEDs—one to initialize them, and then one each to drive the "ping out" and "ping in" LEDs.

The program begins by calling the LED initialization function **pingleds_setup( )**. More importantly, it then calls **sock_init( )**, which initializes the packet driver and the TCP manager using the compiler defaults. This function must always be called before any other TCP/IP functions.

The program then resolves the address to be pinged into a numeric value. using the library function **resolve()**. If the defined address is numeric, it converts the define string into truly numeric form. If the address is a domain name, the function queries the indicated DNS server to obtain the numeric address. (If the function is unable to resolve the address—if, for example, the numeric address is incomplete or badly formed, or the DNS server is unable to identify the domain name—the program will print a message to the screen and terminate.)

The program then begins an endless loop using **for(;;)**. Within this loop, the program executes the following steps:

1. Calls **tcp_tick( )** to perform the basic housekeeping functions for the socket;
2. As a costatement, waits for the duration of **PING_DELAY** (defined by default as 500 mS or one-half second), issues a ping to the resolved address using the **_ping()** function, and flashes LED DS2;
3. As a second costatement, checks for a ping return using the **_chk_ping()** function. If the ping is successful, the costatement flashes LED DS3.

If you uncomment the **#VERBOSE** define near the beginning of the program, the ping return costatement will also print a message to the screen indicating each successful ping.

More Information

Refer to the ***Dynamic C TCP/IP Software User's Manual*** for complete details on the Dynamic C implementation of TCP/IP protocols.

4.7 Sample Program: EthCore1.c

The RCM2100 modules with Ethernet ports can act as micro Web page servers, with dynamic interaction between the controller and the web pages. This sample program demonstrates how a web page can be used to both monitor and control an RCM2100 module.

Compile & Run Program

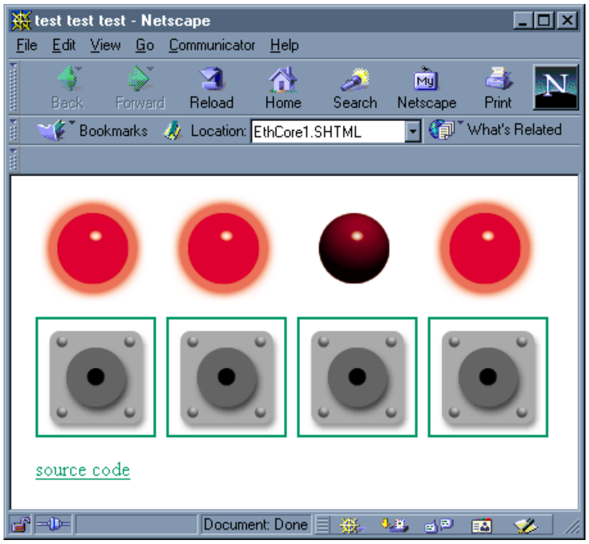
Open the source file **`EthCore1.c`**, located in the **Samples\RCM2100** folder. Edit the three **`#define`** values near the beginning of the file to represent valid network addresses for your setup.

Press **F9** to compile and run the program.

> **NOTE:** The RCM2100 must be connected to a network as described in Section 2.3.3, ”Connect Ethernet Network Cable,” in order for this program to work.

> **TIP:** This program will be more interesting to observe if LEDs DS4 and DS5 are installed in the Prototyping Board.

When the program starts, LEDs DS2, DS3 and DS5 will be lit, and DS4 will be dark. Open a web browser and enter the IP address you defined for the RCM2100 module in the program in the address window. A page like that shown in Figure 8 should appear.


test test test - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print
Bookmarks
What's Related
source code
Document: Done

Location: EthCore1.SHTML

Figure 8. Browser screen for Sample Program `EthCore1.c`.

Clicking on each of the button images in the browser window will toggle the state of the associated LED image, and will toggle the state of the corresponding LED on the Prototyping Board. Since the web page is generated by the RabbitCore module (using Dynamic HTML), the LED image and the corresponding LED’s real state will always be in step.

Program Description

This program begins to show the range of applications for an Ethernet-enabled embedded system controller, so let's look closely at its operation.

As with **`PingLED.c`**, several network addresses must be defined before this application can work. The most important is again **`MY_IP_ADDRESS`**, which defines the RCM2100 system's IP address. The netmask value should be set if the default of 255.255.255.0 is not correct.

