



# **MiniCom (OP6800)**

C-Programmable Operator Interface

## **User's Manual**

019-0106 • 020122-A

# **MiniCom (OP6800) User's Manual**

Part Number 019-0106 • 020122-A • Printed in U.S.A.

©2002 Z-World Inc. • All rights reserved.

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.

All Z-World 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. Z-World 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.

## **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) 757-3792

[www.zworld.com](http://www.zworld.com)

# TABLE OF CONTENTS

<b>Chapter 1. Introduction</b>	<b>1</b>
1.1 Description .....	1
1.2 Features .....	1
1.3 Development and Evaluation Tools .....	2
1.3.1 Tool Kit .....	2
1.3.2 Software .....	3
<b>Chapter 2. Getting Started</b>	<b>5</b>
2.1 Connections .....	5
2.2 Demonstration Program on Power-Up .....	8
2.3 Programming Cable Connections .....	9
2.4 Installing Dynamic C Premier .....	10
2.5 Starting Dynamic C .....	10
2.6 PONG.C .....	11
2.7 Where Do I Go From Here? .....	11
<b>Chapter 3. Subsystems</b>	<b>13</b>
3.1 Pinouts .....	14
3.2 Digital I/O .....	15
3.2.1 Digital Inputs .....	15
3.2.2 Digital Outputs .....	16
3.3 Serial Communication .....	17
3.3.1 RS-232 .....	17
3.3.2 RS-485 .....	17
3.3.3 Programming Port .....	19
3.3.4 Ethernet Port (OP6800 models only) .....	20
3.4 Memory .....	21
3.4.1 SRAM .....	21
3.4.2 Flash Memory .....	21
3.5 Keypad Labeling .....	22
<b>Chapter 4. Software</b>	<b>25</b>
4.1 Programming Cable .....	26
4.1.1 Switching Between Program Mode and Run Mode .....	26
4.1.2 Detailed Instructions: Changing from Program Mode to Run Mode .....	26
4.1.3 Detailed Instructions: Changing from Run Mode to Program Mode .....	26
4.2 OP6800 Libraries .....	27
4.2.1 TCP/IP Libraries .....	28
4.3 Sample Programs .....	29
4.3.1 Board ID .....	29
4.3.2 Demonstration Board .....	29
4.3.3 Digital I/O .....	30
4.3.4 Serial Communication .....	30
4.3.5 LCD/Keypad Module Sample Programs .....	30
4.3.6 TCP/IP Sample Programs .....	31
4.4 Font and Bitmap Converter .....	32

<b>Chapter 5. Using the TCP/IP Features</b>	<b>33</b>
5.1 TCP/IP Connections .....	33
5.2 TCP/IP Sample Programs.....	35
5.2.1 How to Set IP Addresses in the Sample Programs.....	35
5.2.2 How to Set Up your Computer's IP Address for a Direct Connection .....	35
5.2.3 Run the PINGME.C Demo.....	36
5.2.4 Running More Demo Programs With a Direct Connection .....	36
5.2.5 LCD/Keypad Sample Programs Showing TCP/IP Features .....	37
5.3 Where Do I Go From Here? .....	38
<b>Chapter 6. Installation and Mounting Guidelines</b>	<b>39</b>
6.1 Installation Guidelines.....	39
6.2 Mounting Instructions .....	40
6.2.1 Bezel-Mount Installation .....	40
<b>Appendix A. Specifications</b>	<b>43</b>
A.1 Electrical and Mechanical Specifications.....	44
A.2 Conformal Coating .....	47
A.3 Jumper Configurations .....	48
A.4 Use of Rabbit 2000 Parallel Ports .....	49
A.5 I/O Address Assignments.....	51
<b>Appendix B. Power Supply</b>	<b>53</b>
B.1 Power Supplies .....	53
B.2 Batteries and External Battery Connections.....	54
B.2.1 Battery-Backup Circuit.....	54
B.2.2 Power to VRAM Switch.....	55
B.2.3 Reset Generator.....	55
B.3 Chip Select Circuit.....	56
<b>Appendix C. Demonstration Board</b>	<b>57</b>
C.1 Mechanical Dimensions and Layout .....	58
C.2 Power Supply.....	59
C.3 Using the Demonstration Board .....	61
<b>Appendix D. OP6800 Function APIs</b>	<b>65</b>
D.1 Board Initialization.....	66
D.2 Digital I/O.....	67
D.3 Serial Communication .....	68
D.4 LEDs.....	69
D.5 LCD Display.....	70
D.6 Keypad.....	86
<b>Appendix E. Programming Cable</b>	<b>89</b>
<b>Index</b>	<b>91</b>
<b>Schematics</b>	<b>95</b>

# 1. INTRODUCTION

The OP6800 intelligent terminal interface is a small, high-performance, C-programmable terminal interface that offers built-in I/O and Ethernet connectivity. A Rabbit 2000 microprocessor operating at 22.1 MHz provides fast data processing.

## 1.1 Description

The OP6800 intelligent terminal interface incorporates the powerful Rabbit 2000 microprocessor, flash memory, static RAM, digital I/O ports, RS-232/RS-485 serial ports, and a 10Base-T Ethernet port.

## 1.2 Features

- 122 × 32 graphic display.
- 7-key keypad.
- 7 LEDs.
- 24 digital I/O: 13 filtered digital inputs, and 11 sinking high-current outputs (7 outputs with LED indicators, and 4 high-current digital outputs with transient protection to drive inductive loads).
- Rabbit 2000™ microprocessor operating at 22.1 MHz.
- 128K static RAM and 256K flash memory standard, may be increased to 512K SRAM and 512K flash memory.
- One RJ-45 Ethernet port compliant with IEEE 802.3 standard for 10Base-T Ethernet protocol (OP6800 only).
- Four serial ports (2 RS-232 or 1 RS-232 with RTS/CTS, 1 RS-485, and 1 CMOS-compatible programming port).
- Battery-backable real-time clock, connection point for external battery included.
- Watchdog.
- Reset generator.
- Meets NEMA 4 watertightness specifications when front-panel mounted.
- Remote program downloading and debugging capability via RabbitLink.

Two OP6800 models are available. Their standard features are summarized in Table 1.

**Table 1. OP6800 Models**

Feature	OP6800	OP6810
Microprocessor	Rabbit 2000 running at 22.1 MHz	
Static RAM	128K	
Flash Memory	256K	
RJ-45 Ethernet Connector and Filter Capacitors	Yes	No
RabbitCore Module Used	RCM2200	RCM2300

Appendix A provides detailed specifications.

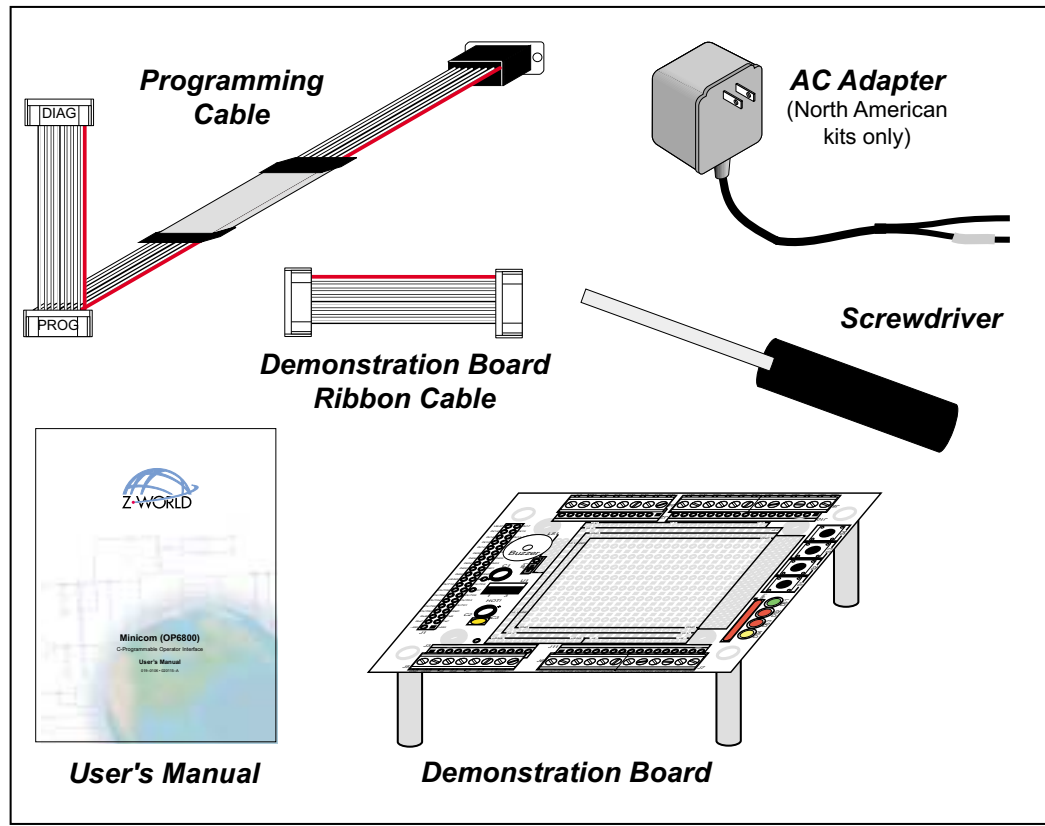
Visit [Z-World's Web site](#) for up-to-date information about additional add-ons and features as they become available. The Web site also has the latest revision of this user's manual.

## 1.3 Development and Evaluation Tools

### 1.3.1 Tool Kit

A Tool Kit contains the hardware essentials you will need to use your OP6800. The items in the Tool Kit and their use are as follows.

- **OP6800 User's Manual** with schematics (this document).
- Programming cable, used to connect your PC serial port to the OP6800.
- 12 V AC adapter, used to power the OP6800. An AC adapter is supplied with tool kits sold in the North American market. If you are using your own power supply, it must provide 9 to 36 V DC.
- Demonstration Board with prototyping area, pushbutton switches, and LEDs. The Demonstration Board can be hooked up to the OP6800 to demonstrate the I/O, and the prototyping area can be used for custom circuits.
- Ribbon cable to connect Demonstration Board to OP6800.
- Screwdriver.
- **Rabbit 2000 Processor Easy Reference** poster.
- Registration card.



**Figure 1. OP6800 Tool Kit**

### **1.3.2 Software**

The OP6800 is programmed using version 7.06 or later of Z-World's Dynamic C Premier, an integrated development environment that includes an editor, a C compiler, and a debugger. Library functions provide an easy-to-use interface for the OP6800. Software drivers for the display and keypad, TCP/IP, I/O, and serial communication are included with Dynamic C Premier.



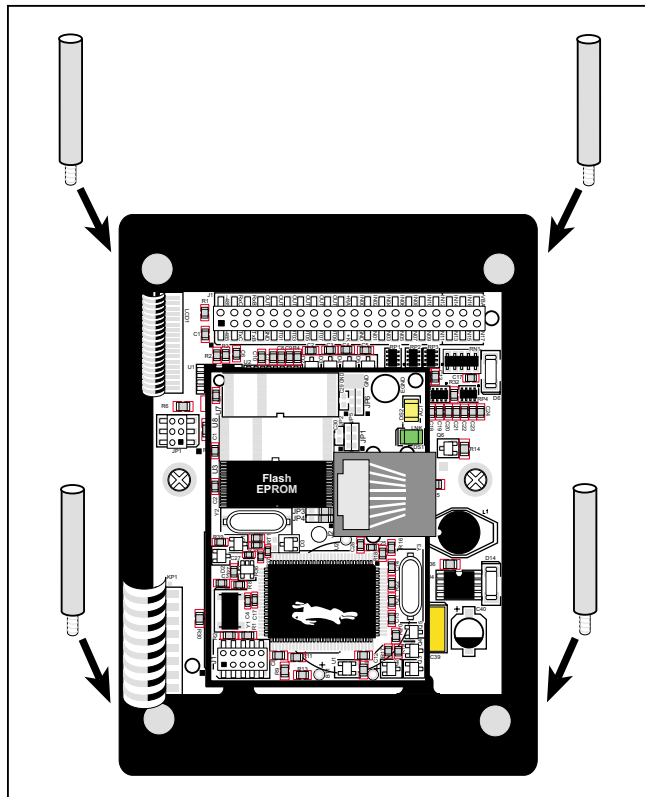


## 2. GETTING STARTED

Chapter 2 explains how to connect the programming cable and power supply to the OP6800. 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.

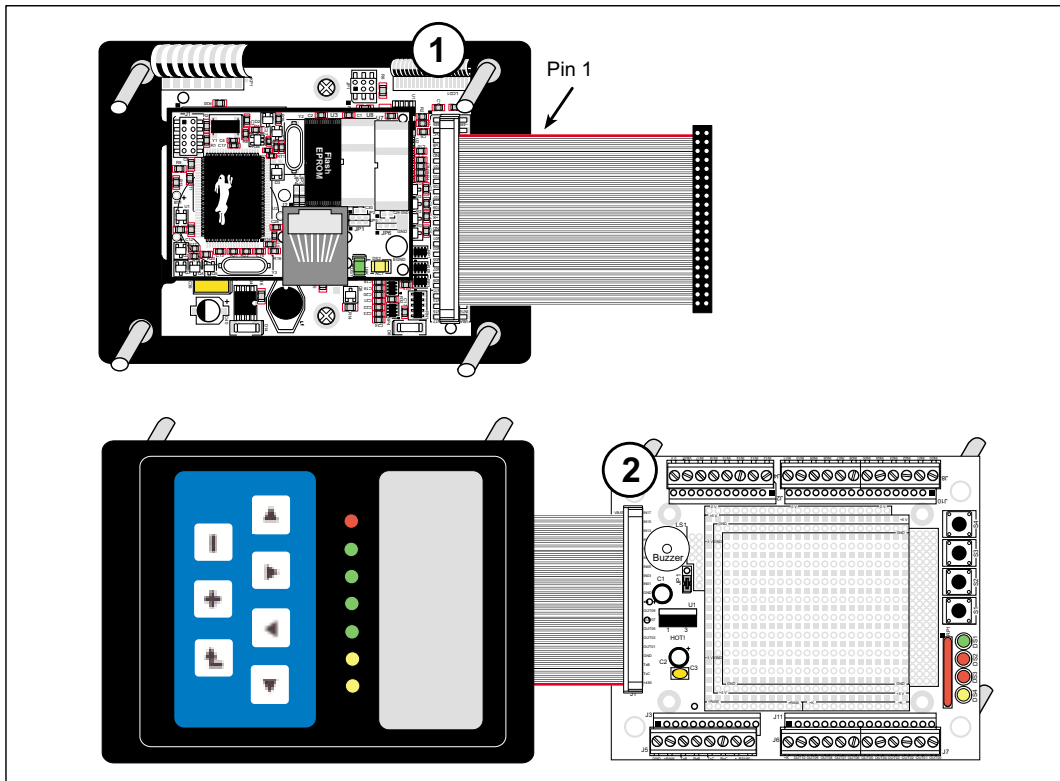
### 2.1 Connections

1. Screw in the four standoffs included with the Tool Kit into the four mounting threads on the OP6800 as shown in Figure 2.



**Figure 2. Screw In Standoffs Into OP6800 Mounting Threads**

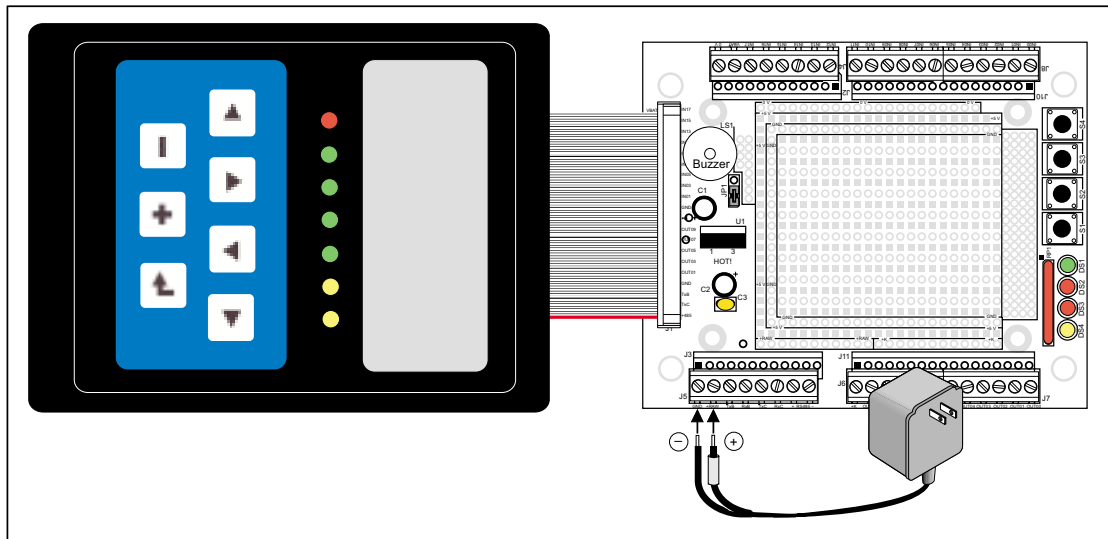
2. Connect the OP6800 to the Demonstration Board from the Tool Kit using the ribbon cable connector as shown in Figure 3. First, connect the ribbon cable to header J1 on the OP6800 (Step 1), then turn the OP6800 over and connect the other end of the ribbon cable to header J1 on the Demonstration Board (Step 2). By connecting the boards this way, you have the option of placing the Demonstration Board behind your OP6800 in your final installation as explained in Appendix C.



**Figure 3. Connect the OP6800 to the Demonstration Board**

3. Connect the power supply.

Connect the bare ends of the power supply to the **+RAW** and **GND** positions on screw terminal header J5 of the Demonstration Board as shown in Figure 4.



**Figure 4. Power Supply Connections**

**NOTE:** The OP6800 itself has reverse polarity protection, but the Demonstration Board does not. Be careful to connect the positive and negative leads as shown to avoid damaging the Demonstration Board.

**NOTE:** If you are using your own power supply, Z-World recommends using a 9 V to 25 V DC power supply. The linear regulator on the Demonstration Board can handle up to 35 V, but can get extremely hot.

4. Apply power.

Plug in the AC adapter.

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

## 2.2 Demonstration Program on Power-Up

A repeating sequence of graphics and messages in various languages will be displayed on the LCD and the LEDs will flash on and off in sequence when power is first applied to the OP6800. Try pressing the buttons on the keypad. The LED immediately above that button will light up, and if you pressed one of the keys in the top row of the keypad, the corresponding LED on the Demonstration Board will light up. Similarly, if you press one of the switches on the Demonstration Board, the corresponding LED on the Demonstration Board and on the OP6800 will light up.

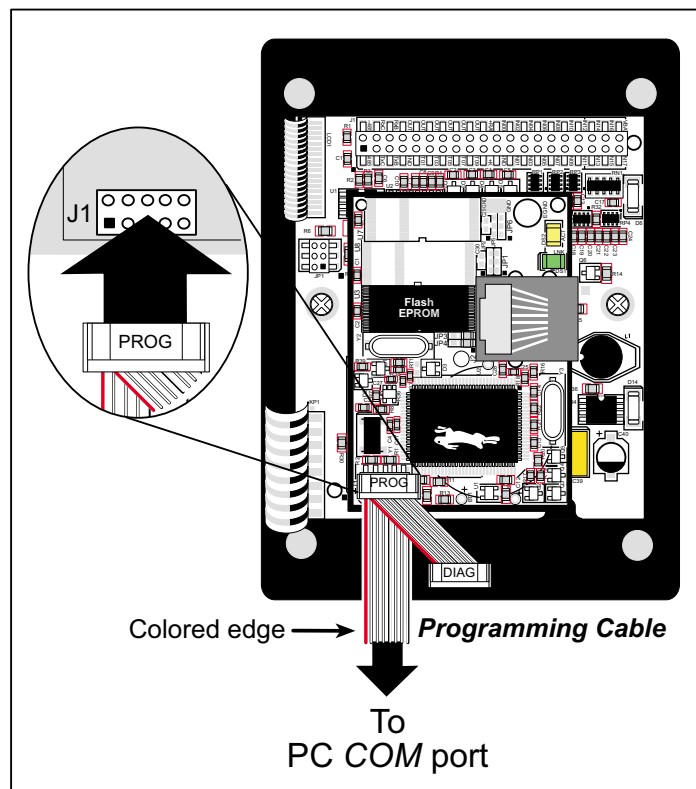
Note that the programming cable does ***not*** have to be connected for this demonstration.

This demonstration will be replaced by a new program when the programming cable is attached and the new program is compiled and run. The demonstration is available for future reference in the Dynamic C Premier **SAMPLES\LCD\_KEYPAD\122x32\_1x7** directory as **FUN.C**.

## 2.3 Programming Cable Connections

1. Connect the programming cable to download programs from your PC and to program and debug the OP6800.

Connect the 10-pin **PROG** connector of the programming cable to header J1 on the OP6800 RabbitCore module. Ensure that the colored edge lines up with pin 1 as shown. (Do not use the **DIAG** connector, which is used for monitoring only, as explained in Appendix E, “Programming Cable.”) 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 will need to have this parameter configured. Note that COM1 on the PC is the default COM port used by Dynamic C Premier.



**Figure 5. Programming Cable Connections**

2. Apply power.

Reset the OP6800 by unplugging the AC adapter, then plugging it back in. The OP6800 is now ready to be used.

## 2.4 Installing Dynamic C Premier

If you have not yet installed Dynamic C version 7.06P2 (or a later version), do so now by inserting the Dynamic C Premier CD in your PC's CD-ROM drive. The CD will auto-install unless you have disabled auto-install on your PC.

If the CD does not auto-install, click **Start > Run** from the Windows **Start** button and browse for the Dynamic C Premier **setup.exe** file on your CD drive. Click **OK** to begin the installation once you have selected the **setup.exe** file.

The *Dynamic C Premier User's Manual* provides detailed instructions for the installation of Dynamic C and any future upgrades.

**NOTE:** If you have an earlier version of Dynamic C already installed, the default installation of the later version will be in a different folder, and a separate icon will appear on your desktop.