If you are using the system on a local network and will not be trying to access it from outside that network, the **`MY_GATEWAY`** value does not matter. If you want to be able to reach the system from outside the local network, you must specify a valid gateway address.

Generally, the other defined values may be left at their default settings. If you are operating the system behind a firewall or proxy and need to specify a host port for redirection, you should comment out the line reading:

```
#define REDIRECTHOST MY_IP_ADDRESS
```

Then uncomment the next line, which defines a specific redirection host and port:

```
#define REDIRECTHOST "my host.com:8080"
```

Be sure to enter the host port where indicated by **`"my host.com:8080".`**

This application creates dynamic HTML web pages on the fly. For simplicity, all of the web page components—shell HTML, image GIFs, etc.—are imported into flash memory using the **`#ximport`** statements. It is also possible to read these files from other locations, including the onboard flash file system, but this application keeps things simple by loading all the components into working memory.

The program then defines four instances of an LED toggling function, which are basic CGI functions that swap the values "ledon.gif" and "ledoff.gif" as the contents of the **`led`*`n`*** strings, and then force a reload of the web page to change the associated LED image. The physical LEDs on the Prototyping Board are turned on or off to match the **`led`*`n`*** strings displayed on the web page.

More Information

Refer to the ***Dynamic C TCP/IP Software User's Manual*** for complete details on the Dynamic C implementation of TCP/IP protocols.

4.8 Where Do I Go From Here?

The Dynamic C **`\Samples`** folders contain dozens of sample programs. Some of them are intended for other hardware with somewhat different characteristics, but most can be run on the RabbitCore RCM2100 series modules, either as-is or with some modification.

(All programs in the **`\Samples\RCM2100`** folder will run without changes on the RCM2100 series.)

We suggest that you continue exploring the sample programs, loading, running and modifying them to gain further insight into how to develop embedded system applications using these modules.

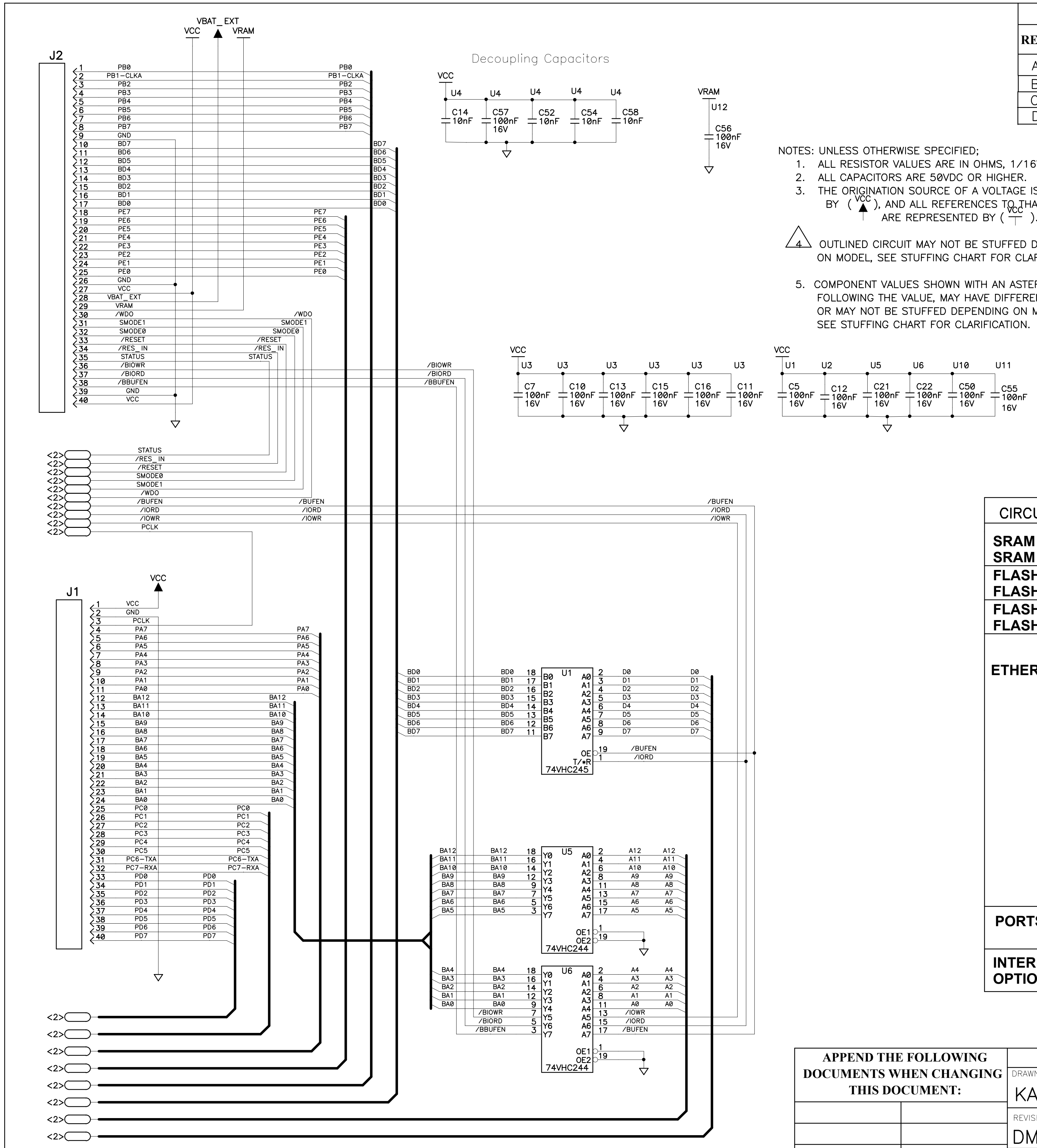
SCHEMATICS

090-0114 RCM2100 Schematic

090-0116 RCM2100 Prototyping Board Schematic

090-0128 Programming Cable Schematic

REVISION HISTORY			REVISION APPROVAL			
REV	ECO	DESCRIPTION OF CHANGE	PROJECT ENGINEER	APPROVAL DATE	DOCUMENT CONTROL	APPROVAL DATE
A	E11313	INITIAL RELEASE	DM	19Feb01	KIS	3/24/01
B	E11491	CORRECTED FLASH 2 SELECT TO JP2	DM	24Apr01	KIS	4/25/01
C	E11522	CHANGED C8, C9 TO 10pF	DM	14May01	KIS	5/14/01
D	E11580	MODIFIED VBAT CIRCUITRY, ADDED EXTERNAL 32KHz OSC	DM	17AUG01		



NOTES: UNLESS OTHERWISE SPECIFIED;

1. ALL RESISTOR VALUES ARE IN OHMS, 1/16W, 5%
2. ALL CAPACITORS ARE 50VDC OR HIGHER.
3. THE ORIGINATION SOURCE OF A VOLTAGE IS REPRESENTED BY (▲VCC), AND ALL REFERENCES TO THAT VOLTAGE ARE REPRESENTED BY (┬VCC).

△4 OUTLINED CIRCUIT MAY NOT BE STUFFED DEPENDING ON MODEL, SEE STUFFING CHART FOR CLARIFICATION.

5. COMPONENT VALUES SHOWN WITH AN ASTERISK (*) FOLLOWING THE VALUE, MAY HAVE DIFFERENT VALUES, OR MAY NOT BE STUFFED DEPENDING ON MODEL. SEE STUFFING CHART FOR CLARIFICATION.

TABLE A

REF DES	DEVICE	DEVICE VOLTAGE INFORMATION						DEVICE: FILTER CAP REF DES(s)
		AGND	GND	VCC	VRAM	uPBAT		
U1	74VHC245		10	20				C5
U2	ETC811L		2	4				C12
U3	RTL8019AS		14,28,44 52,83,86	6,17,47 57,70,89				C7,10,12,13,15,16
U4	RABBIT 2000		2,27,39 52,77,89	3,28,53, 78,92		42		C14, 57, 52, 54, 58 C25 – PIN42 (uPBAT)
U5	74VHC244		10	20				C21
U6	74VHC244		10	20				C22
U10	FLASH		24	8				C50
U11	FLASH		24	8				C55
U12	SRAM 512K X 8		15		32			C56
	SRAM 128K X 8		15		32			
	SRAM 32K X 8		14		28			