## 2.5 Starting Dynamic C

Once the OP6800 is connected to your PC and to a power source, start Dynamic C by double-clicking on the Dynamic C icon or by double-clicking on the **.exe** file associated with **DcRab** in the Dynamic C directory.

Dynamic C assumes, by default, that you are using serial port COM1 on your PC. If you *are* using COM1, then Dynamic C should detect the OP6800 and go through a sequence of steps to cold-boot the OP6800 and to compile the BIOS. If the error message "Rabbit Processor Not Detected" appears, you have probably connected to a different PC serial port such as COM2, COM3, or COM4. You can change the serial port used by Dynamic C with the **OPTIONS** menu, then try to get Dynamic C to recognize the OP6800 by selecting **Reset Target/Compile BIOS** on the **Compile** menu. Try the different COM ports in the **OPTIONS** menu until you find the one you are connected to. If you still can't get Dynamic C to recognize the target on any port, then the hookup may be wrong or the COM port might not be working on your PC.

If you receive the "BIOS successfully compiled ..." message after pressing **<Ctrl-Y>** or starting Dynamic C, and this message is followed by a communications 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.

## 2.6 PONG.C

You are now ready to test your programming connections 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 shows that the CPU is working. The sample program described in Section 5.2.3, “Run the PINGME.C Demo,” tests the TCP/IP portion of the board (if you have the OP6800 model—the OP6810 does not have an Ethernet capability).

## 2.7 Where Do I Go From Here?

**NOTE:** If you purchased your OP6800 through a distributor or Z-World partner, contact the distributor or Z-World partner first for technical support.

If there are any problems at this point:

- Check the Z-World Technical Bulletin Board at [www.zworld.com/support/bb/](http://www.zworld.com/support/bb/).
- Use the Technical Support e-mail form at [www.zworld.com/support/support\\_submit.html](http://www.zworld.com/support/support_submit.html).
- Call Z-World Technical Support at (530)757-3737.

If the sample program ran fine, you are now ready to go on to explore other OP6800 features and develop your own applications.

The following sample programs illustrate the features and operation of the OP6800.

<b>OP6800</b> <b>(SAMPLES\LCD_KEYPAD\122x32_1x7)</b>	<b>Demonstration Board</b> <b>(SAMPLES\OP6800\DEMO_BD)</b>
KEYBASIC.C KEYMENU.C SCROLLING.C TEXT.C	KEYPAD.C SWITCHES.C

These sample programs can be used as templates for applications you may wish to develop.

Chapter 3, “Subsystems,” provides a description of the OP6800’s features, Chapter 4, “Software,” describes the Dynamic C software libraries and describes the sample programs, and Chapter 5, “Using the TCP/IP Features,” explains the TCP/IP features and describes some sample programs.





## 3. SUBSYSTEMS

Chapter 3 describes the principal subsystems for the OP6800.

- Digital I/O
- Serial Communication
- Memory

Figure 6 shows these Rabbit-based subsystems designed into the OP6800.

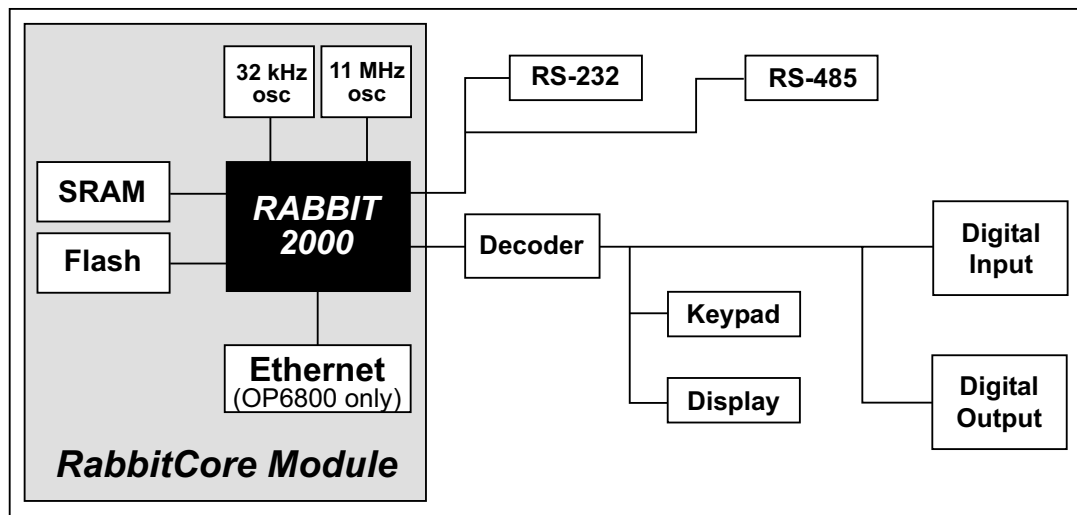
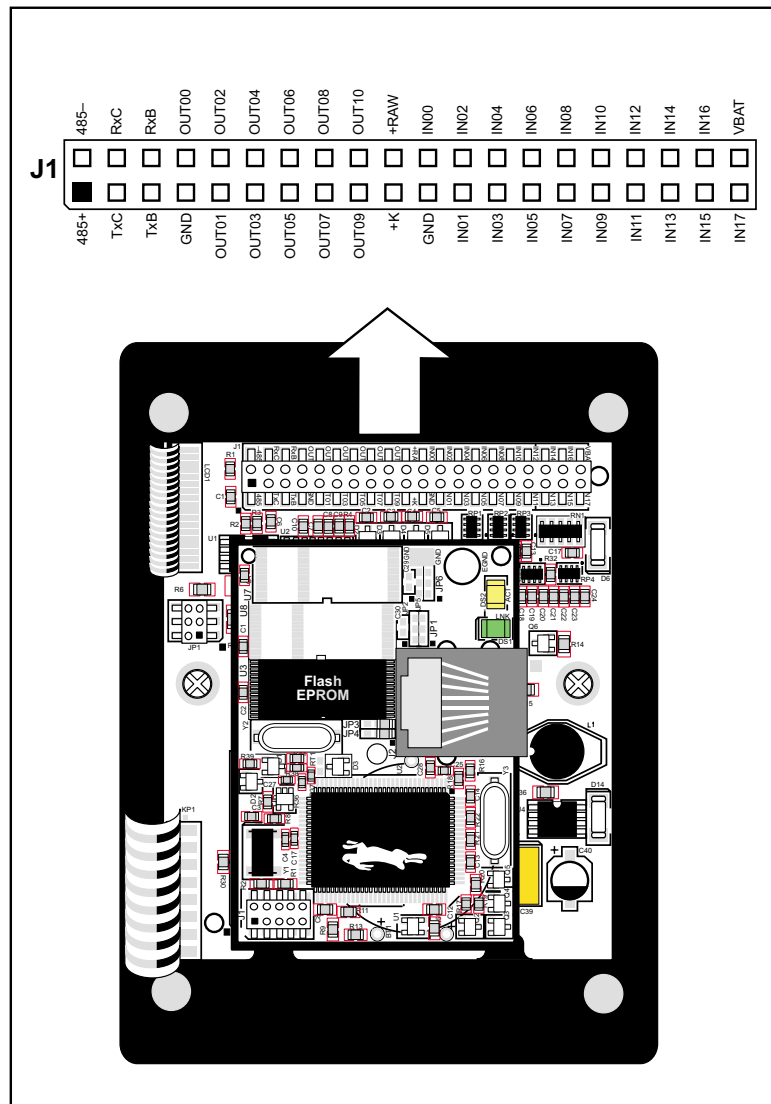


Figure 6. OP6800 Subsystems

### 3.1 Pinouts

Figure 7 shows the OP6800 pinouts.



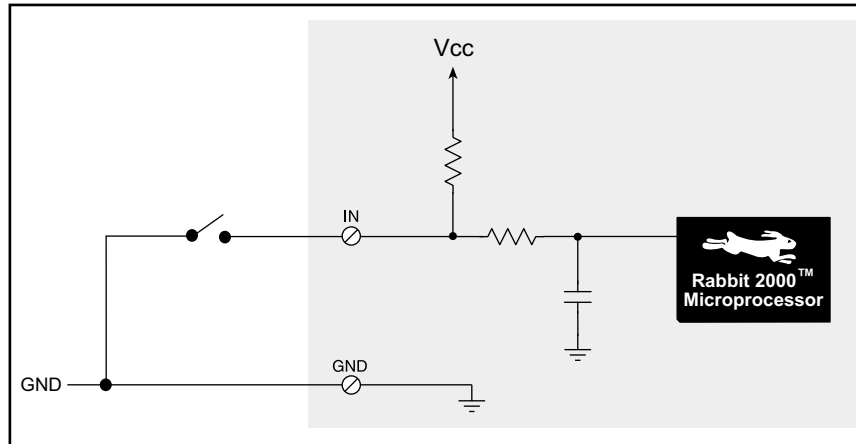
**Figure 7. OP6800 Pinouts**

Header J1 is a standard  $2 \times 20$  header with a nominal 0.1" pitch. The OP6800 also has an RJ-45 Ethernet jack on the RabbitCore module.

## 3.2 Digital I/O

### 3.2.1 Digital Inputs

The OP6800 has eight digital inputs, IN00–IN07, each with a current-limiting resistor of 27 k $\Omega$ , and protected over a range of –36 V to +36 V. The inputs are all pulled up to +5 V as shown in Figure 8.








**Figure 8. OP6800 Digital Inputs**

The OP6800 also has five digital inputs, IN08–IN12, each with a current-limiting resistor of 12 k $\Omega$ , protected over a range of –25 V to +25 V, and pulled up to +5 V.

The actual switching threshold for IN00–IN12 is approximately 2.40 V. Anything below this value is a logic 0, and anything above is a logic 1.

IN13–IN17 are connected in parallel with five of the keypad buttons. These inputs are normally pulled up, but pulling one of these inputs down is the equivalent of pressing the corresponding keypad key remotely.

**Table 2. Remote Keypad Operation**

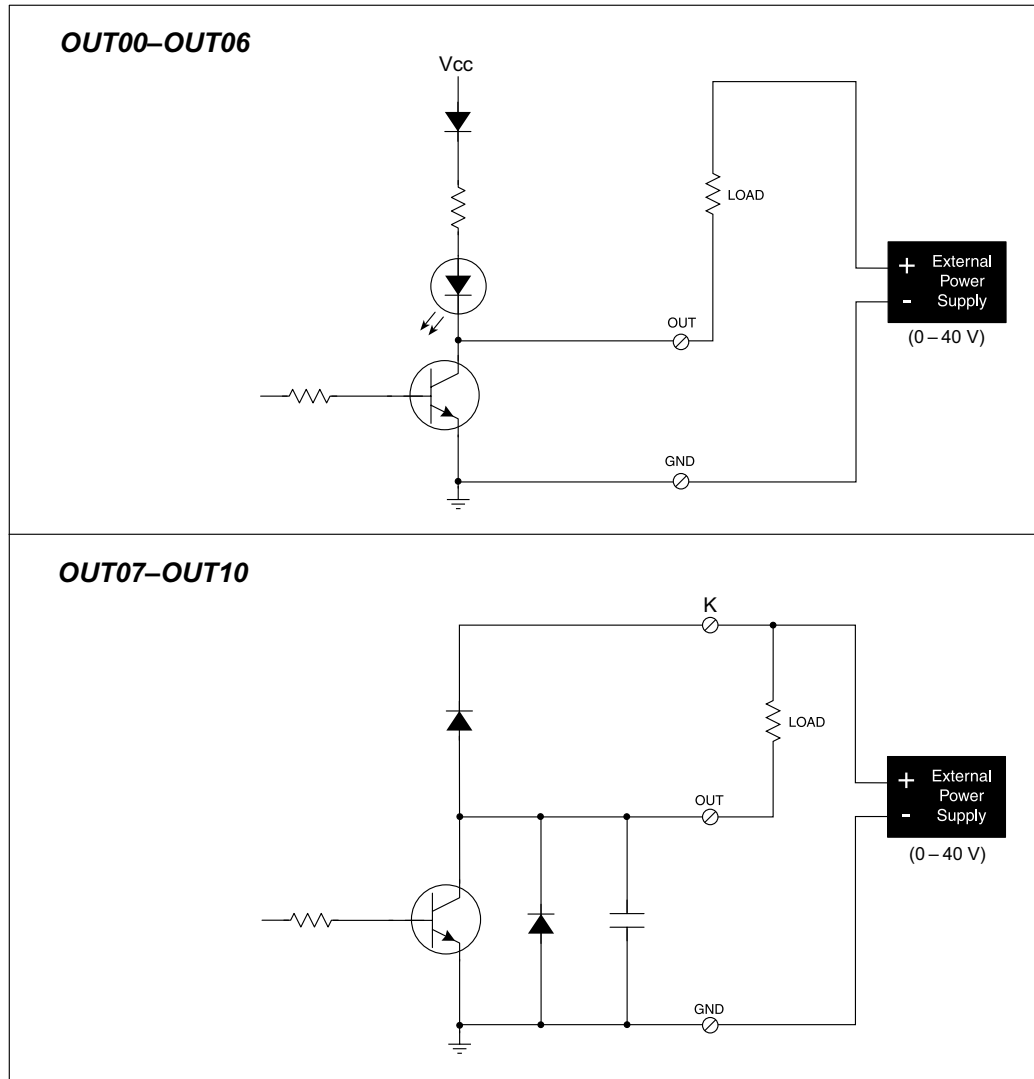
Keypad Key	Remote Keypad Signal Inputs
0 (  )	IN13
1 (  )	IN14
2 (  )	IN15
3 (  )	IN16
6 (  )	IN17

**NOTE:** Remote keypad signal inputs IN13–IN17 are *not* protected, and can only handle a voltage range from 0 to +5 V. These inputs were designed solely to facilitate a remote keypad, and should not be used for other purposes.

### 3.2.2 Digital Outputs

The OP6800 has 11 digital outputs, OUT00–OUT10, which can each sink up to 200 mA. Figure 9 shows a wiring diagram for using the digital outputs.

OUT00–OUT06 can switch up to 40 V and the corresponding LEDs when the outputs are on. OUT07–OUT10 offer protection for inductive loads when K is connected to an external power supply; OUT07–OUT10 are not connected to the LEDs.



**Figure 9. OP6800 Digital Outputs**

It is possible to use an external open-collector driver to control the LEDs associated with OUT00–OUT06. Connect the external driver to the output corresponding to the LED you wish to control, but keep the internal driver turned off. The external driver will then control the LED.

### 3.3 Serial Communication

The OP6800 has two RS-232 serial ports, which can be configured as one RS-232 serial channel (with RTS/CTS) or as two RS-232 (3-wire) channels using the **serMode** software function call. Table 3 summarizes the options.

**Table 3. Serial Communication Configurations**

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

The OP6800 also has one RS-485 serial channel and one CMOS serial channel that serves as the programming port.

All four serial ports operate in an asynchronous mode. An asynchronous port can handle 7 or 8 data bits. A 9th bit address scheme, where an additional bit is sent to mark the first byte of a message, is also supported. Serial Port A, the programming port, can be operated alternately in the clocked serial mode. In this mode, a clock line synchronously clocks the data in or out. Either of the two communicating devices can supply the clock. The OP6800 boards typically use all four ports in the asynchronous serial mode. Serial Ports B and C are used for RS-232 communication, and Serial Port D is used for RS-485 communication. The OP6800 uses an 11.0592 MHz crystal, which is doubled to 22.1184 MHz. At this frequency, the OP6800 supports standard asynchronous baud rates up to a maximum of 230,400 bps.

#### 3.3.1 RS-232

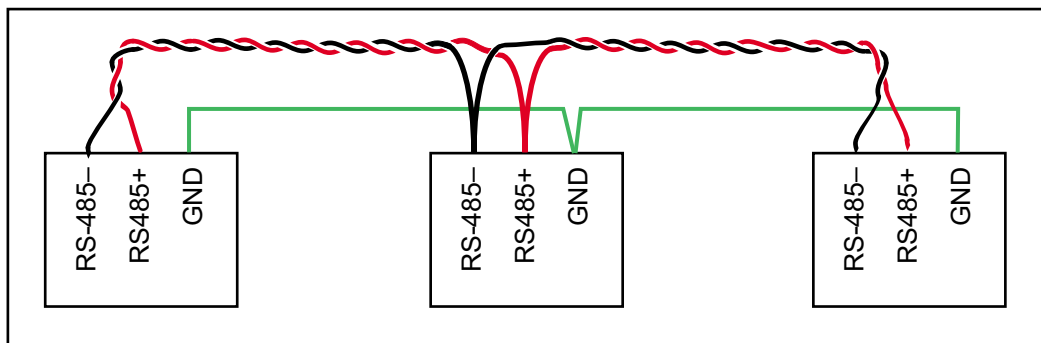
The OP6800 RS-232 serial communication is supported by an RS-232 transceiver. This transceiver provides the voltage output, slew rate, and input voltage immunity required to meet the RS-232 serial communication protocol. Basically, the chip translates the Rabbit 2000's CMOS/TTL signals to RS-232 signal levels. Note that the polarity is reversed in an RS-232 circuit so that a +5 V output becomes approximately -10 V and 0 V is output as +10 V. The RS-232 transceiver also provides the proper line loading for reliable communication.

RS-232 can be used effectively at the OP6800's maximum baud rate for distances of up to 15 m.

#### 3.3.2 RS-485

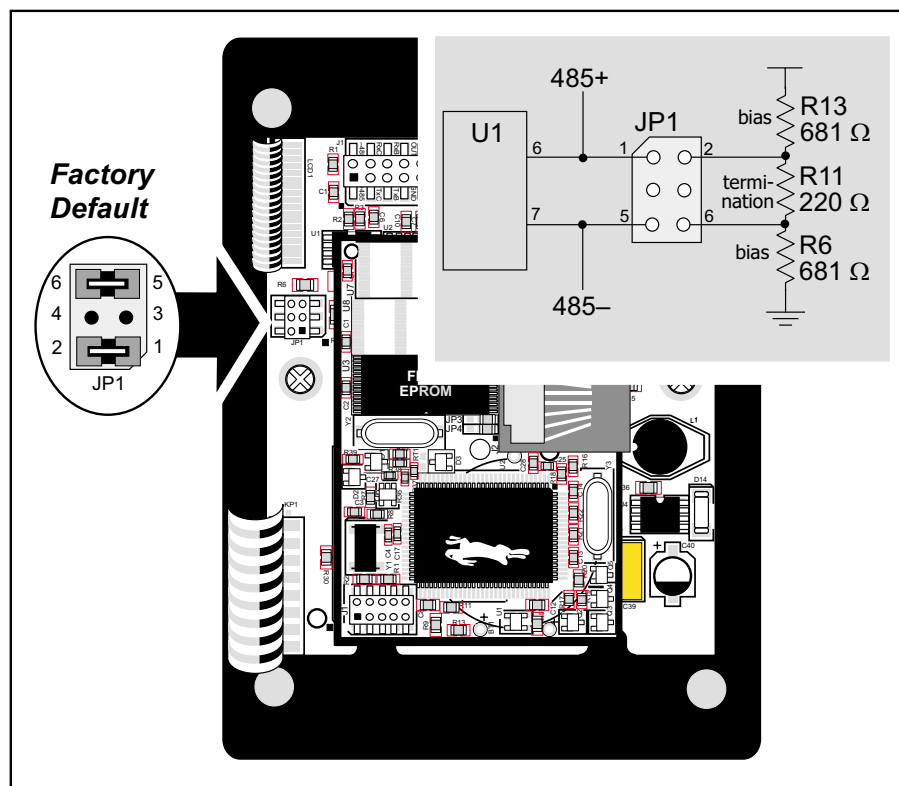
The OP6800 has one RS-485 serial channel, which is connected to the Rabbit 2000 Serial Port D through an RS-485 transceiver. The half-duplex communication uses the Rabbit 2000's PB6 pin to control the transmit enable on the communication line.

The OP6800 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) as shown in Figure 10. Note that a common ground is recommended.



**Figure 10. OP6800 Multidrop Network**

The OP6800 comes with a  $220\ \Omega$  termination resistor and two  $681\ \Omega$  bias resistors installed and enabled with jumpers across pins 1–2 and 5–6 on header JP1, as shown in Figure 11.



**Figure 11. RS-485 Termination and Bias Resistors**

For best performance, 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 any intervening OP6800 units in the network by removing both jumpers from header JP1.

**TIP:** Save the jumpers for possible future use by “parking” them across pins 1–3 and 4–6 of header JP1. Pins 3 and 4 are not otherwise connected to the OP6800.

### 3.3.3 Programming Port

The RabbitCore module on the OP6800 has a 10-pin programming header. The programming port uses the Rabbit 2000's Serial Port A for communication, and is used for the following operations.

- Programming/debugging
- Cloning
- Remote program download/debug over an Ethernet connection via the RabbitLink EG2100

The programming port is used to start the OP6800 in a mode where the OP6800 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.

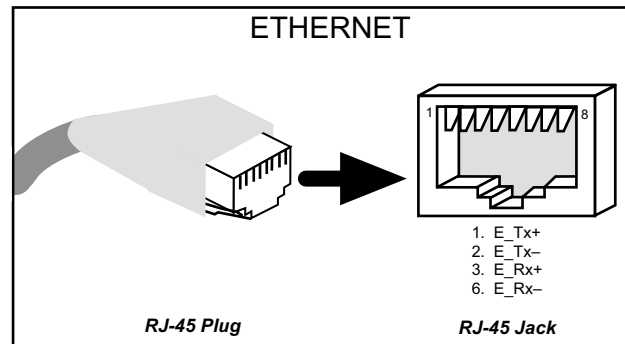
The Rabbit 2000 startup-mode pins (SMODE0, SMODE1) are presented to the programming port so that an externally connected device can force the OP6800 to start up in an external bootstrap mode. The OP6800 can be reset from the programming port via the **/EXT\_RSTIN** line.

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.

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

### 3.3.4 Ethernet Port (OP6800 models only)

Figure 12 shows the pinout for the Ethernet port (J2 on the OP6800 module). Note that there are two standards for numbering the pins on this connector—the convention used here, and numbering in reverse to that shown. 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 12) are always absolute, and the RJ-45 connector will work properly with off-the-shelf Ethernet cables.

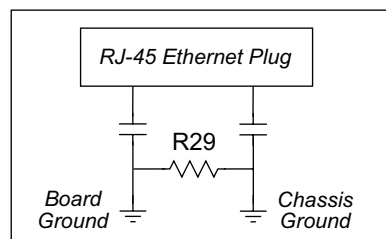


**Figure 12. RJ-45 Ethernet Port Pinout**

RJ-45 pinouts are sometimes numbered opposite to the way shown in Figure 12.

Two LEDs are placed next to the RJ-45 Ethernet jack, one to indicate an Ethernet link (**LNK**) and one to indicate Ethernet activity (**ACT**).

The transformer/connector assembly ground is connected to the BL2100 module printed circuit board digital ground via a 0  $\Omega$  resistor “jumper,” R29, as shown in Figure 13.



**Figure 13. Isolation Resistor R29**

The factory default is for the 0  $\Omega$  resistor “jumper” at R29 to be installed. In high-noise environments, remove R29 and ground the transformer/connector assembly directly through the chassis ground. This will be especially helpful to minimize ESD and/or EMI problems.



## 3.4 Memory

### 3.4.1 SRAM

The OP6800 module is designed to accept 128K to 512K of SRAM packaged in an SOIC case. The standard OP6800 modules come with 128K of SRAM.

### 3.4.2 Flash Memory

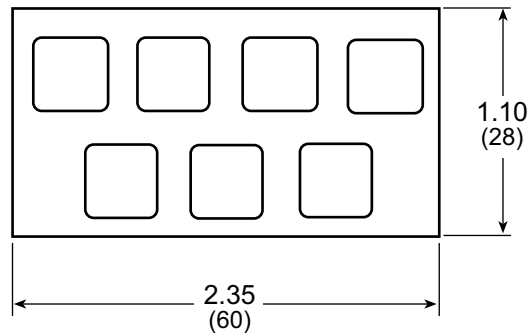
The OP6800 is also designed to accept 128K to 512K of flash memory. The standard OP6800 modules comes with one 256K flash memory.

**NOTE:** Z-World recommends that any customer applications should not be constrained by the sector size of the flash memory since it may be necessary to change the sector size in the future.

A Flash Memory Bank Select jumper configuration option based on 0  $\Omega$  surface-mounted resistors exists at header JP2 on the RabbitCore module. This option, used in conjunction with some configuration macros, allows Dynamic C to compile two different co-resident programs for the upper and lower halves of the 256K flash in such a way that both programs start at logical address 0000. This is useful for applications that require a resident download manager and a separate downloaded program. See Application Note 218, *Implementing a Serial Download Manager for a 256K Flash*, for details.

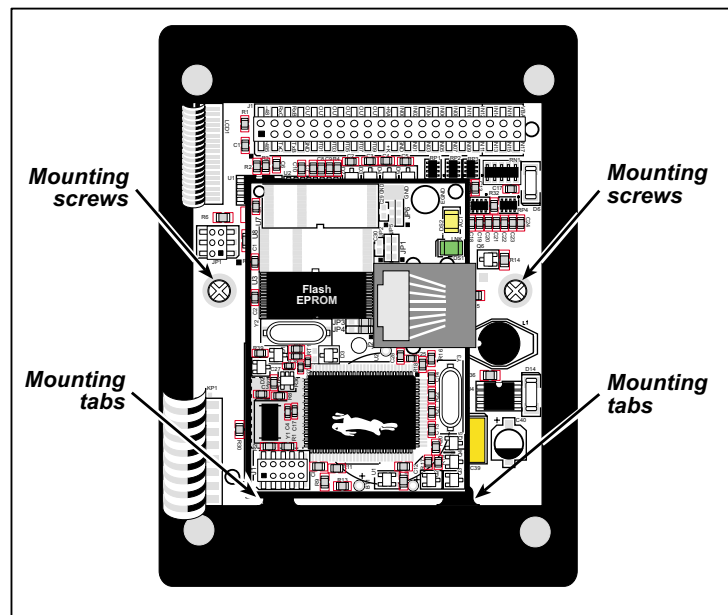
### 3.5 Keypad Labeling

The keypad may be labeled according to your needs. A template is provided in Figure 14 to allow you to design your own keypad label insert.



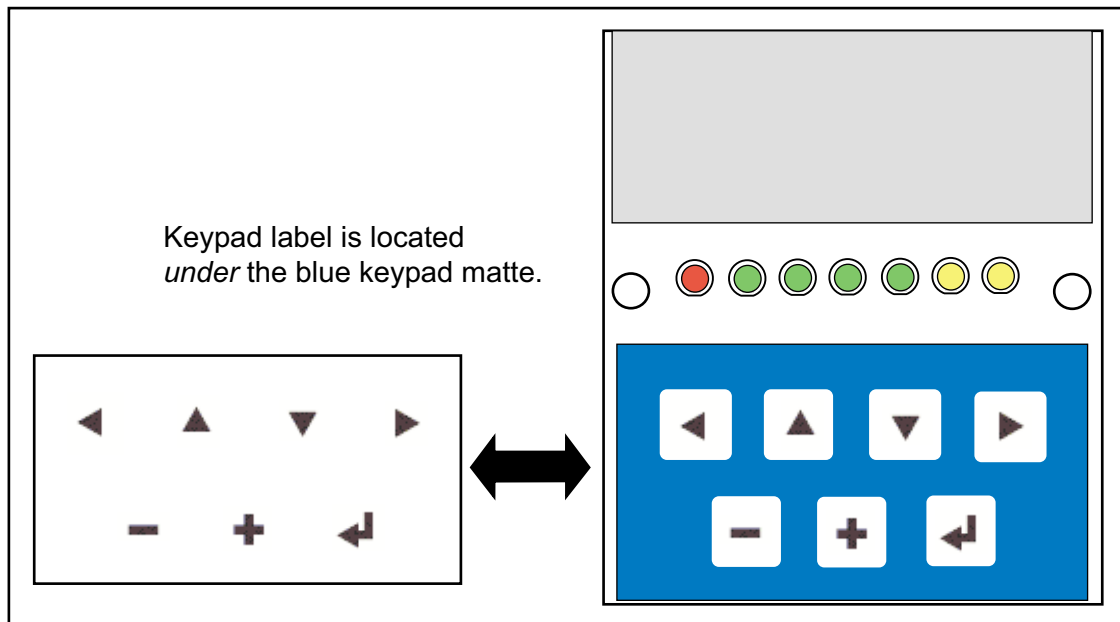
**Figure 14. Keypad Template**

Before you can replace the keypad legend, you will have to remove the LCD/keypad module from the plastic bezel. The LCD/keypad module circuit board is held down with two screws and two tabs as shown in Figure 15.



**Figure 15. Removing LCD/Keypad Module from Plastic Bezel**

To replace the keypad legend, remove the old legend and insert your new legend prepared according to the template in Figure 14. The keypad legend is located under the blue keypad matte, and is accessible from either the left side or the right side as shown in Figure 16. A small screwdriver or a similar small pointed object can be used to nudge the keypad legend in or out.



**Figure 16. Removing and Inserting Keypad Label**

Once you have replaced the keypad label, re-insert the LCD/keypad module circuit board under the mounting tabs in the plastic bezel, as shown in Figure 15. Secure the LCD/keypad module circuit board with the two screws.



## 4. SOFTWARE

Dynamic C Premier is an integrated development system for writing embedded software. It runs on an IBM-compatible PC and is designed for use with Z-World single-board computers and other devices based on the Rabbit microprocessor.

Chapter 4 provides the libraries, function calls, and sample programs related to the OP6800.

You have a choice of doing your software development in the flash memory or in the static RAM included on the OP6800. The advantage of working in RAM is to save wear on the flash memory, which is limited to about 100,000 write cycles.

**NOTE:** An application can be developed in RAM, but cannot run standalone from RAM after the programming cable is disconnected. All standalone applications can only run from flash memory.

The disadvantage of using flash memory for debug is that interrupts must be disabled for approximately 5 ms whenever a break point is set in the program. This can crash fast interrupt routines that are running while you stop at a break point or single-step the program. Flash memory or RAM is selected on the **Options > Compiler** menu.

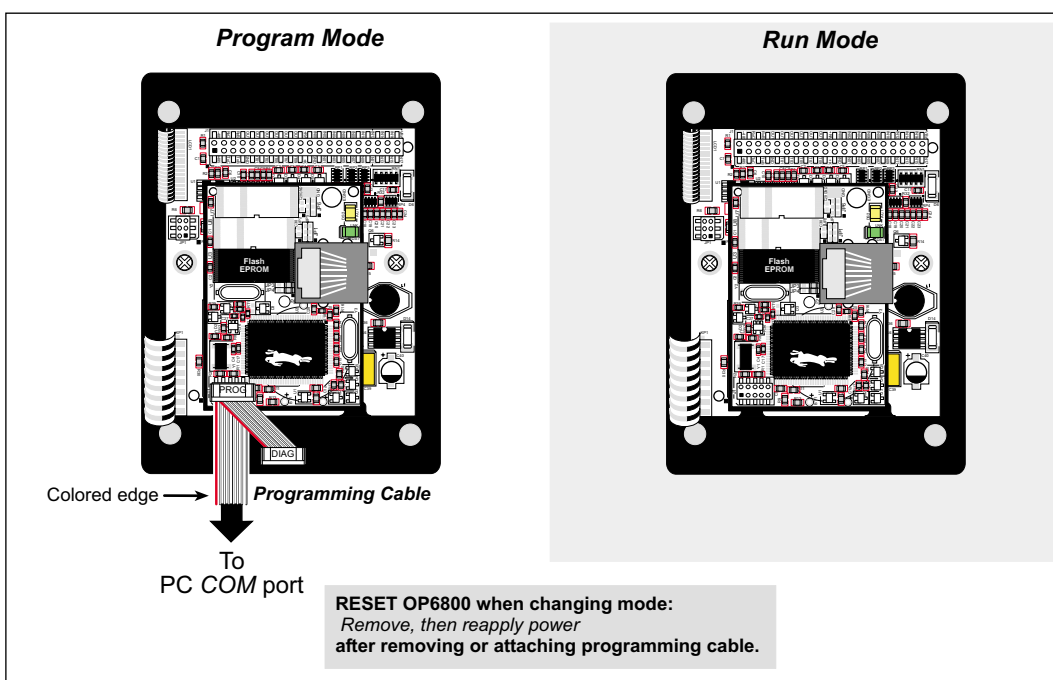
Dynamic C Premier provides a number of debugging features. You can single-step your program, either in C, statement by statement, or in assembly language, instruction by instruction. You can set break points, where the program will stop, on any statement. You can evaluate watch expressions. A watch expression is any C expression that can be evaluated in the context of the program. If the program is at a break point, a watch expression can view any expression using local or external variables.

## 4.1 Programming Cable

The programming cable has a level converter board in the middle of the cable since the OP6800 programming port supports CMOS logic levels, and not the higher voltage RS-232 levels that are used by PC serial ports. When the programming cable is connected, Dynamic C running on the PC can hard-reset the OP6800 and cold-boot it. The cold boot includes compiling and downloading a BIOS program that stays resident while you work. If you crash the target, Dynamic C will automatically reboot and recompile the BIOS if it senses that a target communication error occurred or that the BIOS source code has changed.

### 4.1.1 Switching Between Program Mode and Run Mode

The OP6800 is automatically in Program Mode when the programming cable is attached, and is automatically in Run Mode when no programming cable is attached. See Figure 17.



**Figure 17. OP6800 Program Mode and Run Mode Set-Up**

### 4.1.2 Detailed Instructions: Changing from Program Mode to Run Mode

1. Disconnect the programming cable from header J1 of the OP6800 module.
2. Reset the OP6800 by unplugging the AC adapter, then plugging it back in.

The OP6800 is now ready to operate in the Run Mode.

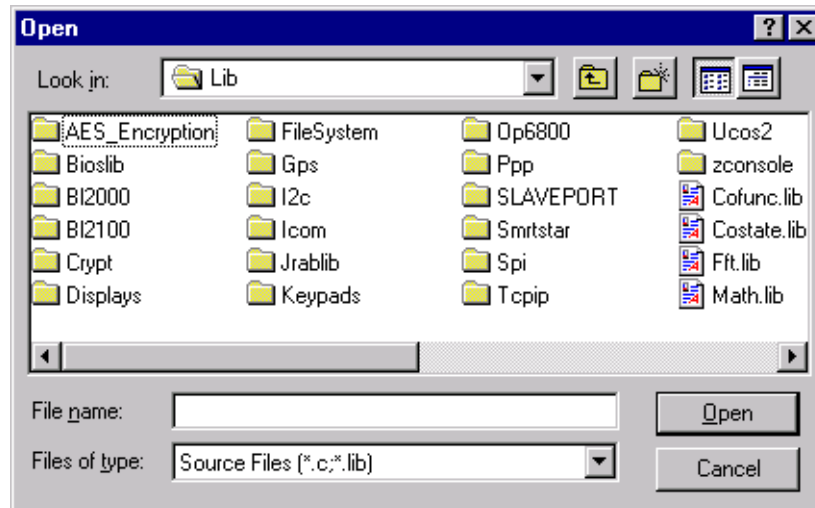
### 4.1.3 Detailed Instructions: Changing from Run Mode to Program Mode

1. Attach the programming cable to header J1 of the OP6800 module.
2. Reset the OP6800 by unplugging the AC adapter, then plugging it back in. Alternatively, you may press **<Ctrl-Y>** on your PC if Dynamic C is running.

The OP6800 is now ready to operate in the Program Mode.

## 4.2 OP6800 Libraries

With Dynamic C running, click **File > Open**, and select **Lib**. The following list of Dynamic C libraries and library directories will be displayed.



Four library directories provide function calls that are specific to the OP6800 or to its features.

- **OP6800**—libraries associated with OP6800 serial communication, I/O, and initialization. The functions in the **OP68xx.LIB** library are described in Appendix D, “OP6800 Function APIs.”
- **DISPLAYS\GRAPHIC**—libraries associated with the LCD display. The functions in these libraries are described in Appendix D, “OP6800 Function APIs.”
- **KEYPADS**—libraries associated with the keypad. The functions in these libraries are described in Appendix D, “OP6800 Function APIs.”
- **TCP/IP**—libraries specific to using TCP/IP functions.

Other generic functions applicable to all devices based on the Rabbit 2000 microprocessor are described in the *Dynamic C Premier User's Manual*.

### 4.2.1 TCP/IP Libraries

The **TCPIP** directory contains libraries with generic TCP/IP functions for the OP6800.

- **ARP.LIB**—address resolution protocol functions.
- **BOOTP.LIB**—bootstrap protocol functions.
- **BSDNAME.LIB**—BSD-style socket routines.
- **DCRTCP.LIB**—TCP/IP functions.
- **DNS.LIB**—handles host name resolution.
- **FTP\_CLIENT.LIB**—FTP client functions.
- **FTP\_SERVER.LIB**—FTP server functions.
- **HTTP.LIB**—HTTP handler.
- **ICMP.LIB**—ICMP handler.
- **IP.LIB**—handles the network layer (just above the link layer and the device driver).
- **MD5.LIB**—implements the MD5 algorithm defined in TCP/IP RFC 1321.
- **NET.LIB**—general networking API. This is the "top-level" library for the networking library suite. It includes the packet-driver interface.
- **PKTDRV.LIB**—packet driver functions.
- **POP3.LIB**—POP3 functions.
- **REALTEK.LIB**—packet driver functions for the RealTek RTL8019AS.
- **SMTP.LIB**—SMTP handler.
- **TCP.LIB**—transmission control protocol.
- **UDP.LIB**—user datagram protocol.
- **VSERIAL.lib**—virtual Telnet functions.
- **ZSERVER.lib**—miscellaneous TCP/IP server data structures and routines.

The functions in these libraries are described in the *Dynamic C TCP/IP User's Manual* included in the manual set with the *Dynamic C Premier User's Manual*.



## 4.3 Sample Programs

Sample programs are provided in the Dynamic C **samples** folder. The sample program **PONG.C** demonstrates the output to the **STDIO** window.

The various directories in the **samples** folder contain specific sample programs that illustrate the use of the corresponding Dynamic C libraries.

The **OP6800** folder provides sample programs specific to the OP6800. Each sample program has comments that describe the purpose and function of the program. Follow the instructions at the beginning of the sample program.

To run a sample 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 OP6800 must be in **Program** mode (see Section 4.1, “Programming Cable,”) and must be connected to a PC using the programming cable as described in Section 2.1, “Connections.”

More complete information on Dynamic C is provided in the *Dynamic C Premier User's Manual*. TCP/IP specific functions are described in the *Dynamic C TCP/IP User's Manual*. Information on using the TCP/IP features and sample programs is provided in Section 5, “Using the TCP/IP Features.”

### 4.3.1 Board ID

The following sample program can be found in the **SAMPLES\OP6800** subdirectory.

- **BOARD\_ID.C**—Detects the type of single-board computer and displays the information in the **STDIO** window. For the OP6800, the **STDIO** window should show **OP6800**.

### 4.3.2 Demonstration Board

The following sample programs are found in the **DEMO\_BD** subdirectory in **SAMPLES\OP6800**.

- **BUZZER.C**—Demonstrates the use of the buzzer on the Demonstration Board. Remember to set the jumper across pins 1–2 of header JP1 on the Demonstration Board (see Figure C-4) to enable the buzzer on. When you finish with **BUZZER.C**, it is recommended that you reconnect the jumper across pins 2–3 of header JP1 on the Demonstration Board to disable the buzzer.
- **KEYPAD.C**—Flashes the LED above a keypad button when the corresponding keypad button is pressed. The corresponding LED on the Demonstration Board will also flash if a keypad button in the top row of the keypad is pressed. A message is also displayed on the LCD.
- **SWITCHES.C**—Flashes the LED on the Demonstration Board and the OP6800 when the corresponding pushbutton switch on the Demonstration Board is pressed. A message is also displayed on the LCD.

### 4.3.3 Digital I/O

The following sample programs are found in the **IO** subdirectory in **SAMPLES\OP6800**.

- **DIGIN.C**—Demonstrates the use of the digital inputs. By pressing a pushbutton switch on the Demonstration Board, you can view an input channel toggle from HIGH to LOW on your PC monitor. The four pushbutton switches correspond to IN00–IN03 on the OP6800. IN04–IN12 can also be toggled by momentarily grounding the inputs.
- **DIGOUT.C**—Demonstrates the use of the sinking high-current outputs. By pressing a pushbutton switch on the Demonstration Board, you can view an output channel toggle the corresponding LEDs on/off. The four pushbutton switches correspond to OUT07–OUT10.

### 4.3.4 Serial Communication

The following sample programs are found in the **RS232** subdirectory in **SAMPLES\OP6800**.

- **PUTS.C**—Transmits and then receives an ASCII string on Serial Ports B and C. It also displays the serial data received from both ports in the **STDIO** window.
- **RELAYCHR.C**—This program echoes characters over Serial Port B to Serial Port C. It must be run with a serial utility such as Hyperterminal.

The following sample programs are found in the **RS485** subdirectory in **SAMPLES\OP6800**.

- **MASTER.C**—This program demonstrates a simple RS-485 transmission of lower case letters to a slave OP6800. The slave will send back converted upper case letters back to the master OP6800 and display them in the **STDIO** window. Use **SLAVE.C** to program the slave OP6800.
- **SLAVE.C**—This program demonstrates a simple RS-485 transmission of lower case letters to a slave OP6800. The slave will send back converted upper case letters back to the master OP6800 and display them in the **STDIO** window. Use **MASTER.C** to program the master OP6800.

### 4.3.5 LCD/Keypad Module Sample Programs

The following sample programs are found in the **122x32\_1x7** subdirectory in **SAMPLES\LCD\_Keypad**.

- **ALPHANUM.C**—Demonstrates how to create messages using the keypad and then displaying them on the LCD display.
- **COFTERMA.C**—Demonstrates cofunctions, the cofunction serial library, and using a serial ANSI terminal such as Hyperterminal from an available COM port connection.
- **DISPPONG.C**—Demonstrates output to LCD display.
- **DKADEMO1.C**—Demonstrates some of the LCD/keypad module font and bitmap manipulation features with horizontal and vertical scrolling, and using the **GRAPHIC.LIB** library.
- **FUN.C**—Demonstrates drawing primitive features (lines, circles, polygons) using the **GRAPHIC.LIB** library

- **KEYBASIC.C**—Demonstrates the following keypad functions in the **STDIO** display window:
  - default ASCII keypad return values.
  - custom ASCII keypad return values.
  - keypad repeat functionality.
- **KEYMENU.C**—Demonstrates how to implement a menu system using a highlight bar on a graphic LCD display. The menu options for this sample are as follows.
  1. Set Date/Time
  2. Display Date/Time
  3. Turn Backlight OFF
  4. Turn Backlight ON
  5. Toggle LEDs
  6. Increment LEDs
  7. Disable LEDs
- **LED.C**—Demonstrates how to toggle the LEDs on the LCD/keypad module.
- **SCROLLING.C**—Demonstrates scrolling features of the **GRAPHIC.LIB** library.
- **TEXT.C**—Demonstrates the text functions in the **GRAPHIC.LIB** library. Here is a list of what is demonstrated.
  1. Font initialization.
  2. Text window initialization.
  3. Text window, end-of-line wraparound, end-of-text window clipping, line feed, and carriage return.
  4. Creating 2 different TEXT windows for display.
  5. Displaying different FONT sizes.

#### 4.3.6 TCP/IP Sample Programs

TCP/IP sample programs are described in Chapter 5.

## 4.4 Font and Bitmap Converter

A *Font and Bitmap Converter* tool is available to convert Windows fonts and monochrome bitmaps to a library file format compatible with Z-World's Dynamic C applications and graphical displays. Non-Roman characters can also be converted by applying the monochrome bitmap converter to their bitmaps.

Start the *Font and Bitmap Converter* tool by double-clicking on the `fbmcnvtr.exe` file in the Dynamic C directory. You then select and convert existing fonts or bitmaps. Complete instructions are available via the **Help** menu that is in the *Font and Bitmap Converter* tool.