STUFFING TABLE

CIRCUIT	PART	RCM2100	RCM2110	*RCM2115	RCM2120	RCM2130
SRAM **SRAM select**	U12 JP3	512K 2–3	128K 1–2	128K 1–2	512K 2–3	128K 1–2
FLASH 1 **FLASH 1 select**	U10 JP1	256K 1–2	256K 1–2	256K 1–2	256K 1–2	256K 1–2
FLASH 2 **FLASH 2 select**	U11 JP2	256K 1–2	NOT INSTALLED	NOT INSTALLED	256K 1–2	NOT INSTALLED
ETHERNET OPTION	U3	INSTALLED	INSTALLED	INSTALLED	NOT INSTALLED	NOT INSTALLED
	J3	NOT INSTALLED	NOT INSTALLED	INSTALLED	NOT INSTALLED	NOT INSTALLED
	J4	INSTALLED	INSTALLED	NOT INSTALLED	NOT INSTALLED	NOT INSTALLED
	R1, R2 DS1, DS2	INSTALLED	INSTALLED	NOT INSTALLED	NOT INSTALLED	NOT INSTALLED
	R6, C1, C2, C3, C4	INSTALLED	INSTALLED	NOT INSTALLED	NOT INSTALLED	NOT INSTALLED
	R10, R11, R19 C1, C2, C3 C4, C7, C8 C9, C11, C12 C13, C15, C16	INSTALLED	INSTALLED	INSTALLED	NOT INSTALLED	NOT INSTALLED
	Y1	INSTALLED	INSTALLED	INSTALLED	NOT INSTALLED	NOT INSTALLED
PORTS D-E OPTION	R21, R24, R35 R36, R37, R38	NOT INSTALLED	NOT INSTALLED	NOT INSTALLED	INSTALLED	INSTALLED
INTERRUPT/JUMPER OPTIONS	R12, R18, R25 R58, R59	NOT INSTALLED	NOT INSTALLED	NOT INSTALLED	NOT INSTALLED	NOT INSTALLED

*Note: RCM2115 only availible for custom orders

08/17/01

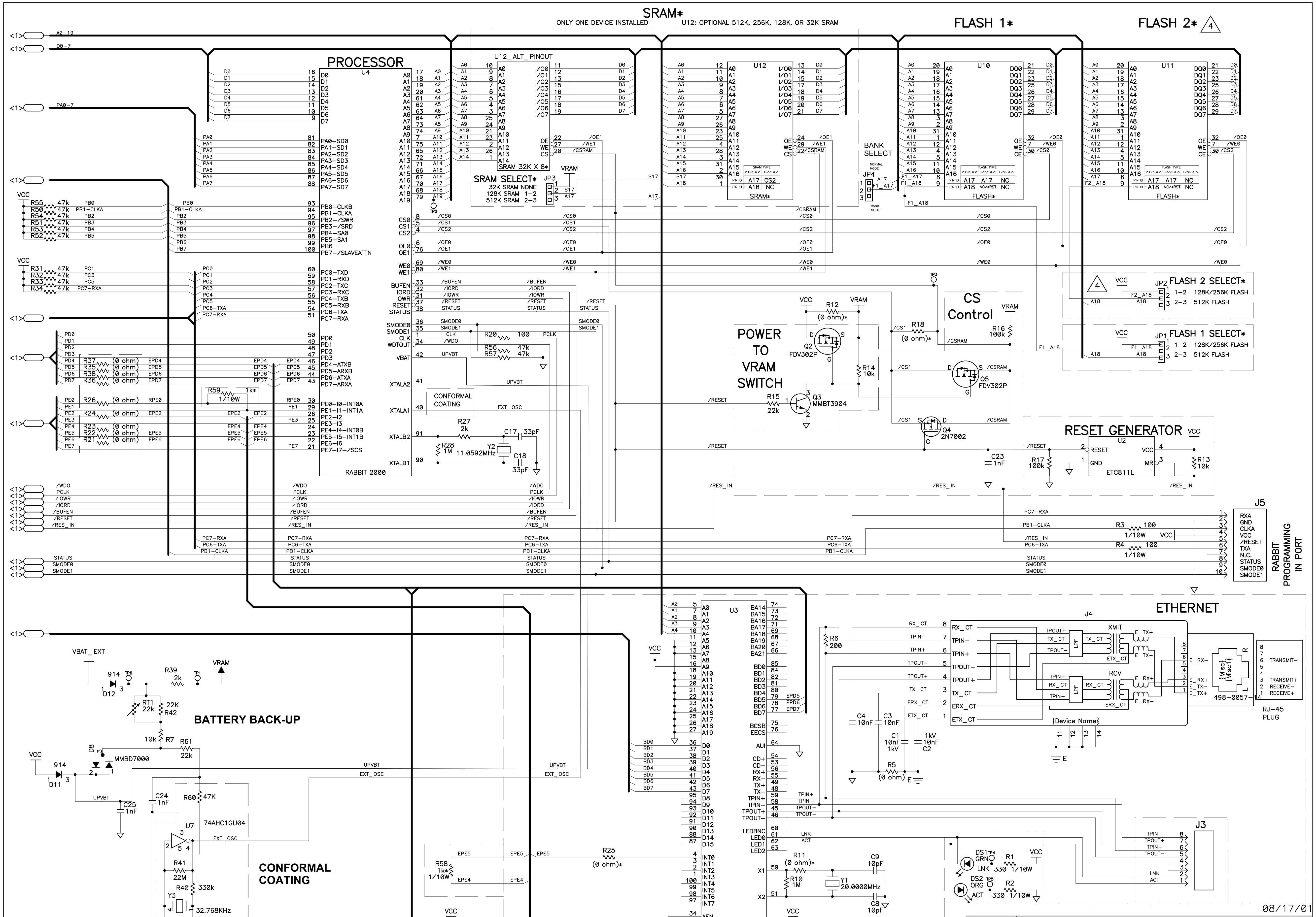
APPEND THE FOLLOWING DOCUMENTS WHEN CHANGING THIS DOCUMENT:		DRAWING CONTENT:	
		DRAWN BY: (INITIAL RELEASE) KAH	25JUL00
		REVISED BY: DM	08/17/01
		APPROVALS: INITIAL RELEASE	
		PROJECT ENGINEER:	
		ENGINEERING MANAGER:	
		SIGNATURES	DATE