Once you are done, the converted file is displayed in the editing window. Editing may be done, but should not be necessary. Save the file as `libraryfilename.lib`, where `libraryfilename` is a file name of your choice.

Add the library file(s) to applications with the statement `#use libraryfilename.lib`, or by cutting and pasting from the library file(s) you created into the application program.

**TIP:** If you used the `#use libraryfilename.lib` statement, remember to enter `libraryfilename.lib` into `lib.dir`, which is located in your Dynamic C directory.

You are now ready to add the font or bitmap to your application using the `glXFontInit` or the `glXPutBitmap` function calls.

## 5. USING THE TCP/IP FEATURES

Chapter 5 discusses using the TCP/IP features on the OP6800 boards. The TCP/IP feature is *not* available on OP6810 versions.

### 5.1 TCP/IP Connections

Before proceeding you will need to have the following items.

- If you don't have an Ethernet connection, you will need to install a 10Base-T Ethernet card (available from your favorite computer supplier) in your 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 Z-World in a TCP/IP tool kit. More information is available at [www.zworld.com](http://www.zworld.com).

1. Connect the AC adapter and the programming cable as shown in Chapter 2, "Getting Started."
2. Ethernet Connections
  - If you do not have access to an Ethernet network, use a crossover Ethernet cable to connect the OP6800 to a PC that at least has a 10Base-T Ethernet card.
  - If you have an Ethernet connection, use a straight-through Ethernet cable to establish an Ethernet connection to the OP6800 from an Ethernet hub. These connections are shown in Figure 18.

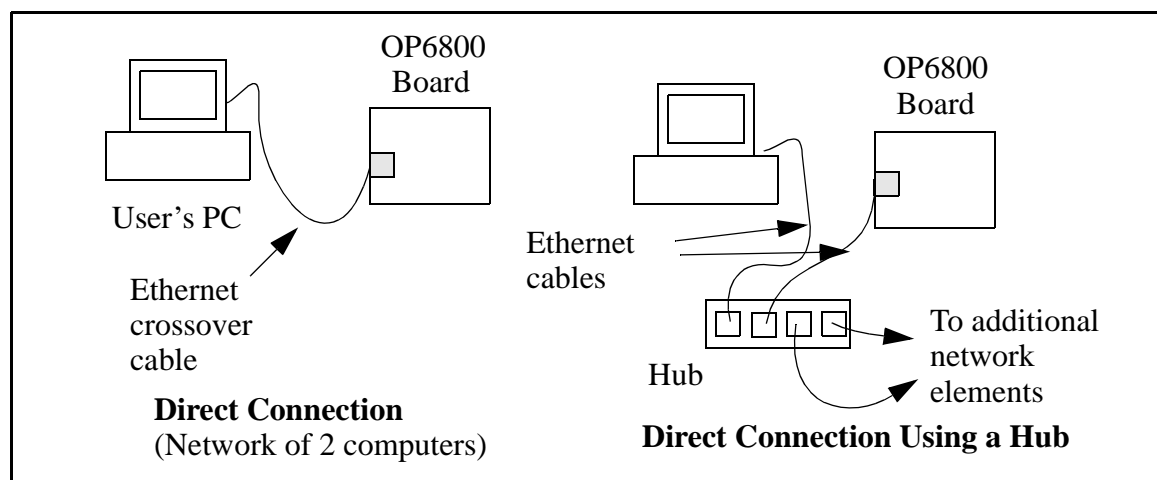


Figure 18. Ethernet Connections

### 3. Apply Power

Plug in the AC adapter. The OP6800 is now ready to be used.

**NOTE:** A hardware RESET is accomplished by unplugging the AC adapter, then plugging it back in, or by momentarily grounding the board reset input at pin 9 on screw terminal header J2.

The green **LNK** light on the OP6800 Rabbitcore module is on when the OP6800 is properly connected either to an Ethernet hub or to an active Ethernet card. The orange **ACT** light flashes each time a packet is received.

## 5.2 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 you connect your PC and the OP6800 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.

### 5.2.1 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"
```

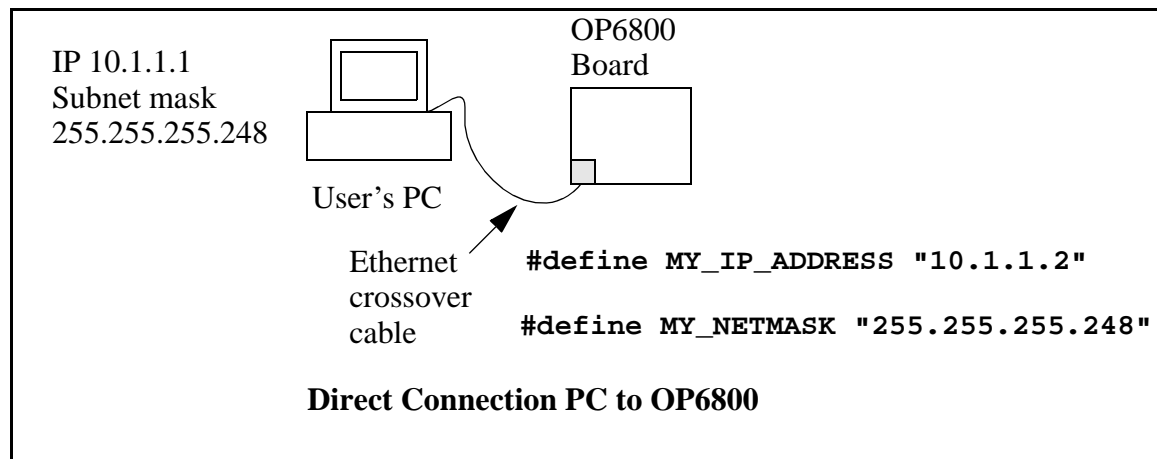
In order to do a direct connection, the following IP addresses can be used for the OP6800:

```
#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.

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

When your computer is connected directly to the OP6800 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.

### 5.2.3 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 OP6800’s RJ-45 Ethernet connector. When the program starts running, the green **LNK** light on the OP6800 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 orange **ACT** light flashes on the OP6800 while the ping is taking place, and indicates the transfer of data. The ping routine will ping the board four times and write a summary message on the screen describing the operation.

### 5.2.4 Running More Demo Programs With a Direct Connection

The program **SSI.C** (**SAMPLES\OP6800\TCPIP\**) demonstrates how to make the OP6800 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 sample program **TELNET.C** (**SAMPLES\OP6800\TCPIP\**) allows you to communicate with the OP6800 using the Telnet protocol. To run this program, edit the IP address, compile the program, 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.

### 5.2.5 LCD/Keypad Sample Programs Showing TCP/IP Features

The following sample programs, found in the **TCPIP** subdirectory in **SAMPLES/LCD\_Keypad/122x32\_1x7**, are targeted at the Ethernet-enabled versions of the OP6800. Remember to configure the IP address, netmask, and gateway as indicated in the sample programs.

- **MBOXDEMO.C**—This program implements a web server that allows Web e-mail messages to be entered that are then shown on the LCD display. The keypad allows you to scroll within messages, flip to other e-mails, mark messages as read, and delete e-mails. When a new e-mail arrives, an LED turns on, and turns off once the message has been marked as read. A log of all e-mail actions is kept, and can be displayed in the Web browser. All current e-mails can also be read with the Web browser.

When using **MBOXDEMO.C**, connect the OP6800 and a PC (or other device with a Web Browser) to an Ethernet. If you connect the PC and the OP6800 directly, be sure to use a crossover Ethernet cable; straight-through Ethernet cables and a hub may be used instead.

- **TCP\_RESPOND.C**—This program and **TCP\_SEND.C** are executed on two separate single-board computers to demonstrate how the two boards communicate with each other. Use **PCSEND.EXE** on the PC console side at the command prompt if you do not have a second board. **PCSEND.EXE** is located with source code in the **SAMPLES/LCD\_Keypad/Windows** directory.

**TCP\_RESPOND.C** waits for a message from another single-board computer. The message received is displayed on the LCD, and you may respond by pressing a key on the keypad. The response is then sent to the remote single-board computer.

- **TCPSEND.C**—This program and **TCP\_RESPOND.C** are executed on two separate single-board computers to demonstrate how the two boards communicate with each other. Use **PCRESPOND.EXE** on the PC console side at the command prompt if you do not have a second board. **PCRESPOND.EXE** is located with source code in the **SAMPLES/LCD\_Keypad/Windows** directory.

When a key on the keypad is pressed, a message associated with that key is sent to a specified destination address and port. The destination then responds to that message. The response is displayed on the LCD.

Note that only the **LEFT** and **UP** scroll keys are set up to cause a message to be sent.

When using **TCPSEND.C** and **TCP\_RESPOND.C**, connect the OP6800 and the other single-board computer to an Ethernet. If you connect them directly, be sure to use a crossover Ethernet cable; straight-through Ethernet cables and a hub may be used instead.

## 5.3 Where Do I Go From Here?

**NOTE:** If you purchased your OP6800 through a distributor or Z-World partner, contact the distributor or Z-World partner first for technical support.

If there are any problems at this point:

- Check the Z-World Technical Bulletin Board at [www.zworld.com/support/bb/](http://www.zworld.com/support/bb/).
- Use the Technical Support e-mail form at [www.zworld.com/support/support\\_submit.html](http://www.zworld.com/support/support_submit.html).
- Call Z-World Technical Support at (530)757-3737.

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

Additional sample programs are described in the *Dynamic C TCP/IP User's Manual*.

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 available on [Z-World's Web site](#).

## 6. INSTALLATION AND MOUNTING GUIDELINES

Chapter 6 describes some considerations for mounting the OP6800 in a panel, and includes detailed mounting instructions.

### 6.1 Installation Guidelines

When possible, following these guidelines when mounting an OP6800.

1. Leave sufficient ventilation space.
2. Do not install the OP6800 directly above machinery that radiates a lot of heat (for example, heaters, transformers, and high-power resistors).
3. Leave at least 8" (20 cm) distance from electric power lines and even more from high-voltage devices.
4. When installing the OP6800 near devices with strong electrical or magnetic fields (such as solenoids), allow a least 3" (8 cm), more if necessary.

The OP6800 has strong environmental resistance and high reliability, but you can maximize system reliability by avoiding or eliminating the following conditions at the installation site.

- Abrupt temperature changes and condensation
- Ambient temperatures exceeding a range of 0°C to 50°C
- Relative humidity exceeding a range of 5% to 95%
- Strong magnetism or high voltage
- Corrosive gasses
- Direct vibration or shock
- Excessive iron dust or salt
- Spray from harsh chemicals

## 6.2 Mounting Instructions

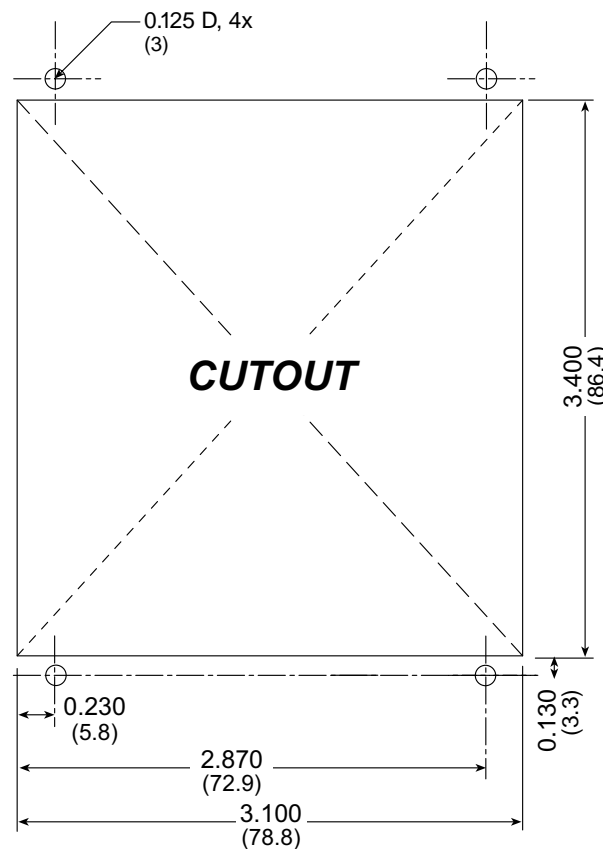
A bezel and a gasket are included with the OP6800. When properly mounted in a panel, the bezel of the OP6800 is designed to meet NEMA 4 specifications for water resistance.

Since the OP6800 employs an LCD display, the viewing angle must be considered when mounting the display. Install the OP6800 at a height and angle that makes it easy for the operator to see the screen.

### 6.2.1 Bezel-Mount Installation

This section describes and illustrates how to bezel-mount the OP6800. Follow these steps for bezel-mount installation.

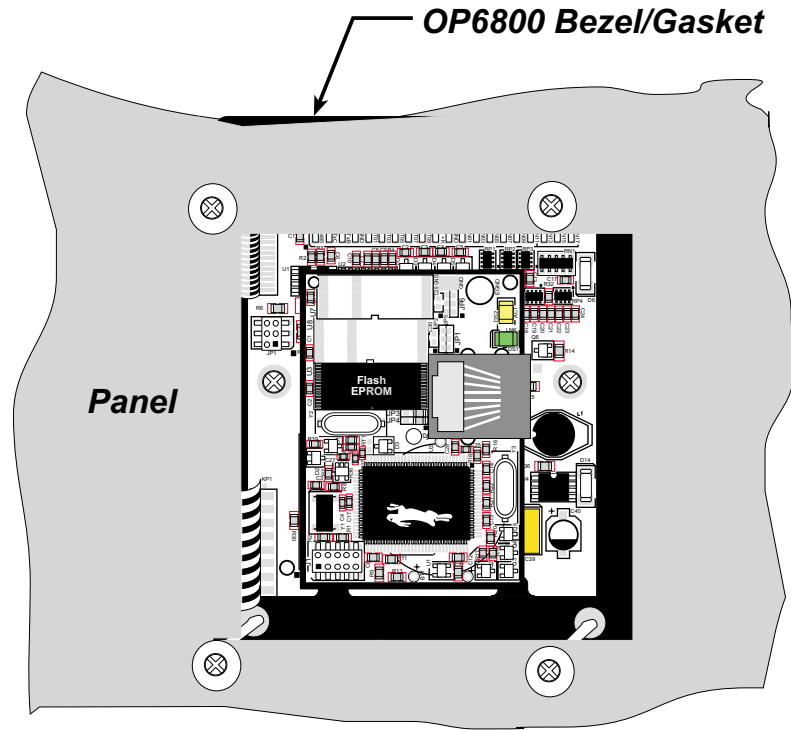
1. Cut mounting holes in the mounting panel in accordance with the recommended dimensions in Figure 19, then use the bezel faceplate to mount the OP6800 onto the panel.



**Figure 19. Recommended Cutout Dimensions**

2. Remove the standoffs added to the OP6800 as described in Chapter 2, “Getting Started.” The standoffs were used to prop up the OP6800 beside the Demonstration Board, and are not needed to mount the OP6800.
3. Carefully “drop in” the OP6800 with the bezel and gasket attached.

4. Fasten the unit with the four 4-40 screws and washers included with the OP6800. If your panel is thick, use a 4-40 screw that is approximately 3/16" (5 mm) longer than the thickness of the panel.



**Figure 20. OP6800 Mounted in Panel (rear view)**

Carefully tighten the screws until the gasket is compressed and the plastic bezel faceplate is touching the panel.

Do not tighten each screw fully before moving on to the next screw. Apply only one or two turns to each screw in sequence until all are tightened manually as far as they can be so that the gasket is compressed and the plastic bezel faceplate is touching the panel.



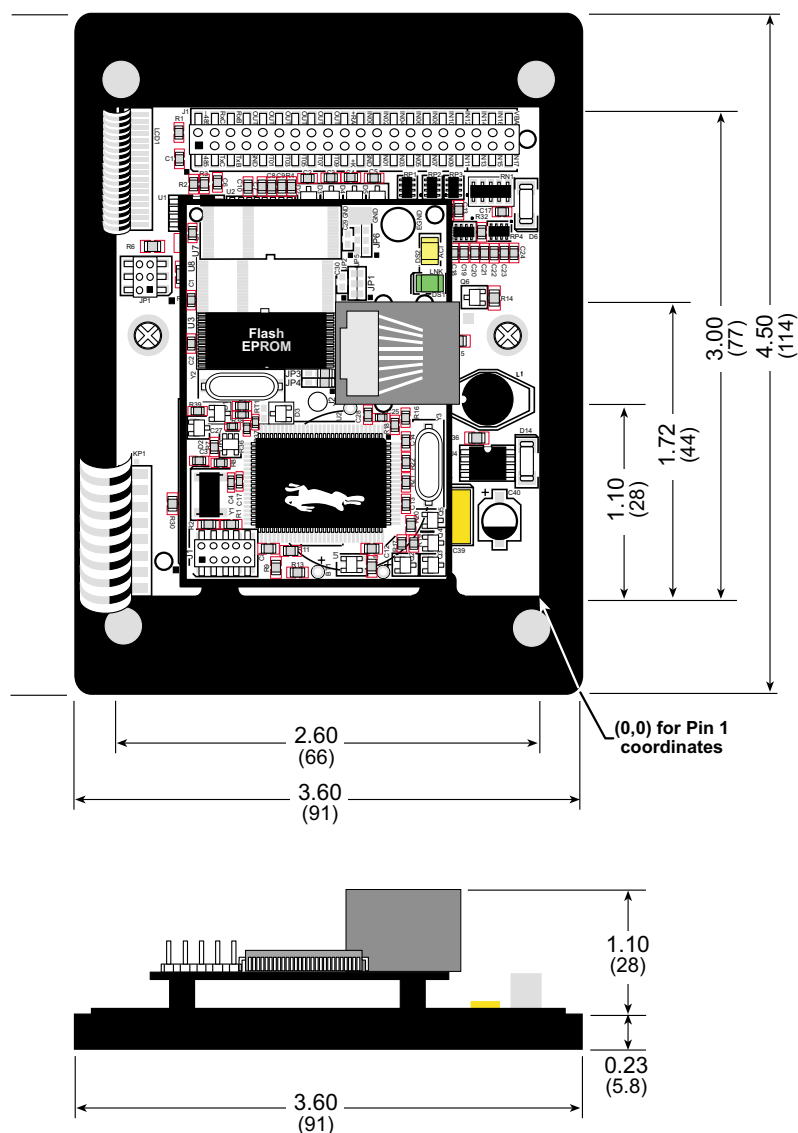


## **APPENDIX A. SPECIFICATIONS**

Appendix A provides the specifications for the OP6800 and describes the conformal coating.

## A.1 Electrical and Mechanical Specifications

Figure A-1 shows the mechanical dimensions for the OP6800.



**Figure A-1. OP6800 Dimensions**

**NOTE:** All measurements are in inches followed by millimeters enclosed in parentheses.

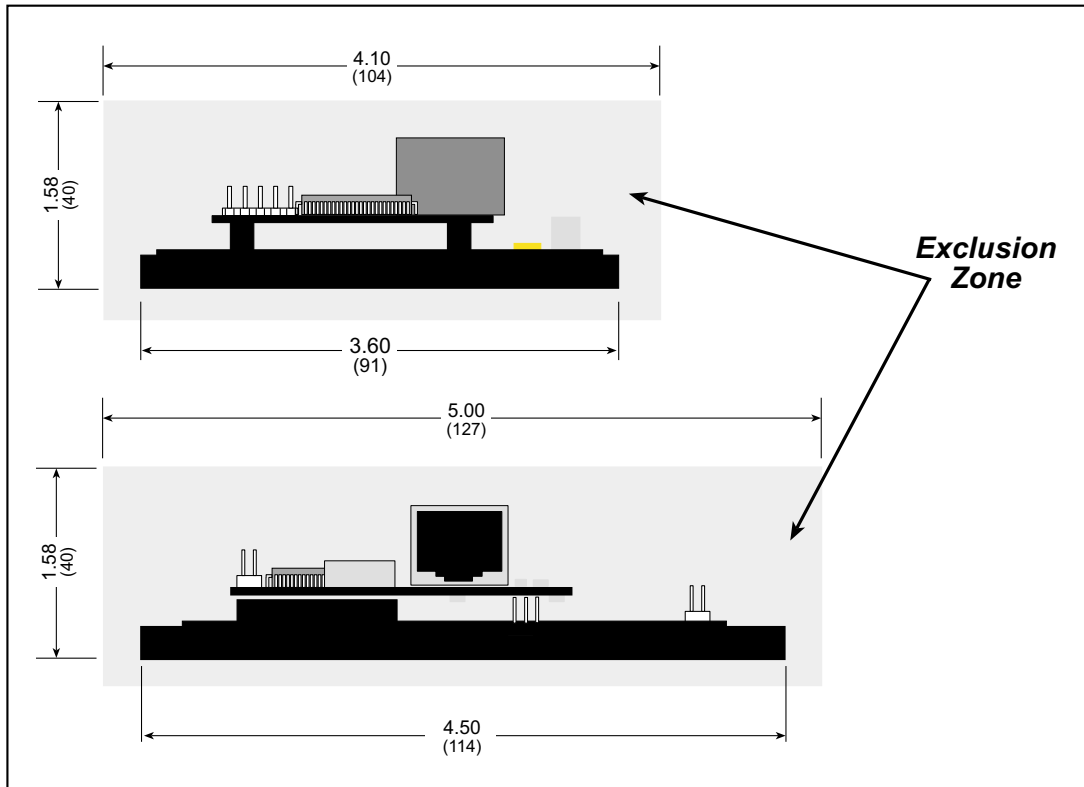
Table A-1 provides the pin 1 locations for the OP6800 headers as viewed in Figure A-1.

**Table A-1. OP6800 Header J1  
Pin 1 Locations**

Header	Pin 1 (x,y) Coordinates (inches)
J1	(-2.101, 2.720)



It is recommended that you allow for an “exclusion zone” of 0.25" (6 mm) around the OP6800 in all directions when the OP6800 is incorporated into an assembly that includes other components. This “exclusion zone” that you keep free of other components and boards will allow for sufficient air flow, and will help to minimize any electrical or EMI interference between adjacent boards. Figure A-2 shows this “exclusion zone.”



**Figure A-2. OP6800 “Exclusion Zone”**

Table A-2 lists the electrical, mechanical, and environmental specifications for the OP6800.

**Table A-2. OP6800 Specifications**

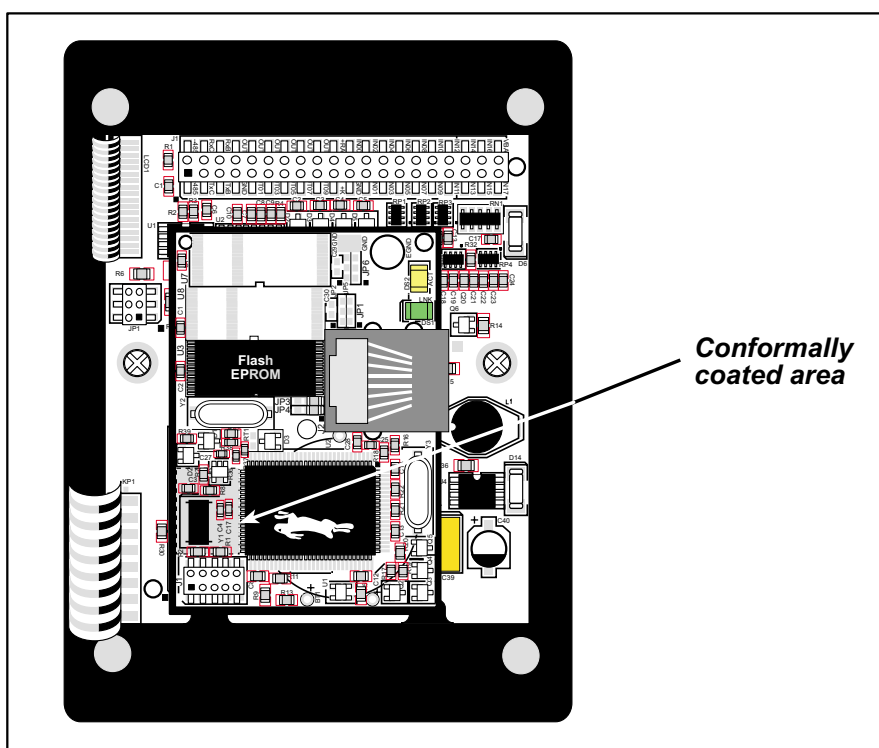
Parameter	Specification
Board Size	2.60" × 3.00" × 1.10" (66 mm × 76 mm × 28 mm)
Bezel Size	4.50" × 3.60" × 0.23" (114 mm × 91 mm × 6 mm)
Connectors	one RJ-45 (Ethernet) (OP6800) one 2 × 20, 0.1" pitch ribbon-cable header
Ethernet Interface (OP6800)	Direct connection to 10Base-T Ethernet networks via RJ-45 connection
Temperature	Operating Range: 0°C to +50°C Storage Range: -40°C to +85°C
Humidity	5% to 95%, noncondensing
External Input Voltage	9 V to 36 V DC
Power Consumption	1.5 W maximum
Digital I/O	24 digital I/O: 8 inputs (IN00–IN07): filtered pulled up, ± 36 V DC, 27 kΩ input impedance, switching threshold 2.4 V typical 5 inputs (IN08–IN12): filtered pulled up, ± 25 V DC, 12 kΩ input impedance, switching threshold 2.4 V typical 7 outputs (OUT00–OUT06): sinking, with LED indicators, +40 V DC, 200 mA maximum per channel 4 outputs (OUT07–OUT10): sinking, open collector with inductive load-protection diode built in , +40 V DC, 200 mA maximum per channel
Microprocessor	Rabbit 2000™
Clock	22.1 MHz
SRAM	128K, surface mount
Flash EPROM	256K, surface mount
Timers	Five 8-bit timers, one 10-bit timer with two match registers, five timers are cascable
Serial Ports	4 serial ports: <ul style="list-style-type: none"> <li>• two RS-232 or one RS-232 (with CTS/RTS)</li> <li>• one RS-485, onboard network termination and bias resistors</li> <li>• one 5 V CMOS-compatible programming port</li> </ul>
Serial Rate	Maximum standard asynchronous 230,400 bps
LCD Panel Size	122 × 32 graphic display
Keypad	7-key keypad

**Table A-2. OP6800 Specifications (continued)**

Parameter	Specification
Watchdog/Supervisor	Yes
Time/Date Clock	Yes
Backup Battery	Connection available for external battery to back up real-time clock and SRAM

## A.2 Conformal Coating

The areas around the crystal oscillator and the battery backup circuit on the OP6800 module 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 contaminants over time.



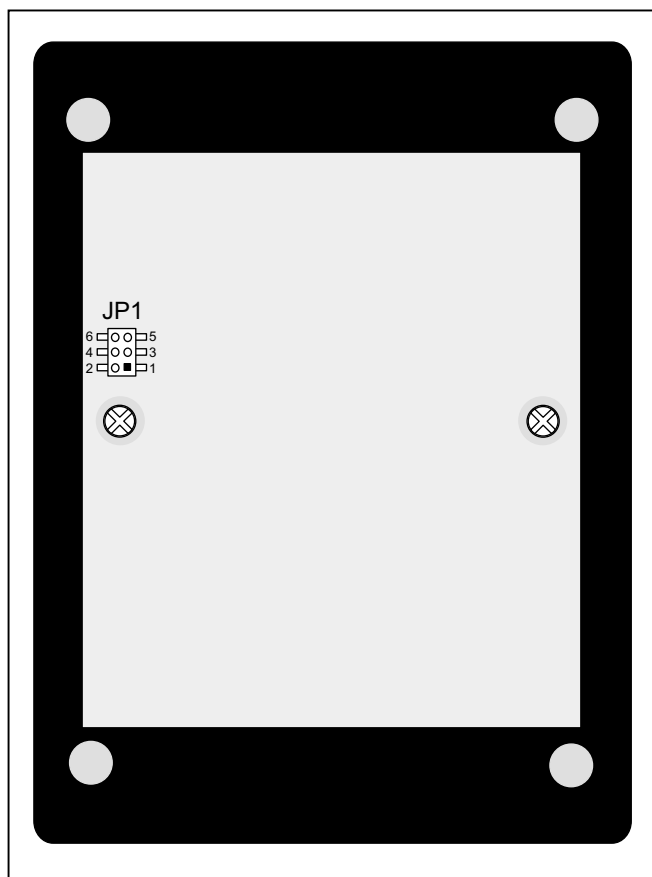
**Figure A-3. OP6800 Areas Receiving Conformal Coating**

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 Rabbit Semiconductor Technical Note 303, *Conformal Coatings*.

## A.3 Jumper Configurations

Figure A-4 shows the header locations used to configure the various OP6800 options via jumpers.



**Figure A-4. Location of BL2100 Configurable Positions**

Table A-3 lists the configuration options.

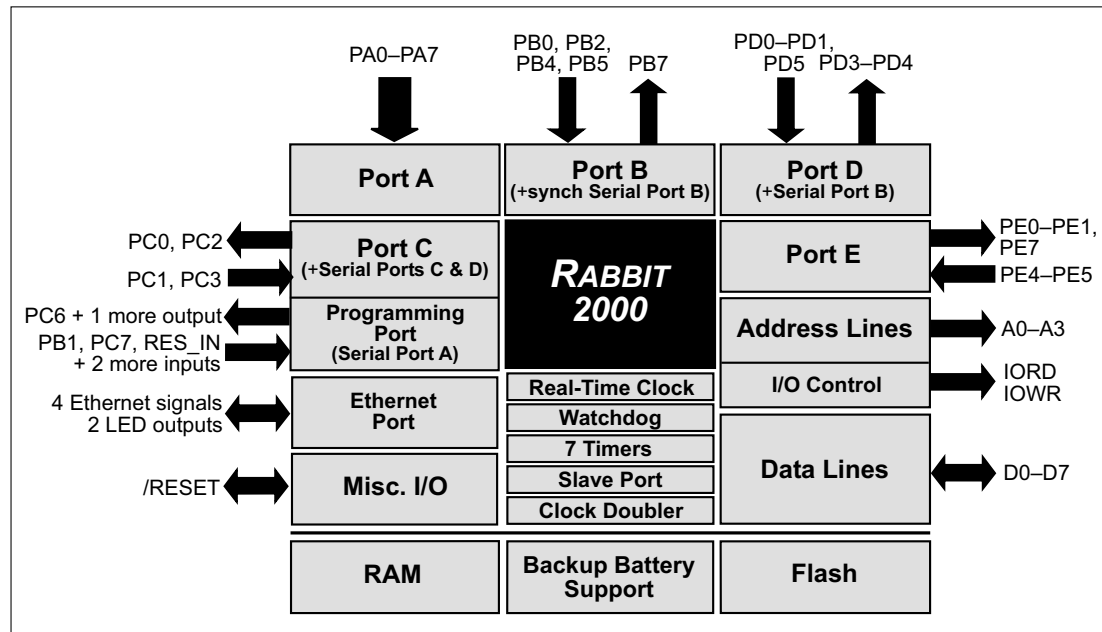
**Table A-3. OP6800 Jumper Configurations**

Header	Description	Pins Connected		Factory Default
JP1	RS-485 Bias and Termination Resistors	1–2 5–6	Bias and termination resistors connected	×
		1–3 4–6	Bias and termination resistors <i>not</i> connected*	

\* Although pins 1–3 and 4–6 of header JP1 are shown “jumpared” for the termination and bias resistors *not* connected, pins 3 and 4 are not actually connected to anything, and this configuration is a “parking” configuration for the jumpers so that they will be readily available should you need to enable the termination and bias resistors in the future.

## A.4 Use of Rabbit 2000 Parallel Ports

Figure A-5 shows the Rabbit 2000 parallel ports.



**Figure A-5. OP6800 Rabbit-Based Subsystems**

Table A-4 lists the Rabbit 2000 parallel ports and their use in the OP6800.

**Table A-4. Use of Rabbit 2000 Parallel Ports**

Port	I/O	Signal	Output Function State
PA0	Input	IN00	Pulled up
PA1	Input	IN01	Pulled up
PA2	Input	IN02	Pulled up
PA3	Input	IN03	Pulled up
PA4	Input	IN04	Pulled up
PA5	Input	IN05	Pulled up
PA6	Input	IN06	Pulled up
PA7	Input	IN07	Pulled up
PB0	Input	IN08	Pulled up
PB1	Input	Not Used	Pulled up
PB2	Input	IN09	Pulled up
PB3	Input	IN10	Pulled up
PB4	Input	IN11	Pulled up

**Table A-4. Use of Rabbit 2000 Parallel Ports (continued)**

Port	I/O	Signal		Output Function State
PB5	Input	Connected to PB7		Driven by PB7
PB6	Output	Not Used		Low
PB7	Output	Connected to PB5		Low
PC0	Output	TXD RS-485	Serial Port D	Inactive high
PC1	Input	RXD RS-485		Inactive high
PC2	Output	RTS/TXC RS-232	Serial Port C	Inactive high
PC3	Input	CTS/RXC RS-232		Inactive high
PC4	Output	TPOUT– (Realtek reset)		Initialized by <b>sock_init</b>
PC5	Input	TPOUT+ (Realtek INT0)		Pulled up
PC6	Output	TXA Programming Port	Serial Port A	Inactive high
PC7	Input	RXA Programming Port		Pulled up
PD0	Input Output	Realtek CLK (OP6800) Not used (OP6810)		Initialized by <b>sock_init</b> Low
PD1	Input Output	Realtek SDO (OP6800) Not used (OP6810)		Initialized by <b>sock_init</b> Low
PD2	Output	Not used		Low
PD3	Output	OUT07		Low (output driver off)
PD4	Output	ATXB RS-232	Serial Port B	Inactive high
PD5	Input	ARXB RS-232		Inactive high
PD6	Output	Not used		Low
PD7	Output	Not used		Low
PE0	Output	RS-485 control register		Low (Tx disabled)
PE1	Output	OUT08		Low (output driver off)
PE2	N/A Output	Realtek IORB strobe (OP6800) Not used (OP6810)		Initialized by <b>sock_init</b> Low
PE3	N/A Output	Realtek SDI line (OP6800) Not used (OP6810)		Initialized by <b>sock_init</b> Low
PE4	Input	OUT09		Low (output driver off)
PE5	Input	OUT10		Low (output driver off)
PE6	N/A Output	Realtek IOWB strobe (OP6800) Not used (OP6810)		Initialized by <b>sock_init</b> Low
PE7	Output	LCD_KEYPAD strobe		Inactive high

## A.5 I/O Address Assignments

Table A-5 lists the external I/O addresses for the display and keypad I/O.

**Table A-5. Display and Keypad Output Addresses**

External Address	Name	Function
E000–E007	<b>LCD</b>	LCD control
E008	<b>EN</b>	Output enable for LEDs
E00A	<b>KPEN</b>	Read keypad and IN12
E00B	<b>LED</b>	LED0–LED6 and LCD backlight

PE7 serves as a system-enable control and LCD/keypad strobe. When PE7 is high or in a high-impedance status, all OP6800 outputs are disabled (digital outputs and display outputs are disabled, and RS-485 is at listen status).



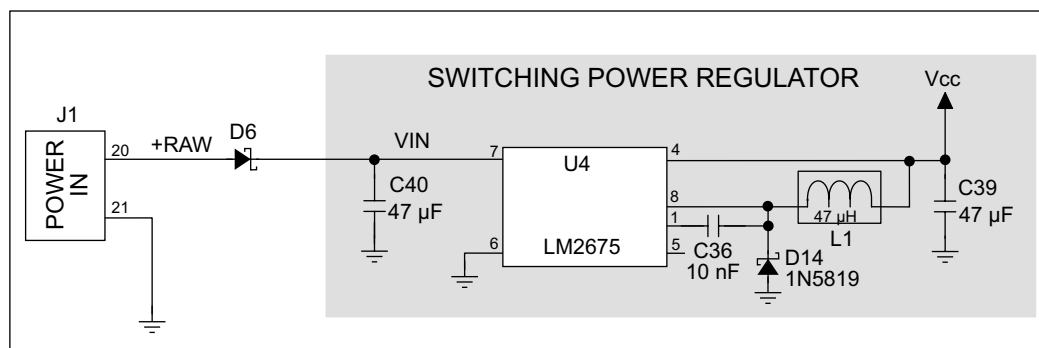


## APPENDIX B. POWER SUPPLY

Appendix B describes the power circuitry provided on the OP6800.

### B.1 Power Supplies

Power is supplied to the OP6800 via pins 20 and 21 of header J1, which is connected by a ribbon cable to either the Demonstration Board or to your system. The OP6800 is protected against reverse polarity by a diode at D6 as shown in Figure B-1.



**Figure B-1. OP6800 Power Supply**

The input voltage range is from 9 V to 36 V. A switching power regulator is used to provide a Vcc of +5 V for the OP6800 logic circuits. Vcc is not accessible to the user.

**NOTE:** In addition to supplying +RAW to the OP6800 switching power regulator, the Demonstration Board has its own independent linear power regulator to supply the electronics in the demonstration area of the Demonstration Board. See Appendix C for more information.

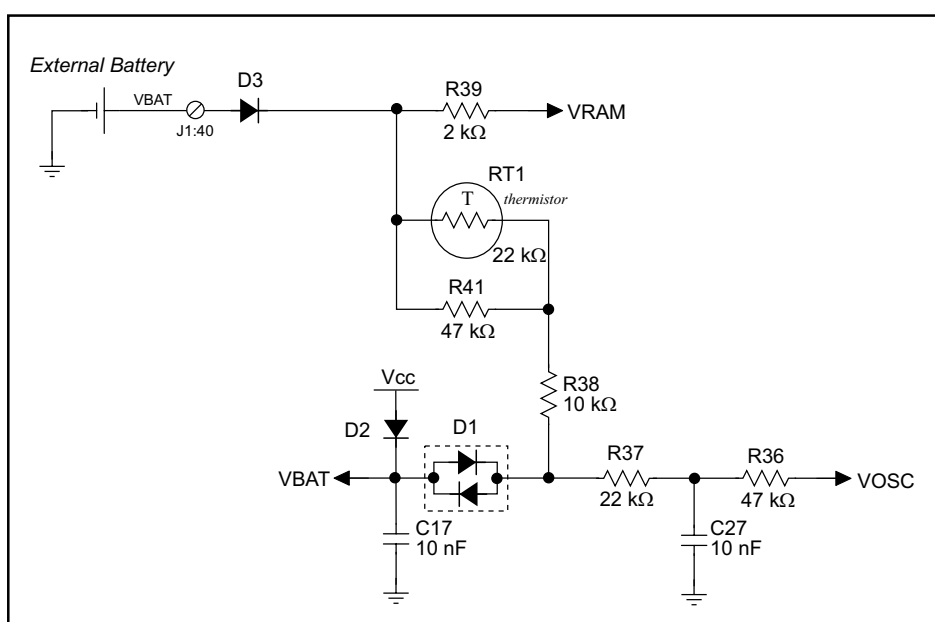
## B.2 Batteries and External Battery Connections

The SRAM and the real-time clock have provision for battery backup. Power to the SRAM and the real-time clock (VRAM) is provided by two different sources, depending on whether the main part of the OP6800 is powered or not. When the OP6800 is powered normally, and Vcc is within operating limits, the SRAM and the real-time clock are powered from Vcc. If power to the board is lost or falls below 4.63 V, the VRAM and real-time clock power must come from a backup battery in your system which you would connect to pin 40 of header J1 on the OP6800 via the ribbon cable. The backup battery should be able to supply 2.85 V–3.15 V at 10  $\mu$ A.

The reset generator circuit controls the source of power by way of its **/RESET** output signal.

### B.2.1 Battery-Backup Circuit

Figure B-2 shows the battery-backup circuit located on the OP6800 module.



**Figure B-2. OP6800 Backup Battery Circuit**

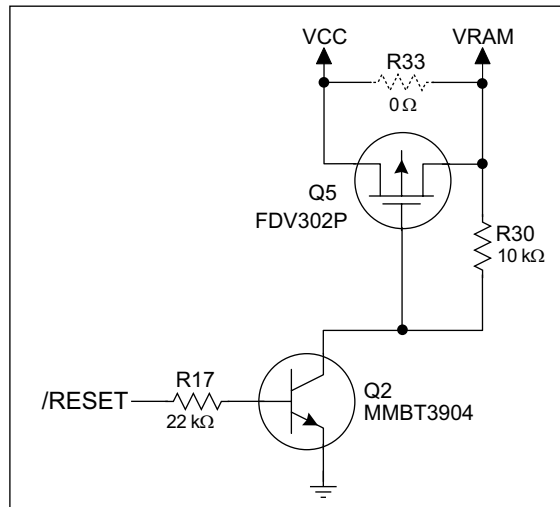
The battery-backup circuit serves three purposes:

- 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 U6, 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 OP6800.

### B.2.2 Power to VRAM Switch

The VRAM switch on the OP6800 module, shown in Figure B-3, 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-3. VRAM Switch**

Field-effect transistor Q5 is needed to provide a very small voltage drop between Vcc and VRAM (<100 mV, typically 10 mV) so that the board components powered by Vcc will not have a significantly different voltage than VRAM.

When the OP6800 is *not* in reset, the **/RESET** line will be high. This turns on Q2, causing its collector to go low. This turns on Q5, allowing VRAM to nearly equal Vcc.

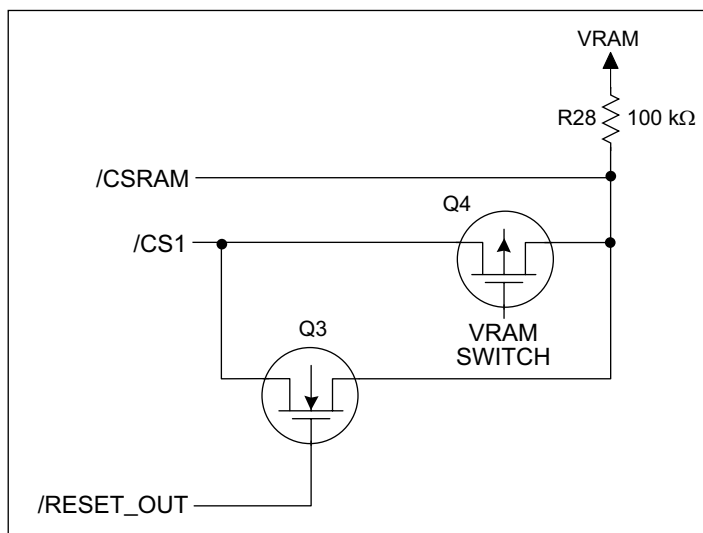
When the OP6800 *is* in reset, the **/RESET** line will go low. This turns off Q2 and Q5, providing an isolation between Vcc and VRAM.

### B.2.3 Reset Generator

The OP6800 module uses a reset generator on the module, U1, 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.3 Chip Select Circuit

Figure B-4 shows a schematic of the chip select circuit located on the OP6800 module.



**Figure B-4. Chip Select Circuit**

The current drain on the battery in a battery-backed circuit must be kept at a minimum. When the OP6800 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  $V_{cc}$  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 SRAM's chip select 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). Q3 and Q4 are MOSFET transistors with complementary polarity. They are both turned on when power is applied to the circuit. They allow the CS signal to pass from the processor to the SRAM so that the processor can periodically access the SRAM. When power is removed from the circuit, the transistors will turn off and isolate /CSRAM from the processor. The isolated /CSRAM line has a 100 kΩ pullup resistor to VRAM (R28). This pullup resistor keeps /CSRAM at the VRAM voltage level (which under no power condition is the backup battery's regulated voltage at a little more than 2 V).

Transistors Q3 and Q4 are of opposite polarity so that a rail-to-rail voltage can be passed. When the /CS1 voltage is low, Q3 will conduct. When the /CS1 voltage is high, Q4 conducts. It takes time for the transistors to turn on, creating a propagation delay. This propagation delay is typically very small, about 10 ns to 15 ns.