TITLE

SCHEMATIC DIAGRAM
RCM2100 SERIES

Z-WORLD
2900 SPAFFORD ST.
DAVIS, CA 95616
530 - 757-4616

SIZE	DWG NO.		
C	090–0114		
SCALE NONE	RELEASE DATE 3/24/01	SHEET 1	OF 2



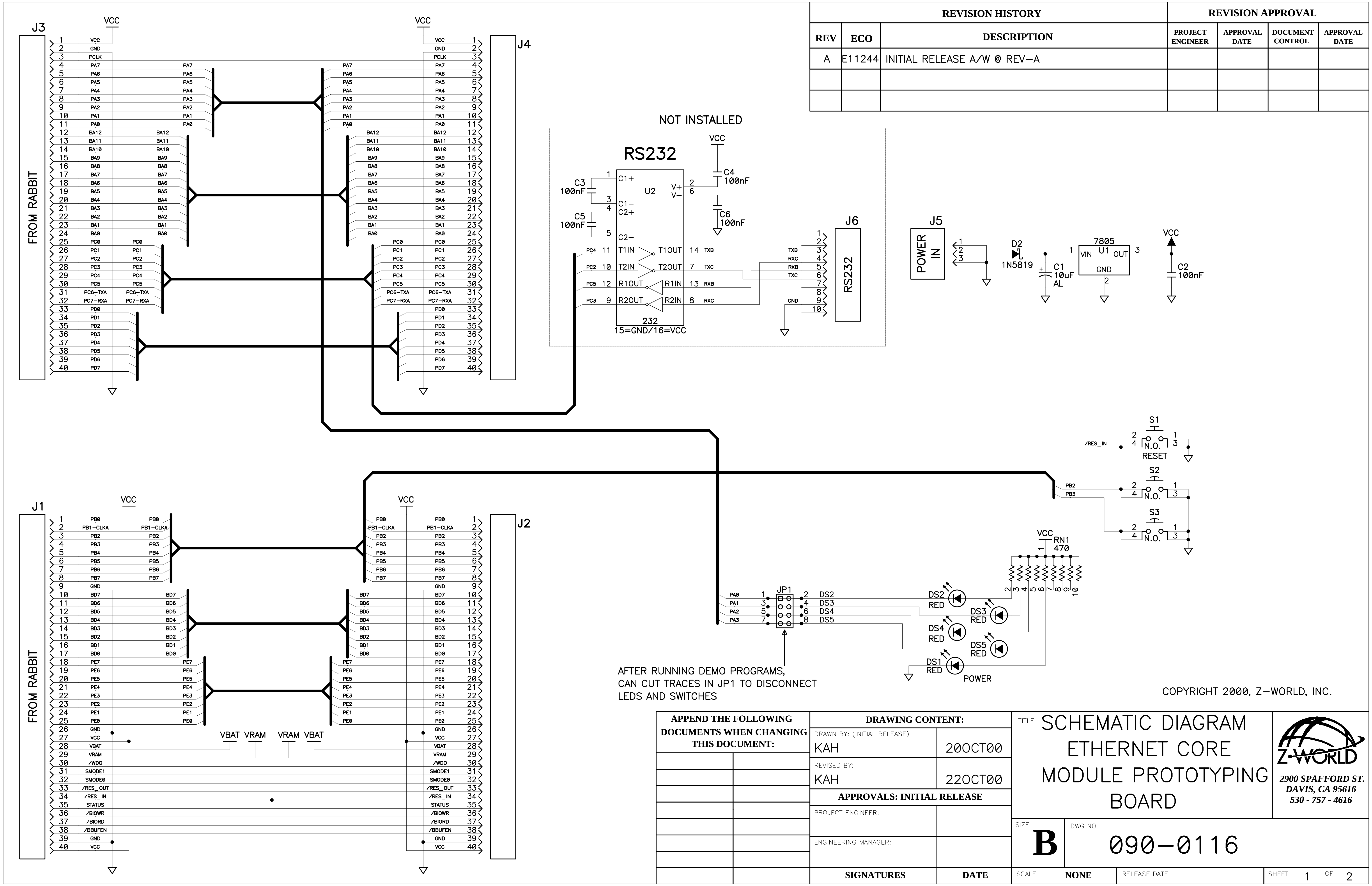
COPYRIGHT 2001, Z-WORLD, INC.

08/17/01

SIZE	DWG NO.
C	090–0114

SCALE	REV LTR	SHEET	OF
NONE	D	2	2

REVISION HISTORY			REVISION APPROVAL			
REV	ECO	DESCRIPTION	PROJECT ENGINEER	APPROVAL DATE	DOCUMENT CONTROL	APPROVAL DATE
A	E11244	INITIAL RELEASE A/W @ REV-A				


J3
FROM RABBIT
J4
J1
J2
VCC
VCC GND PCLK PA7 PA6 PA5 PA4 PA3 PA2 PA1 PA0
BA12 BA11 BA10 BA9 BA8 BA7 BA6 BA5 BA4 BA3 BA2 BA1 BA0
PC0 PC1 PC2 PC3 PC4 PC5 PC6-TXA PC7-RXA
PD0 PD1 PD2 PD3 PD4 PD5 PD6 PD7
PB0 PB1-CLKA PB2 PB3 PB4 PB5 PB6 PB7 GND
BD7 BD6 BD5 BD4 BD3 BD2 BD1 BD0
PE7 PE6 PE5 PE4 PE3 PE2 PE1 PE0
GND VCC VBAT VRAM /WDO SMODE1 SMODE0 /RES_OUT /RES_IN STATUS /BIOWR /BIORD /BBUFEN GND VCC
VBAT VRAM
VRAM VBAT
NOT INSTALLED
RS232
U2
C1+ C1– C2+ C2– V+ V–
C3 100nF
C5 100nF
C4 100nF
C6 100nF
T1IN T1OUT T2IN T2OUT R1OUT R1IN R2OUT R2IN
PC4 11 PC2 10 PC5 12 PC3 9
14 TXB 7 TXC 13 RXB 8 RXC
232
15=GND/16=VCC
J6
TXB RXC RXB TXC GND
RS232
J5
POWER IN
D2
1N5819
C1 10uF AL
7805
U1
VIN OUT GND
C2 100nF
S1
/RES_IN
N.O.
RESET
S2
PB2
PB3
S3
VCC
RN1 470
JP1
PA0 PA1 PA2 PA3
DS2 DS3 DS4 DS5
DS2 RED
DS3 RED
DS4 RED
DS5 RED
DS1 RED
POWER
AFTER RUNNING DEMO PROGRAMS,
CAN CUT TRACES IN JP1 TO DISCONNECT
LEDS AND SWITCHES