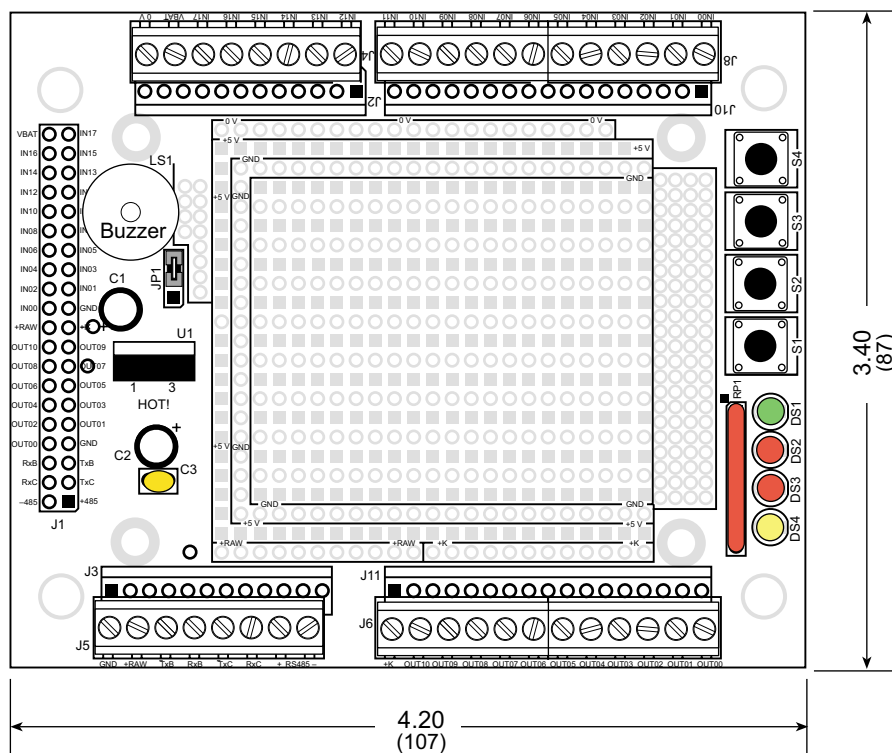


## **APPENDIX C. DEMONSTRATION BOARD**

Appendix C describes the features and accessories of the Demonstration Board, and explains the use of the Demonstration Board to demonstrate the OP6800 and to build prototypes of your own circuits.

## C.1 Mechanical Dimensions and Layout

Figure C-1 shows the mechanical dimensions and layout for the OP6800 Demonstration Board.



**Figure C-1. OP6800 Demonstration Board Dimensions**

Table C-1 lists the electrical, mechanical, and environmental specifications for the Demonstration Board.

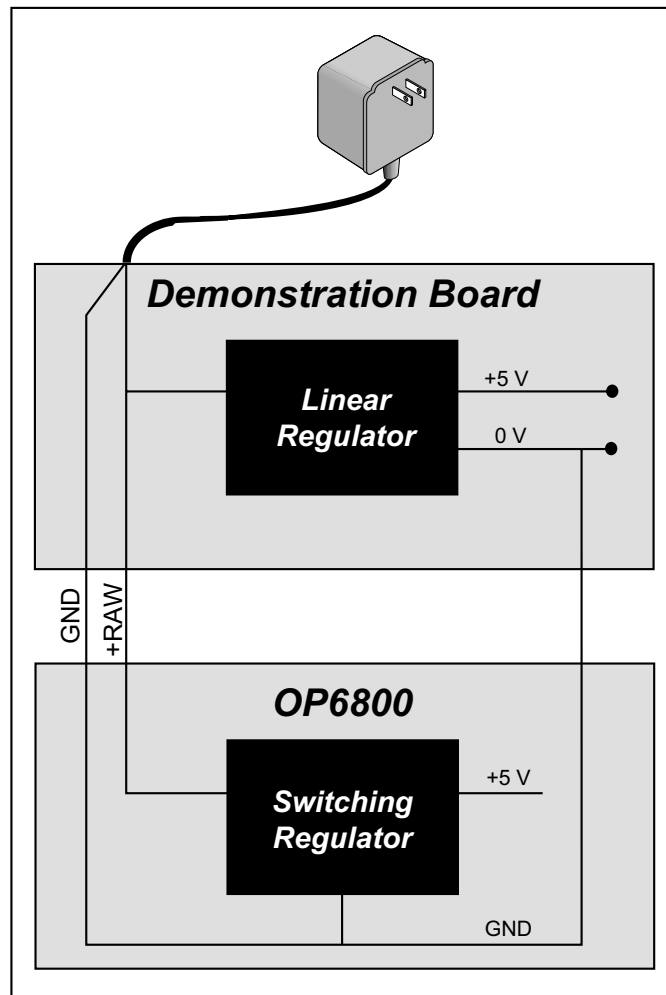
**Table C-1. Demonstration Board Specifications**

Parameter	Specification
Board Size	3.40" × 4.20" × 1.19" (87 mm × 107 mm × 30 mm)
Operating Temperature	−40°C to +70°C
Humidity	5% to 95%, noncondensing
Input Voltage	7.5 V to 25 V DC
Maximum Current Draw (including user-added circuits)	140 mA at 12 V and 25°C, 100 mA at 12 V and 70°C
Prototyping Area	1.7" × 2.1" (43 mm × 53 mm) through hole, 0.1" spacing
Standoffs/Spacers	4, accept 4-40 x 11/8 screws

## C.2 Power Supply

The OP6800 requires an unregulated +RAW power input of 9 V to 36 V DC, which can be supplied from the Demonstration Board through the ribbon cable connection. The OP6800 has its own switching voltage regulator.

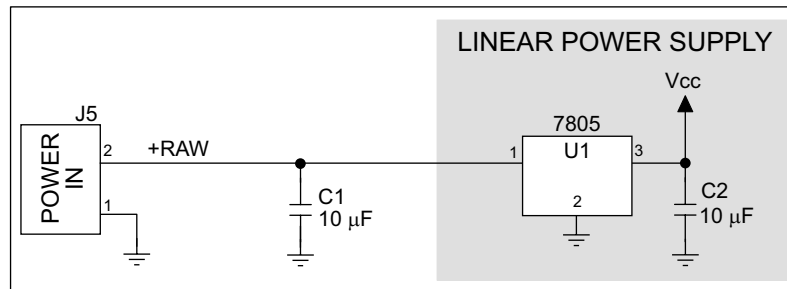
Figure C-2 shows the distribution of the +RAW input power to the OP6800 through the Demonstration Board. The reference grounds on the OP6800, GND, and on the Demonstration Board, 0 V, are tied together at one connection point only to avoid creating a ground loop, which could lead to considerable electromagnetic interference.



**Figure C-2. Power Distribution to OP6800 and Demonstration Board**

The Demonstration Board has an onboard LM7805 linear regulator for the circuits on the Demonstration Board only. Its major drawback is its inefficiency, which is directly proportional to the voltage drop across it. The voltage drop creates heat and wastes power.

You may wish to use a switching power supply in your applications where better efficiency is desirable. The LM2575 is an example of an easy-to-use switching voltage regulator. This part greatly reduces the heat dissipation of the regulator. The drawback in using a switching voltage regulator is its higher cost.



**Figure C-3. Demonstration Board Power Supply**

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

Be careful to limit the current draw in any prototype circuits you build on the prototyping area of the Demonstration Board to avoid operating the linear regulator outside its recommended limits. The LEDs and buzzer together can draw up to 70 mA, which still leaves some current capacity for your own circuits (see Table C-1) if you plan to use them with the LEDs and the buzzer.

If you need additional current from the linear regulator beyond that specified in Table C-1, consider adding a heat sink to the linear regulator (remember to use silicone grease between the tab and the heat sink), or use a lower voltage power supply.

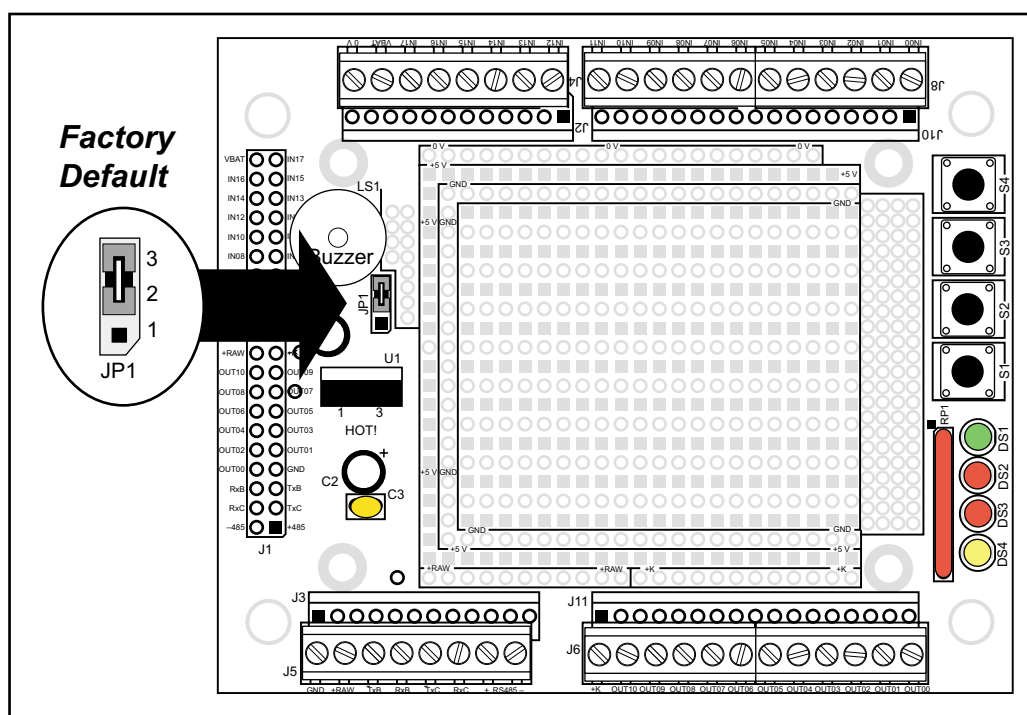


### C.3 Using the Demonstration Board

The Demonstration Board is actually both a demonstration board and a prototyping board. As a demonstration board, it can be used to demonstrate the functionality of the OP6800 right out of the box without any modifications to either board. There are no jumpers or dip switches to configure or misconfigure on the Demonstration Board so that the initial setup is very straightforward.

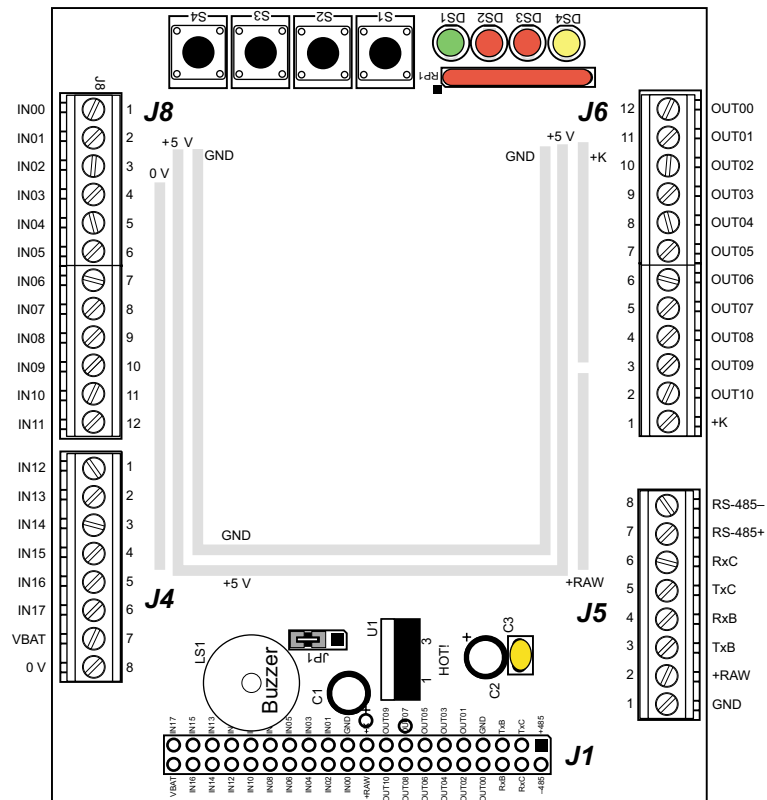
The Demonstration Board comes with the basic components necessary to demonstrate the operation of the OP6800. Four LEDs (DS1–DS4) are connected to OUT07–OUT10, and four switches (S1–S4) are connected to IN00–IN03 to demonstrate the interface to the OP6800.

The Demonstration Board has a buzzer that is normally off. The buzzer can be enabled to be on by setting the jumper across pins 1–2 on header JP1 on the Demonstration Board as shown in Figure C-4. When enabled on, the buzzer will sound whenever the OUT0 digital output on the OP6800 is on.



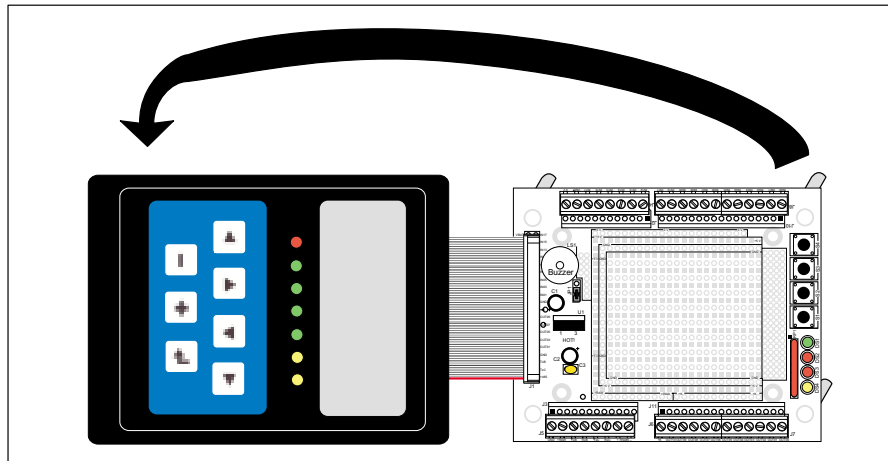
**Figure C-4. Demonstration Board Header JP1  
(Buzzer On/Off)**

The Demonstration Board provides the user with OP6800 connection points brought out conveniently to labeled points at headers J4, J5, J6, and J8 on the Demonstration Board. Small to medium circuits can be prototyped using point-to-point wiring with 20 to 30 AWG wire on the prototyping area. The holes are spaced at 0.1" (2.5 mm). The pinouts for headers J4, J5, J6, and J8 are shown in Figure C-5.



**Figure C-5. OP6800 Demonstration Board Pinout**

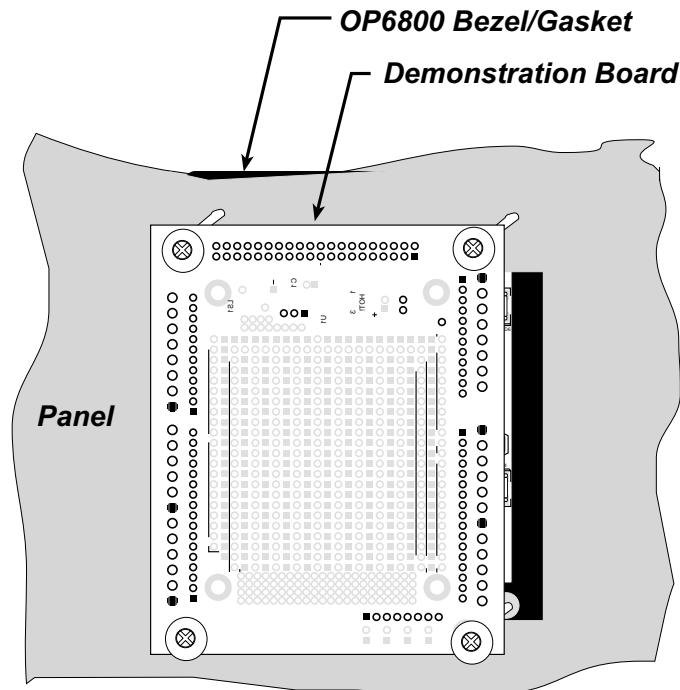
The Demonstration Board can then be rotated and mounted behind the OP6800 as shown in Figure C-6 to allow the Demonstration Board and the OP6800 to be used together.



**Figure C-6. Mounting Demonstration Board on OP6800**

**NOTE:** Remove the standoffs behind the OP6800 before attempting to mount the Demonstration Board.

The OP6800 may also be panel-mounted with the Demonstration Board attached. Follow the instructions in Chapter 6, “Installation and Mounting Guidelines.” Use 4-40 screws that are 1 3/16" (plus the thickness of the panel) in length. Note that the Demonstration Board and the OP6800 end up on opposite sides of the panel as shown in Figure C-7.



**Figure C-7. OP6800 with Demonstration Board Mounted in Panel (rear view)**





## **APPENDIX D. OP6800 FUNCTION APIs**

Appendix D provides the function calls related to the operation of the OP6800 board, I/O, serial channels, display, and keypad.

## D.1 Board Initialization (OP68xx.LIB)

```
void brdInit (void);
```

Call this function at the beginning of your program. This function initializes the system I/O ports. This function also turns off LED DS1 to indicate that the initialization was successful.

The ports are initialized according to Table A-4.

### SEE ALSO

`digIn`, `digOut`, `serMode`, `ledOut`

## D.2 Digital I/O (OP68xx.LIB)

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

Reads the state of an input channel.

A runtime error will occur if **brdInit** was not executed before executing **digIn**, or when **channel** is out of range.

### PARAMETER

**channel** is the input channel number (0–12), where IN00–IN12 are the normal user digital inputs.

### RETURN VALUE

The state of the input (0 or 1).

### SEE ALSO

**brdInit**, **digOut**, **ledOut**

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

Sets the state of a digital output (OUT00–OUT10).

Remember to call the **brdInit** function before executing this function.

A runtime error will occur if **brdInit** was not executed before executing **digOut**, or when **channel** or **value** is out of range.

**NOTE:** The LEDs and digital outputs OUT00–OUT06 are driven by the same driver chip. Do not use both **ledOut** and **digOut** to control the same LED or digital output in a given application.

### PARAMETERS

**channel** is the output channel number (0–10).

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

### SEE ALSO

**brdInit**, **digIn**, **ledOut**

## D.3 Serial Communication (OP68xx.LIB)

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*.

Use the following function calls with the OP6800.

```
int serMode(int mode);
```

User interface to set up OP6800 serial communication lines. Call this function after **serXOpen( )**.

Whether you are opening one or multiple serial ports, this function must be executed after executing the last **serXOpen** function AND before you start using any of the serial ports. This function is non-reentrant.

If Mode 1 is selected, CTS/RTS flow control is exercised using the **serCflowcontrolOn** and **serCflowcontrolOff** functions from the **RS232.LIB** library.

### PARAMETER

**mode** is the defined serial port configuration.

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

### RETURN VALUE

0 if valid mode, 1 if not.

### SEE ALSO

**ser485Tx**, **ser485Rx**

**NOTE:** Be sure to call **serMode** before either of the next two functions.

```
void ser485Tx(void);
```

Sets pin 3 (DE) high to enable the RS-485 transmitter. Remember to call **serMode** before calling **ser485Tx**.

### SEE ALSO

**serMode**, **ser485Rx**

```
void ser485Rx(void);
```

Resets pin 3 (DE) low to disable the RS-485 transmitter. Remember to call **serMode** before calling **ser485Rx**.

### SEE ALSO

**serMode**, **ser485Tx**, **serCflowcontrolOn**, **serCflowcontrolOff**



## D.4 LEDs (OP68xx.LIB)

When power is applied to the OP6800 for the first time, the red LED (DS1) will come on, indicating that power is being applied to the OP6800. The red LED is turned off when the **brdInit** function executes.

The LEDs are in series with the open-ouput collector that drives digital outputs OUT00–OUT06, and so the same function call that turns on one of these digital outputs will also turn on the corresponding LED.

```
void ledOut(int led, int value);
```

LED on/off control.

A runtime error will occur if **brdInit** was not executed before executing **ledOut**, or when **led** or **value** is out of range.

**NOTE:** The LEDs and digital outputs OUT00–OUT06 are driven by the same driver chip. Do not use both **ledOut** and **digOut** to control the same LED or digital output in a given application.

### PARAMETERS

**led** is the LED to control.

- 0 = LED DS1
- 1 = LED DS2
- 2 = LED DS3
- 3 = LED DS4
- 4 = LED DS5
- 5 = LED DS6
- 6 = LED DS7

**value** is the value used to control whether the LED is on or off (0 or 1).

- 0 = off
- 1 = on

### RETURN VALUE

None.

### SEE ALSO

**brdInit**, **digOut**

## D.5 LCD Display

The functions used to control the LCD display are contained in the **GRAPHIC.LIB** library located in the Dynamic C **DISPLAYS\GRAPHIC** library directory.

```
void glInit(void);
```

Initializes the display devices, clears the screen.

### RETURN VALUE

None.

### SEE ALSO

`glDispOnOFF`, `glBacklight`, `glSetContrast`, `glPlotDot`, `glBlock`, `glPlotDot`,  
`glPlotPolygon`, `glPlotCircle`, `glHScroll`, `glVScroll`, `glXFontInit`, `glPrintf`,  
`glPutChar`, `glSetBrushType`, `glBuffLock`, `glBuffUnlock`, `glPlotLine`

```
void glBackLight(int onOff);
```

Sets the intensity of the backlight, if circuitry is installed.

### PARAMETER

: **onOff** reflects the low to high values (typically 0 to 255, depending on the board design) to set the back-light intensity (0 will turn the backlight off completely.)

### RETURN VALUE

None.

### SEE ALSO

`glInit`, `glDispOnoff`, `glSetContrast`

```
void glDispOnOff(int onOff);
```

Sets the LCD screen on or off. Data will not be cleared from the screen.

### PARAMETER

**onOff** turns the LCD screen on or off

1—turn the LCD screen on

0—turn the LCD screen off

### RETURN VALUE

None.

### SEE ALSO

`glInit`, `glSetContrast`, `glBackLight`

```
void glSetContrast(unsigned level);
```

Sets display contrast (the circuitry is *not* installed on the LCD/keypad module used with the OP6800).

**PARAMETER**

**level** reflects low to high values (typically 0 to 255, depending on the board design) to give high to low contrast respectively.

**RETURN VALUE**

None.

**SEE ALSO**

`glInit`, `glBacklight`, `glDispOnoff`

```
void glFillScreen(char pattern);
```

Fills the LCD display screen with a pattern.

**PARAMETER**

The screen will be set to all black if **pattern** is 0xFF, all white if **pattern** is 0x00, and vertical stripes for any other pattern.

**RETURN VALUE**

None.

**SEE ALSO**

`glBlock`, `glBlankScreen`, `glPlotPolygon`, `glPlotCircle`

```
void glBlankScreen(void);
```

Blanks the LCD display screen (sets LCD display screen to white).

**RETURN VALUE**

None.

**SEE ALSO**

`glFillScreen`, `glBlock`, `glPlotPolygon`, `glPlotCircle`

```
void glBlock(int x, int y, int bmWidth,  
            int bmHeight);
```

Draws a rectangular block in the page buffer and on the LCD if the buffer is unlocked. Any portion of the block that is outside the LCD display area will be clipped.

#### PARAMETERS

**x** is the *x* coordinate of the upper left corner of the block.

**y** is the *y* coordinate of the left top corner of the block.

**bmWidth** is the width of the block.

**bmHeight** is the height of the block.

#### RETURN VALUE

None.

#### SEE ALSO

`glFillScreen`, `glBlankScreen`, `glPlotPolygon`, `glPlotCircle`

```
void glPlotVPolygon(int n, int *pFirstCoord);
```

Plots the outline of a polygon in the LCD page buffer, and on the LCD if the buffer is unlocked. Any portion of the polygon that is outside the LCD display area will be clipped. The function will also return, doing nothing, if there are less than 3 vertices.

#### PARAMETERS

**n** is the number of vertices.

**\*pFirstCoord** is a pointer to array of vertex coordinates: **x1,y1,x2,y2,x3,y3,...**

#### RETURN VALUE

None.

#### SEE ALSO

`glPlotPolygon`, `glFillPolygon`, `glFillVPolygon`

```
void glPlotPolygon(int n, int y1, int x2, int y2,  
...);
```

Plots the outline of a polygon in the LCD page buffer and on the LCD if the buffer is unlocked. Any portion of the polygon that is outside the LCD display area will be clipped. The function will also return, doing nothing, if there are less than 3 vertices.

#### PARAMETERS

**n** is the number of vertices.

**y1** is the y coordinate of the first vertex.

**x1** is the x coordinate of the first vertex.

**y2** is the y coordinate of the second vertex.

**x2** is the x coordinate of the second vertex.

**...** are the coordinates of additional vertices.

#### RETURN VALUE

None.

#### SEE ALSO

`glPlotVPolygon`, `glFillPolygon`, `glFillVPolygon`

```
void glFillVPolygon(int n, int *pFirstCoord);
```

Fills a polygon in the LCD page buffer and on the LCD screen if the buffer is unlocked. Any portion of the polygon that is outside the LCD display area will be clipped. The function will also return, doing nothing, if there are less than 3 vertices.

#### PARAMETERS

**n** is the number of vertices.

**\*pFirstCoord** is a pointer to array of vertex coordinates: **x1,y1,x2,y2,x3,y3,...**

#### RETURN VALUE

None.

#### SEE ALSO

`glFillPolygon`, `glPlotPolygon`, `glPlotVPolygon`

```
void glFillPolygon(int n, int x1, int y1,  
    int x2, int y2, ...);
```

Fills a polygon in the LCD page buffer and on the LCD if the buffer is unlocked. Any portion of the polygon that is outside the LCD display area will be clipped.

#### PARAMETERS

**n** is the number of vertices.

**x1** is the x coordinate of the first vertex.

**y1** is the y coordinate of the first vertex.

**x2** is the x coordinate of the second vertex.

**y2** is the y coordinate of the second vertex.

**...** are the coordinates of additional vertices.

#### RETURN VALUE

None.

#### SEE ALSO

`glFillVPolygon`, `glPlotPolygon`, `glPlotVPolygon`

```
void glPlotCircle(int xc, int yc, int rad);
```

Draws a circle in the LCD page buffer and on the LCD if the buffer is unlocked. Any portion of the circle that is outside the LCD display area will be clipped.

#### PARAMETERS

**xc** is the x coordinate of the center of the circle.

**yc** is the y coordinate of the center of the circle.

**rad** is the radius of the center of the circle (in pixels).

#### RETURN VALUE

None.

#### SEE ALSO

`glFillCircle`, `glPlotPolygon`, `glFillPolygon`

```
void glFillCircle(int xc, int yc, int rad);
```

Draws a filled circle in the LCD page buffer and on the LCD if the buffer is unlocked. Any portion of the circle that is outside the LCD display area will be clipped.

#### PARAMETERS

**xc** is the x coordinate of the center of the circle.

**yc** is the y coordinate of the center of the circle.

**rad** is the radius of the center of the circle (in pixels).

#### RETURN VALUE

None.

#### SEE ALSO

`glPlotCircle`, `glPlotPolygon`, `glFillPolygon`

```
void glXFontInit(fontInfo *pInfo, char pixWidth,
                 char pixHeight, unsigned startChar,
                 unsigned endChar, unsigned long xmemBuffer);
```

Initializes the font descriptor structure, where the font is stored in **xmem**. Each font character's bitmap is column major and byte-aligned.

#### PARAMETERS

**\*pInfo** is a pointer to the font descriptor to be initialized.

**pixWidth** is the width (in pixels) of each font item.

**pixHeight** is the height (in pixels) of each font item.

**startChar** is the value of the first printable character in the font character set.

**endChar** is the value of the last printable character in the font character set.

**xmemBuffer** is the **xmem** pointer to a linear array of font bitmaps.

#### RETURN VALUE

None.

#### SEE ALSO

`glPrintf`

```
unsigned long glFontCharAddr(fontInfo *pInfo,
                             char letter);
```

Returns the **xmem** address of the character from the specified font set.

#### PARAMETERS

**\*pInfo** is the **xmem** address of the bitmap font set.

**letter** is an ASCII character.

#### RETURN VALUE

**xmem** address of bitmap character font, column major, and byte-aligned.

#### SEE ALSO

`glPutFont`, `glPrintf`

```
void glPutFont(int x, int y, fontInfo *pInfo,  
char code);
```

Puts an entry from the font table to the page buffer and on the LCD if the buffer is unlocked. Each font character's bitmap is column major and byte-aligned. Any portion of the bitmap character that is outside the LCD display area will be clipped.

#### PARAMETERS

**x** is the *x* coordinate (column) of the upper left corner of the text.

**y** is the *y* coordinate (row) of the left top corner of the text.

**\*pInfo** is a pointer to the font descriptor.

**code** is the ASCII character to display.

#### RETURN VALUE

None.

#### SEE ALSO

`glFontCharAddr`, `glPrintf`

```
void glSetPfStep(int stepX, int stepY);
```

Sets the `glPrintf()` printing step direction. The *x* and *y* step directions are independent signed values. The actual step increments depend on the height and width of the font being displayed, which are multiplied by the step values.

#### PARAMETERS

**stepX** is the `glPrintf` *x* step value

**stepY** is the `glPrintf` *y* step value

#### RETURN VALUE

None.

#### SEE ALSO

Use `glGetPfStep()` to examine the current *x* and *y* printing step direction.

```
int glGetPfStep(void);
```

Gets the current `glPrintf()` printing step direction. Each step direction is independent of the other, and is treated as an 8-bit signed value. The actual step increments depends on the height and width of the font being displayed, which are multiplied by the step values.

#### RETURN VALUE

The *x* step is returned in the MSB, and the *y* step is returned in the LSB of the integer result.

#### SEE ALSO

Use `glGetPfStep()` to control the *x* and *y* printing step direction.



```
void glPutChar(char ch, char *ptr, int *cnt,  
glPutCharInst *pInst)
```

Provides an interface between the **STDIO** string-handling functions and the graphic library. The **STDIO** string-formatting function will call this function, one character at a time, until the entire formatted string has been parsed. Any portion of the bitmap character that is outside the LCD display area will be clipped.

#### PARAMETERS

**ch** is the character to be displayed on the LCD.

**\*ptr** is not used, but is a place holder for **STDIO** string functions.

**\*cnt** is not used, is a place holder for **STDIO** string functions.

**\*pInst** is a font descriptor pointer.

#### RETURN VALUE

None.

#### SEE ALSO

`glPrintf, glPutFont, doprint`

```
void glPrintf(int x, int y, fontInfo *pInfo,  
char *fmt, ...);
```

Prints a formatted string (much like **printf**) on the LCD screen. Only the character codes that exist in the font set are printed, all others are skipped. For example, `\b`, `\t`, `\n` and `\r` (ASCII backspace, tab, new line, and carriage return, respectively) will be printed if they exist in the font set, but will not have any effect as control characters. Any portion of the bitmap character that is outside the LCD display area will be clipped.

#### PARAMETERS

**x** is the *x* coordinate (column) of the upper left corner of the text.

**y** is the *y* coordinate (row) of the upper left corner of the text.

**\*pInfo** is a font descriptor pointer.

**\*fmt** is a formatted string.

**...** are formatted string conversion parameter(s).

#### EXAMPLE

```
glprintf(0,0, &fi12x16, "Test %d\n", count);
```

#### RETURN VALUE

None.

#### SEE ALSO

`glXFontInit`

## **void glBuffLock(void);**

Increments LCD screen locking counter. Graphic calls are recorded in the LCD memory buffer and are not transferred to the LCD if the counter is non-zero.

**NOTE:** `glBuffLock()` and `glBuffUnlock()` can be nested up to a level of 255, but be sure to balance the calls. It is not a requirement to use these procedures, but a set of `glBuffLock()` and `glBuffUnlock()` bracketing a set of related graphic calls speeds up the rendering significantly.

### **RETURN VALUE**

None.

### **SEE ALSO**

`glBuffUnlock`, `glSwap`

## **void glBuffUnlock(void);**

Decrements the LCD screen locking counter. The contents of the LCD buffer are transferred to the LCD if the counter goes to zero.

### **RETURN VALUE**

None.

### **SEE ALSO**

`glBuffLock`, `glSwap`

## **void glSwap(void);**

Checks the LCD screen locking counter. The contents of the LCD buffer are transferred to the LCD if the counter is zero.

### **RETURN VALUE**

None.

### **SEE ALSO**

`glBuffUnlock`, `glBuffLock`, `_glSwapData` (located in the library specifically for the LCD that you are using)

## **void glSetBrushType(int type);**

Sets the drawing method (or color) of pixels drawn by subsequent graphic calls.

### **PARAMETER**

**type** value can be one of the following macros.

**PIXBLACK** draws black pixels.

**PIXWHITE** draws white pixels.

**PIXXOR** draws old pixel XOR'ed with the new pixel.

### **RETURN VALUE**

None.

### **SEE ALSO**

`glGetBrushType`

```
int glGetBrushType(void);
```

Gets the current method (or color) of pixels drawn by subsequent graphic calls.

**RETURN VALUE**

The current brush type.

**SEE ALSO**

`glSetBrushType`

```
void glPlotDot(int x, int y);
```

Draws a single pixel in the LCD buffer, and on the LCD if the buffer is unlocked. If the coordinates are outside the LCD display area, the dot will not be plotted.

**PARAMETERS**

**x** is the *x* coordinate of the dot.

**y** is the *y* coordinate of the dot.

**RETURN VALUE**

None.

**SEE ALSO**

`glPlotline`, `glPlotPolygon`, `glPlotCircle`

```
void glPlotLine(int x0, int y0, int x1, int y1);
```

Draws a line in the LCD buffer, and on the LCD if the buffer is unlocked. Any portion of the line that is beyond the LCD display area will be clipped.

**PARAMETERS**

**x0** is the *x* coordinate of one endpoint of the line.

**y0** is the *y* coordinate of one endpoint of the line.

**x1** is the *x* coordinate of the other endpoint of the line.

**y1** is the *y* coordinate of the other endpoint of the line.

**RETURN VALUE**

None.

**SEE ALSO**

`glPlotDot`, `glPlotPolygon`, `glPlotCircle`

```
void glLeft1(int left, int top, int cols, int rows);
```

Scrolls byte-aligned window left one pixel, right column is filled by current pixel type (color).

#### PARAMETERS

**left** is the upper left corner of bitmap, must be evenly divisible by 8.

**top** is the left top corner of the bitmap.

**cols** is the number of columns in the window, must be evenly divisible by 8.

**rows** is the number of rows in the window.

#### RETURN VALUE

None.

#### SEE ALSO

`glHScroll`, `glRight1`

```
void glRight1(int left, int top, int cols, int rows);
```

Scrolls byte-aligned window right one pixel, left column is filled by current pixel type (color).

#### PARAMETERS

**left** is the upper left corner of bitmap, must be evenly divisible by 8.

**top** is the left top corner of the bitmap.

**cols** is the number of columns in the window, must be evenly divisible by 8.

**rows** is the number of rows in the window.

#### RETURN VALUE

None.

#### SEE ALSO

`glHScroll`, `glLeft1`

```
void glUp1(int left, int top, int cols, int rows);
```

Scrolls byte-aligned window up one pixel, bottom column is filled by current pixel type (color).

#### PARAMETERS

**left** is the upper left corner of bitmap, must be evenly divisible by 8.

**top** is the left top corner of the bitmap.

**cols** is the number of columns in the window, must be evenly divisible by 8.

**rows** is the number of rows in the window.

#### RETURN VALUE

None.

#### SEE ALSO

`glVScroll`, `glDown1`

```
void glDown1(int left, int top, int cols, int rows);
```

Scrolls byte-aligned window down one pixel, top column is filled by current pixel type (color).

#### PARAMETERS

**left** is the upper left corner of bitmap, must be evenly divisible by 8.

**top** is the left top corner of the bitmap.

**cols** is the number of columns in the window, must be evenly divisible by 8.

**rows** is the number of rows in the window.

#### RETURN VALUE

None.

#### SEE ALSO

`glVScroll`, `glUp1`

```
void glHScroll(int left, int top, int cols,  
               int rows, int nPix);
```

Scrolls right or left, within the defined window by  $x$  number of pixels. The opposite edge of the scrolled window will be filled in with white pixels. The window must be byte-aligned.

Parameters will be verified for the following:

1. The **left** and **cols** parameters will be verified that they are evenly divisible by 8. If not, they will be changed to a value that is a multiple of 8.
2. Parameters will be checked to verify that the scrolling area is valid. The minimum scrolling area is a width of 8 pixels and a height of one row.

#### PARAMETERS

**left** is the upper left corner of bitmap, must be evenly divisible by 8.

**top** is the left top corner of the bitmap.

**cols** is the number of columns in the window, must be evenly divisible by 8.

**rows** is the number of rows in the window.

**nPix** is the number of pixels to scroll within the defined window (a negative value will produce a scroll to the left).

#### RETURN VALUE

None.

#### SEE ALSO

`glVScroll`

```
void glVScroll(int left, int top, int cols,
               int rows, int nPix);
```

Scrolls up or down, within the defined window by *x* number of pixels. The opposite edge of the scrolled window will be filled in with white pixels. The window must be byte-aligned.

Parameters will be verified for the following:

1. The **left** and **cols** parameters will be verified that they are evenly divisible by 8. If not, they will be changed to a value that is a multiple of 8.
2. Parameters will be checked to verify that the scrolling area is valid. The minimum scrolling area is a width of 8 pixels and a height of one row.

#### PARAMETERS

**left** is the upper left corner of bitmap, must be evenly divisible by 8.

**top** is the left top corner of the bitmap.

**cols** is the number of columns in the window, must be evenly divisible by 8.

**rows** is the number of rows in the window.

**nPix** is the number of pixels to scroll within the defined window (a negative value will produce a scroll up).

#### RETURN VALUE

None.

#### SEE ALSO

`glHScroll`

```
void glXPutBitmap(int left, int top, int width,
                  int height, unsigned long bitmap);
```

Draws bitmap in the specified space. The data for the bitmap are stored in **xmem**. This function calls **glXPutFastmap** automatically if the bitmap is byte-aligned (the left edge and the width are each evenly divisible by 8).

Any portion of a bitmap image or character that is outside the LCD display area will be clipped.

#### PARAMETERS

**left** is the upper left corner of the bitmap.

**top** is the upper left corner of the bitmap.

**width** is the width of the bitmap.

**height** is the height of the bitmap.

**bitmap** is the address of the bitmap in **xmem**.

#### RETURN VALUE

None.

#### SEE ALSO

`glXPutFastmap`, `glPrintf`

```
void glXPutFastmap(int left, int top, int width,
    int height, unsigned long bitmap);
```

Draws bitmap in the specified space. The data for the bitmap are stored in **xmem**. This function is like **glXPutBitmap**, except that it is faster. The restriction is that the bitmap must be byte-aligned.

Any portion of a bitmap image or character that is outside the LCD display area will be clipped.

#### PARAMETERS

**left** is the upper left corner of the bitmap, must be evenly divisible by 8.

**top** is the upper left corner of the bitmap.

**width** is the width of the bitmap, must be evenly divisible by 8.

**height** is the height of the bitmap.

**bitmap** is the address of the bitmap in **xmem**.

#### RETURN VALUE

None.

#### SEE ALSO

**glXPutBitmap**, **glPrintf**

```
int TextWindowFrame(windowFrame *window,
    fontInfo *pFont, int x, int y, int winWidth,
    int winHeight)
```

Defines a text-only display window. This function provides a way to display characters within the text window using only character row and column coordinates. The text window feature provides end-of-line wrapping and clipping after the character in the last column and row is displayed.

**NOTE:** Execute the **TextWindowFrame** function before other **Text...** functions.

#### PARAMETERS

**\*window** is a window frame descriptor pointer.

**\*pFont** is a font descriptor pointer.

**x** is the x coordinate of where the text window frame is to start.

**y** is the y coordinate of where the text window frame is to start.

**winWidth** is the width of the text window frame.

**winHeight** is the height of the text window frame.

#### RETURN VALUE

0—window frame was successfully created.

-1—x coordinate + width has exceeded the display boundary.

-2—y coordinate + height has exceeded the display boundary.

```
void TextGotoXY(windowFrame *window, int col,  
int row);
```

Sets the cursor location on the display of where to display the next character. The display location is based on the height and width of the character to be displayed.

**NOTE:** Execute the **TextWindowFrame** function before using this function.

#### PARAMETERS

**\*window** is a pointer to a font descriptor.

**col** is a character column location.

**row** is a character row location.

#### RETURN VALUE

None.

#### SEE ALSO

**TextPutChar, TextPrintf, TextWindowFrame**

```
void TextCursorLocation(windowFrame *window,  
int *col, int *row);
```

Gets the current cursor location that was set by a Graphic **Text...** function.

**NOTE:** Execute the **TextWindowFrame** function before using this function.

#### PARAMETERS

**\*window** is a pointer to a font descriptor.

**\*col** is a pointer to cursor column variable.

**\*row** is a pointer to cursor row variable.

#### RETURN VALUE

Lower word = Cursor Row location

Upper word = Cursor Column location

#### SEE ALSO

**TextGotoXY, TextPrintf, TextWindowFrame, TextCursorLocation**

```
void TextPutChar(struct windowFrame *window, char ch);
```

Displays a character on the display where the cursor is currently pointing. If any portion of a bitmap character is outside the LCD display area, the character will not be displayed.

**NOTE:** Execute the **TextWindowFrame** function before using this function.

#### PARAMETERS

**\*window** is a pointer to a font descriptor.

**ch** is a character to be displayed on the LCD.

#### RETURN VALUE

None.

#### SEE ALSO

**TextGotoXY, TextPrintf, TextWindowFrame, TextCursorLocation**



```
void TextPrintf(struct windowFrame *window,  
char *fmt, ...);
```

Prints a formatted string (much like **printf**) on the LCD screen. Only printable characters in the font set are printed, also escape sequences, `\r` and `\n` are recognized. All other escape sequences will be skipped over; for example, `\b` and `\t` will print if they exist in the font set, but will not have any effect as control characters.

The text window feature provides end-of-line wrapping and clipping after the character in the last column and row is displayed.

**NOTE:** Execute the **TextWindowFrame** function before using this function.

#### PARAMETERS

**\*window** is a pointer to a font descriptor.

**\*fmt** is a formatted string.

**...** are formatted string conversion parameter(s).

#### EXAMPLE

```
TextPrintf(&TextWindow, "Test %d\n", count);
```

#### RETURN VALUE

None.

#### SEE ALSO

**TextGotoXY**, **TextPutChar**, **TextWindowFrame**, **TextCursorPosition**

## D.6 Keypad

The functions used to control the keypad are contained in the **KEYPAD7.LIB** library located in the Dynamic C **KEYPADS** library directory.

```
void keyInit(void);
```

Initializes keypad process

### RETURN VALUE

None.

### SEE ALSO

**brdInit**

```
void keyConfig(char cRaw, char cPress,  
               char cRelease, char cCntHold, char cSpdLo,  
               char cCntLo, char cSpdHi);
```

Assigns each key with key press and release codes, and hold and repeat ticks for auto repeat and debouncing.

### PARAMETERS

**cRaw** is a raw key code index.

1x7 keypad matrix with raw key code index assignments (in brackets):

[0]	[1]	[2]	[3]
[4]	[5]	[6]	

### User Keypad Interface

**cPress** is a key press code

An 8-bit value is returned when a key is pressed.

0 = Unused.

See **keypadDef ( )** for default press codes.

**cRelease** is a key release code.

An 8-bit value is returned when a key is pressed.

0 = Unused.

**cCntHold** is a hold tick.

How long to hold before repeating.

0 = No Repeat.

**cSpdLo** is a low-speed repeat tick.

How many times to repeat.

0 = None.

**cCntLo** is a low-speed hold tick.

How long to hold before going to high-speed repeat.

0 = Slow Only.

`cSpdHi` is a high-speed repeat tick.

How many times to repeat after low speed repeat.

0 = None.

#### RETURN VALUE

None.

#### SEE ALSO

`keyProcess`, `keyGet`, `keypadDef`

```
void keyProcess(void);
```

Scans and processes keypad data for key assignment, debouncing, press and release, and repeat.