APPEND THE FOLLOWING DOCUMENTS WHEN CHANGING THIS DOCUMENT:		DRAWING CONTENT:	
		DRAWN BY: (INITIAL RELEASE) KAH	20OCT00
		REVISED BY: KAH	22OCT00
		APPROVALS: INITIAL RELEASE	
		PROJECT ENGINEER:	
		ENGINEERING MANAGER:	
		SIGNATURES	DATE

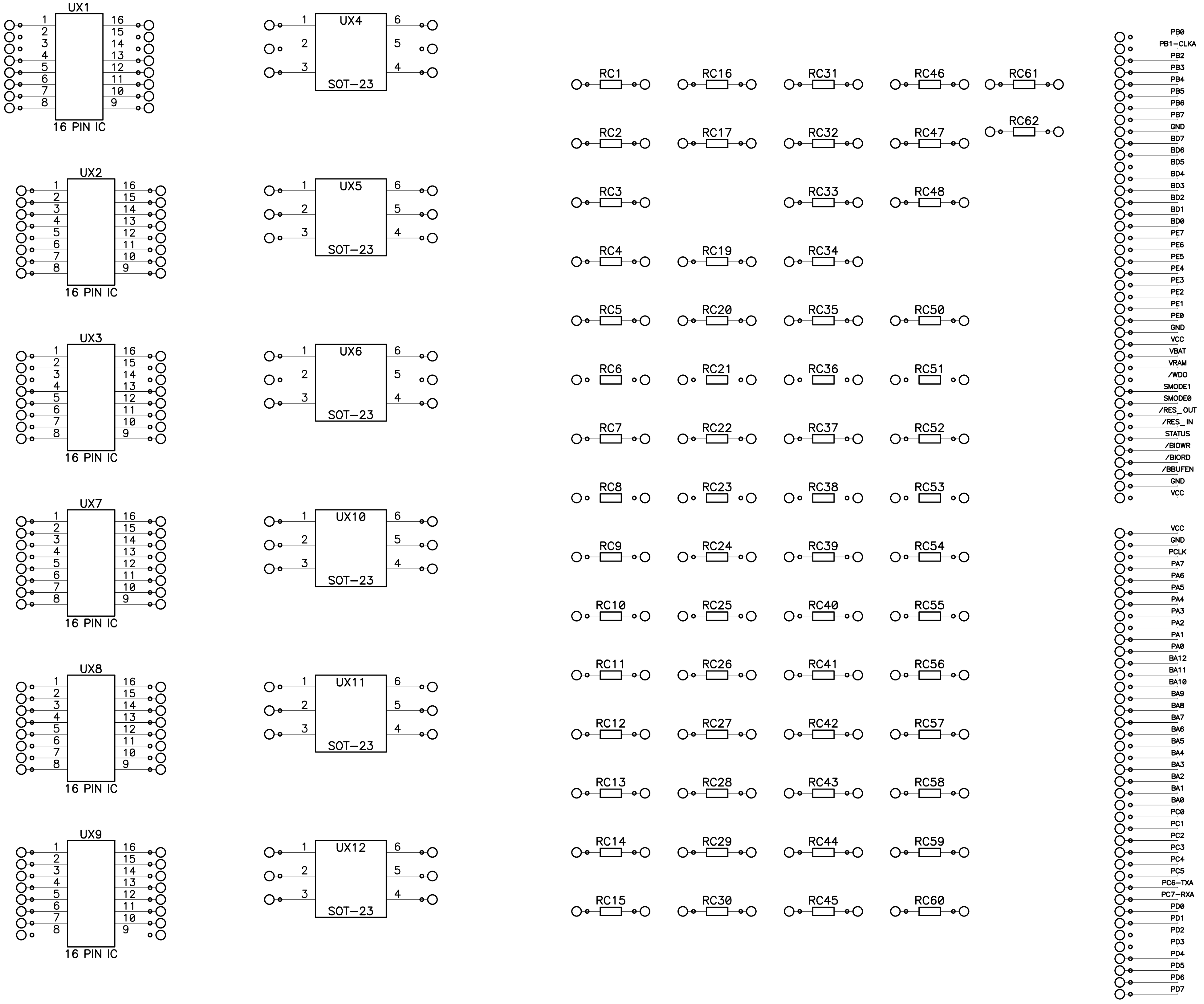
TITLE
SCHEMATIC DIAGRAM ETHERNET CORE MODULE PROTOTYPING BOARD

Z-WORLD
2900 SPAFFORD ST.
DAVIS, CA 95616
530 - 757 - 4616

SIZE	DWG NO.
B	090–0116

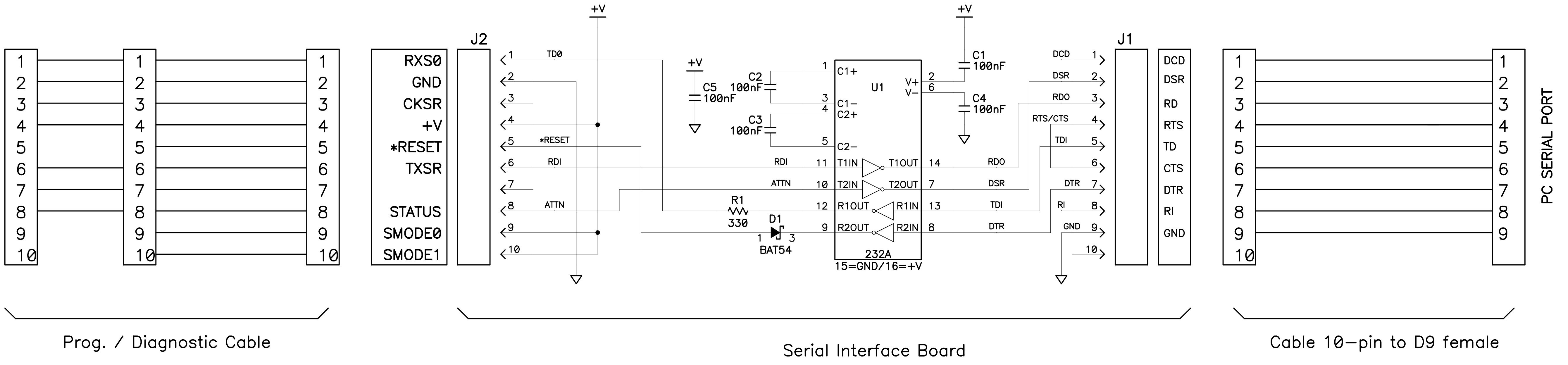
SCALE	NONE	RELEASE DATE		SHEET	1	OF	2

SURFACE MOUNT PROTOTYPING PADS



SIZE	DWG NO.			
B	090-0116			
SCALE NONE	REV LTR A		SHEET 2	OF 2

REVISION HISTORY			REVISION APPROVAL			
REV	ECO	DESCRIPTION	PROJECT ENGINEER	APPROVAL DATE	DOCUMENT CONTROL	APPROVAL DATE
A	E11523	INITIAL RELEASE OF SCHEMATIC				



NOTES: UNLESS OTHERWISE SPECIFIED;

1. ALL RESISTOR VALUES ARE IN OHMS, 1/10W, 5%
2. ALL CAPACITORS ARE 50VDC OR HIGHER.
3. THE ORIGINATION SOURCE OF A VOLTAGE IS REPRESENTED BY (VCC), AND ALL REFERENCES TO THAT VOLTAGE ARE REPRESENTED BY (VCC).

APPEND THE FOLLOWING DOCUMENTS WHEN CHANGING THIS DOCUMENT:	

DRAWING CONTENT:	
DRAWN BY: (INITIAL RELEASE) E. PEAK	14MAY01
REVISED BY:	
APPROVALS: INITIAL RELEASE	
PROJECT ENGINEER:	
ENGINEERING MANAGER:	
SIGNATURES	DATE

TITLE

SCHEMATIC DIAGRAM
RABBIT PROG. CABLE

ZWORLD

2900 SPAFFORD ST.
DAVIS, CA 95616
530 - 757 - 4616

SIZE	DWG NO.		
B	090−0128		
SCALE NONE	RELEASE DATE	SHEET 1	OF 1