**NOTE:** This function is also able to process an  $8 \times 8$  matrix keypad.

#### RETURN VALUE

None

#### SEE ALSO

`keyConfig`, `keyGet`, `keypadDef`

```
char keyGet(void);
```

Get next keypress

#### RETURN VALUE

The next keypress, or 0 if none

#### SEE ALSO

`keyConfig`, `keyProcess`, `keypadDef`

```
int keyUnget(char cKey);
```

Push keypress on top of input queue

#### PARAMETER

`cKey`

#### RETURN VALUE

None.

#### SEE ALSO

`keyGet`

## void keypadDef();

Configures the physical layout of the keypad with the desired ASCII return key codes.

Keypad physical mapping  $1 \times 7$

0	4	1	5	2	6	3
['L']		['U']		['D']		['R']
['-']		['+']		['E']		

where

'E' represents the ENTER key

'D' represents Down Scroll

'U' represents Up Scroll

'R' represents Right Scroll

'L' represents Left Scroll

**Example:** Do the following for the above physical vs. ASCII return key codes.

```
keyConfig ( 3, 'R', 0, 0, 0, 0, 0 );
keyConfig ( 6, 'E', 0, 0, 0, 0, 0 );
keyConfig ( 2, 'D', 0, 0, 0, 0, 0 );
keyConfig ( 4, '-', 0, 0, 0, 0, 0 );
keyConfig ( 1, 'U', 0, 0, 0, 0, 0 );
keyConfig ( 5, '+', 0, 0, 0, 0, 0 );
keyConfig ( 0, 'L', 0, 0, 0, 0, 0 );
```

Characters are returned upon keypress with no repeat.

### RETURN VALUE

None.

### SEE ALSO

keyConfig, keyGet, keyProcess

## void keyScan(char \*pcKeys);

Writes "1" to each row and reads the value. The position of a keypress is indicated by a zero value in a bit position.

### PARAMETER

**\*pcKeys** is the address of the value read.

### RETURN VALUE

None.

### SEE ALSO

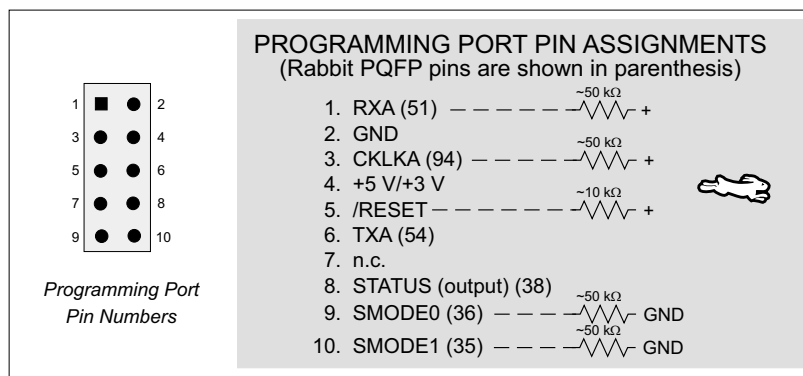
keyConfig, keyGet, keypadDef, keyProcess



## APPENDIX E. PROGRAMMING CABLE

Appendix E 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 J1 on the OP6800 module) 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 a OP6800 system while it is running.

The programming port, which is shown in Figure E-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 E-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.

# INDEX

## B

battery connections ..... 54  
buzzer ..... 61

## C

chip select circuit ..... 56  
connections  
    Ethernet cable ..... 33  
    programming cable ..... 9

## D

Demonstration Board  
    mounting and installation .. 63  
    pinout ..... 62  
    prototyping area ..... 62  
    wire assembly ..... 2  
demonstration program ..... 8  
digital inputs ..... 15  
    remote keypad operation ... 15  
    switching threshold ..... 15  
digital outputs ..... 16  
dimensions  
    Demonstration Board ..... 58  
    LCD/keypad template ..... 22  
    OP6800 ..... 44  
Dynamic C Premier ..... 3, 25  
    changing programming baud  
        rate in BIOS ..... 10  
    debugging features ..... 25  
    installation ..... 10  
    starting ..... 10

## E

Ethernet cables ..... 33  
Ethernet connections ..... 33  
    steps ..... 33  
Ethernet port ..... 20  
    handling EMI and noise .... 20  
    pinout ..... 20  
exclusion zone ..... 45

## F

features ..... 1  
flash memory  
    lifetime write cycles ..... 25  
flash memory bank select ..... 21  
font and bitmap converter ..... 32

## H

headers  
    JP1 ..... 18

## I

I/O address assignments ..... 51  
installation guidelines ..... 39  
introduction ..... 1  
IP addresses ..... 35  
    how to set ..... 35  
    how to set PC IP address ... 35

## J

jumper configurations ..... 48  
    Demonstration Board buzzer 61  
    JP1 (RS-485 bias and termina-  
        tion resistors) ..... 18, 48  
    jumper locations ..... 48

## K

keypad template ..... 22  
    removing and inserting label 23

## L

LCD/keypad module  
    keypad template ..... 22  
    removing and inserting keypad  
        label ..... 23

## M

memory ..... 21  
models ..... 2  
    OP6800 ..... 2  
    OP6810 ..... 2  
mounting and installation  
    Demonstration Board ..... 63  
    OP6800 ..... 40, 41

## O

OP6800  
    introduction ..... 1

## P

pin 1 locations ..... 44  
pinout  
    Demonstration Board ..... 62  
    Ethernet port ..... 20  
    OP6800 headers ..... 14  
    programming port ..... 90  
power distribution ..... 59  
power management ..... 53  
power supply ..... 2, 53, 59  
    backup battery circuit ..... 54  
    battery backup ..... 54  
    chip select circuit ..... 56  
    connections ..... 7  
    power distribution ..... 59  
    switching voltage regulator 53  
    VRAM switch ..... 55  
power-up  
    demonstration program ..... 8  
Program Mode ..... 26  
programming  
    flash vs. RAM ..... 25  
    programming cable ..... 2  
    programming port ..... 19  
programming cable ..... 2  
    connections ..... 9  
    DIAG connector ..... 90  
    switching between Program  
        Mode and Run Mode .... 26

programming port .....	19
pinout .....	90
used as diagnostic port .....	90

## R

Rabbit 2000	
parallel ports .....	49
remote keypad operation .....	15
reset .....	7
hardware .....	7
reset generator .....	55
RS-232 .....	17
RS-485 .....	17
RS-485 network .....	18
termination and bias resistors .....	18
Run Mode .....	26

## S

sample programs .....	29
BOARD_ID.C .....	29
Demonstration Board .....	11
BUZZER.C .....	29
KEYPAD.C .....	29
SWITCHES.C .....	29
digital I/O	
DIGIN.C .....	30
DIGOUT.C .....	30
how to set IP address .....	35
ICOMDEMO.C .....	8
LCD/keypad module .....	30
ALPHANUN.C .....	30
COFTERMA.C .....	30
DISPPONG.C .....	30
DKADEMO1.C .....	30
FUN.C .....	8, 30
KEYBASIC.C .....	31
KEYMENU.C .....	31
LED.C .....	31
SCROLLING.C .....	31
TEXT.C .....	31
LCD/keypad module (with TCP/IP)	
MBOXDEMO.C .....	37
TCP_RESPOND.C .....	37
TCPSEND.C .....	37
OP6800 features .....	11
PONG.C .....	11
power-up demonstration program .....	8

sample programs (continued)	
serial communication	
MASTER.C .....	30
PUTS.C .....	30
RELAYCHR.C .....	30
SLAVE.C .....	30
TCP/IP .....	31, 35
PINGME.C .....	36
SSI.C .....	36
TELNET.C .....	37
serial communication .....	17
programming port .....	19
RS-232 description .....	17
RS-485 description .....	17
RS-485 network .....	18
RS-485 termination and bias resistors .....	18
serial ports	
Ethernet port .....	20
setup .....	5
power supply connections ...	7
programming cable connections .....	9
remove RabbitCore module .....	9
software .....	3
board initialization .....	66
brdInit .....	66
digital I/O	
digIn .....	67
digOut .....	67
keypad	
keyConfig .....	86
keyGet .....	87
keyInit .....	86
keypadDef .....	88
keyProcess .....	87
keyScan .....	88
keyUnget .....	87
LCD display	
glBackLight .....	70
glBlankScreen .....	71
glBlock .....	72
glBuffLock .....	78
glBuffUnlock .....	78
glDispOnOff .....	70
glDown1 .....	81
glFillCircle .....	74
glFillPolygon .....	74
glFillScreen .....	71
glFillVPolygon .....	73
glFontCharAddr .....	75
glGetBrushType .....	79

## software

LCD display (continued)	
glGetPfStep .....	76
glHScroll .....	81
glInit .....	70
glLeft1 .....	80
glPlotCircle .....	74
glPlotDot .....	79
glPlotLine .....	79
glPlotPolygon .....	73
glPlotVPolygon .....	72
glPrintf .....	77
glPutChar .....	77
glPutFont .....	76
glRight1 .....	80
glSetBrushType .....	78
glSetContrast .....	71
glSetPfStep .....	76
glSwap .....	78
glUp1 .....	80
glVScroll .....	82
glXFontInit .....	32, 75
glXPutBitmap .....	32, 82
glXPutFastmap .....	83
TextCursorLocation .....	84
TextGotoXY .....	84
TextPrintf .....	85
TextPutChar .....	84
TextWindowFrame .....	83
LCD/keypad module	
ledOut .....	69
libraries .....	27
ARP.LIB .....	28
BOOTP.LIB .....	28
BSDNAME.LIB .....	28
DCRTCP.LIB .....	28
DNS.LIB .....	28
FTP_CLIENT.LIB .....	28
FTP_SERVER.LIB .....	28
HTTP.LIB .....	28
ICMP.LIB .....	28
IP.LIB .....	28
MD5.LIB .....	28
NET.LIB .....	28
PACKET.LIB .....	68
PKTDRV.LIB .....	28
POP3.LIB .....	28
REALTEK.LIB .....	28
RS232.LIB .....	68
SMTP.LIB .....	28
TCP.LIB .....	28
TCP/IP .....	27



software	
libraries (continued)	
UDP.LIB .....	28
VSERIAL.LIB .....	28
ZSERVER.LIB .....	28
sample programs .....	29
PONG.C .....	11
serial communication	
flow control .....	68
ser485Rx .....	68
ser485Tx .....	68
serCflowcontrolOff .....	68
serCflowcontrolOn .....	68
serMode .....	68
specifications	
Demonstration Board	
dimensions .....	58
electrical .....	58
mechanical .....	58
temperature .....	58
OP6800	
dimensions .....	44
electrical .....	46
exclusion zone .....	45
mechanical .....	46
temperature .....	46
subsystems .....	13

## T

TCP/IP connections .....	33
10Base-T Ethernet card ....	33
additional resources .....	38
Ethernet hub .....	33
steps .....	33
Tool Kit .....	2
AC adapter .....	2
DC power supply .....	2
programming cable .....	2
User's Manual .....	2
wire assembly .....	2





# **SCHEMATICS**

**090-0134 OP6800 Schematic**

**090-0120 RCM2200 RabbitCore Schematic**

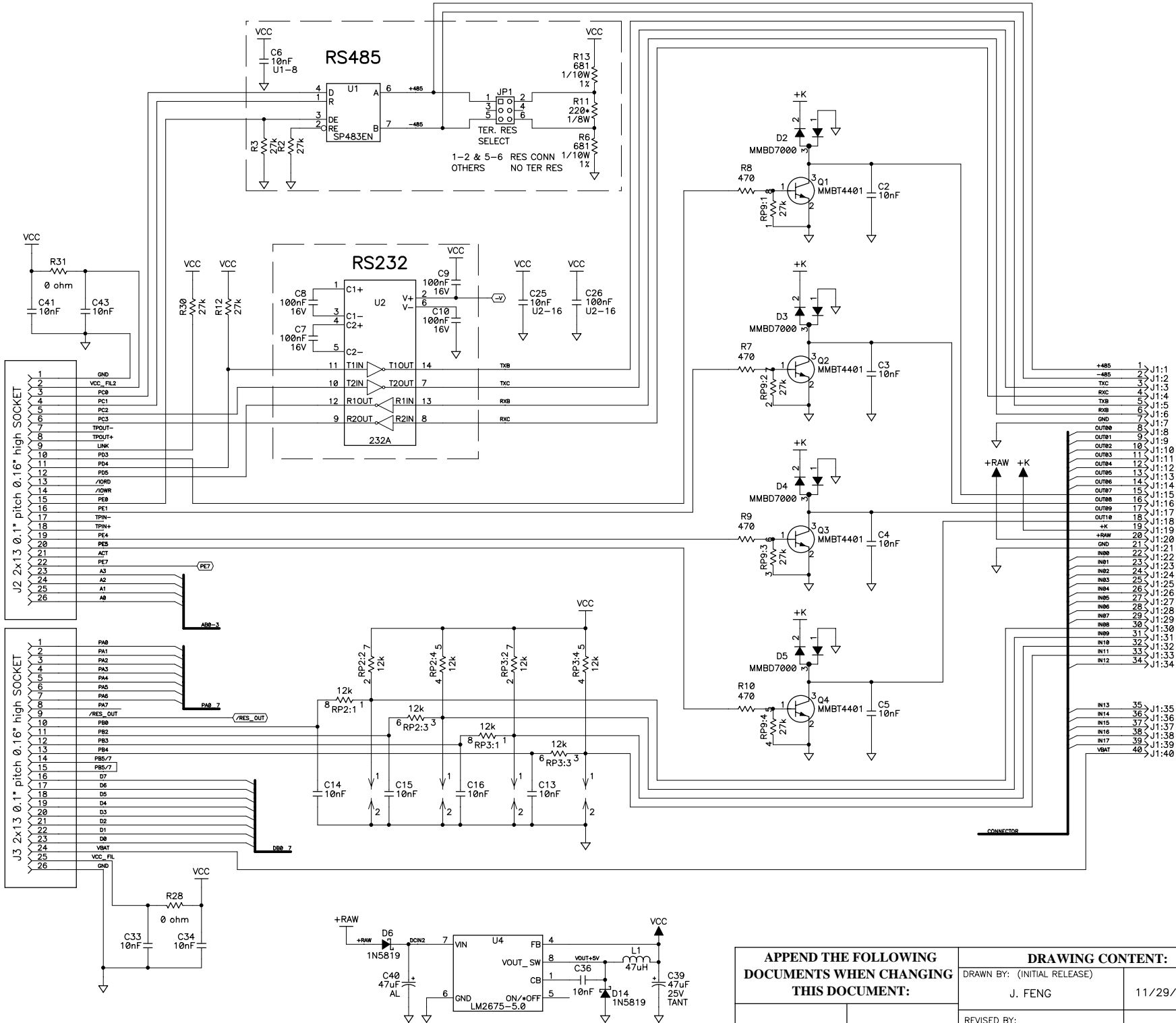
**090-0119 RCM2300 RabbitCore Schematic**

**090-0140 OP6800 Demonstration 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	E11764	INITIAL RELEASE. FOLLOWS REV C PCB.	JF	12DEC01	KLS	12DEC01
B	E11807	CHANGE VALUE OF RP2 & RP3; CORRECT J3 REFERENCE.				

To Micro-core

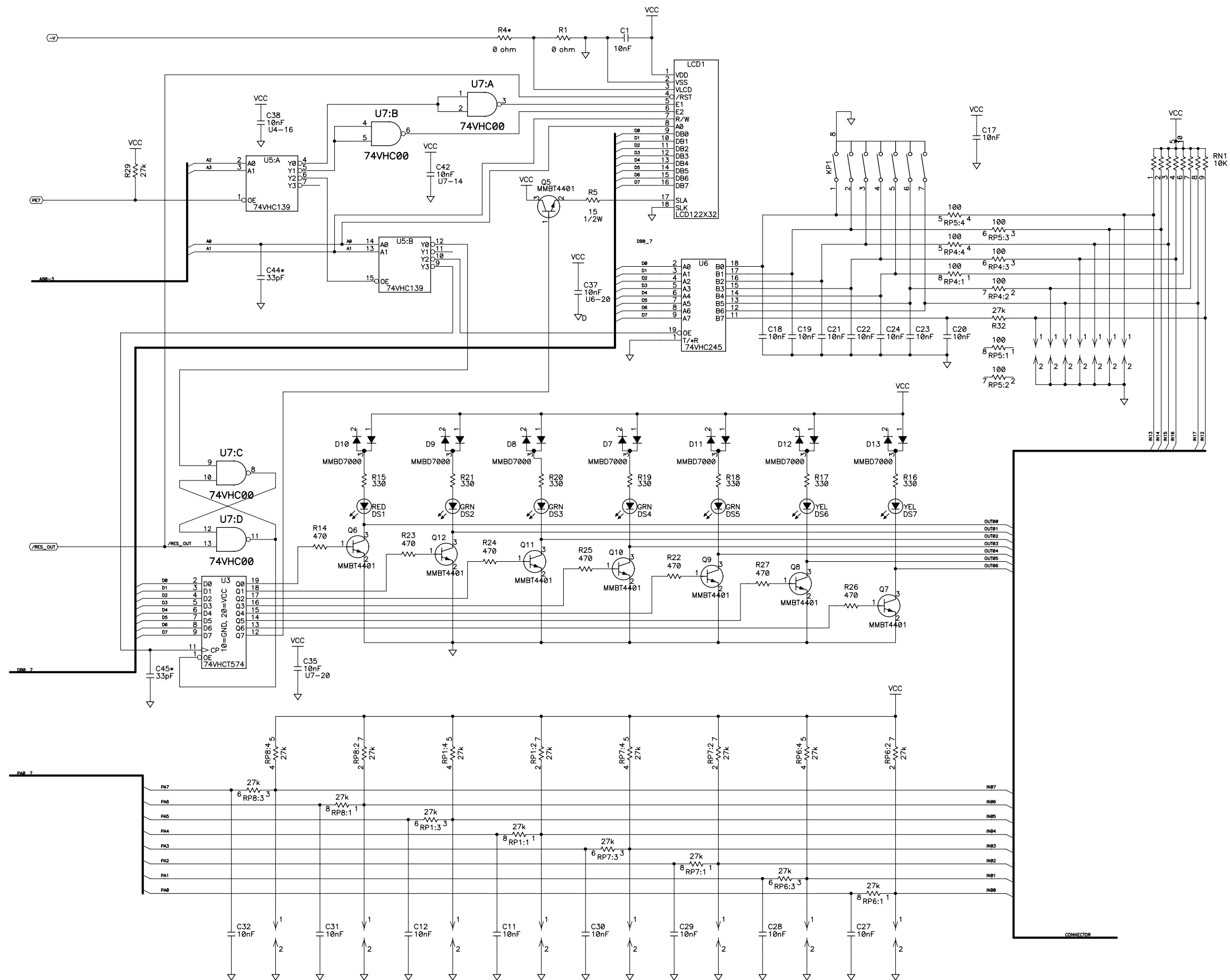


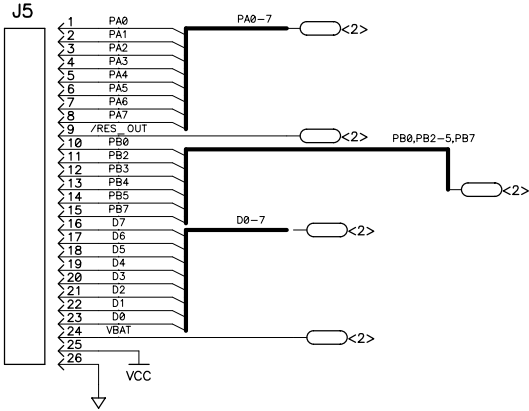
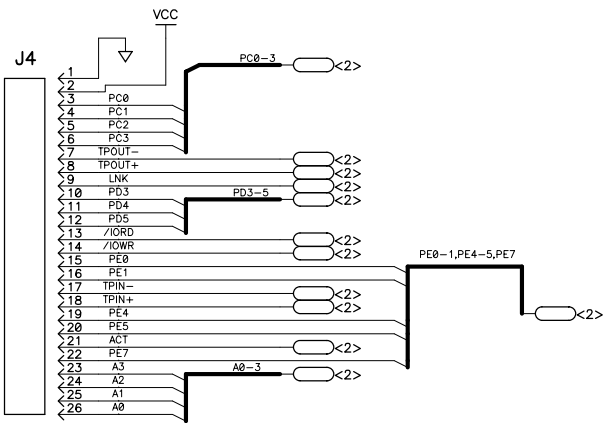
POWER TABLE

REF DES	DEVICE	DEVICE VOLTAGE		FILTER CAP
		VCC	GND	
U1	SP483EN	8	5	C6
U2	232A	16	15	C25, C26
U3	74VHC574	20	10	C35
U5	74VHC139	16	8	C38
U6	74VHC245	20	10	C37
U7	74VHC00	14	7	C42

- NOTES: UNLESS OTHERWISE SPECIFIED:
- ALL RESISTOR VALUES ARE IN OHMS, 1/16W, 5%.
  - ALL CAPACITORS ARE 50VDC OR HIGHER.
  - THE ORIGIN SOURCE OF A VOLTAGE IS REPRESENTED BY (VCC), AND ALL REFERENCES TO THAT VOLTAGE ARE REPRESENTED BY (VCC).
  - COMPONENT VALUES SHOWN WITH AN ASTERISK (\*) FOLLOWING THE VALUE, MAY HAVE DIFFERENT VALUES, OR MAY NOT BE STUFFED.

APPEND THE FOLLOWING DOCUMENTS WHEN CHANGING THIS DOCUMENT:		DRAWING CONTENT:		TITLE		Z-WORLD 2900 SPAFFORD ST. DAVIS, CA 95616 530 - 757-4616	
		DRAWN BY: (INITIAL RELEASE)					
		J. FENG	11/29/01	SCHEMATIC DIAGRAM OP6800 SERIES		C	
		REVISED BY:					
		E. PEAK	14 JAN 02	APPROVALS: INITIAL RELEASE		090-0134	
		PROJECT ENGINEER:					
		J. FENG	12 DEC 01	ENGINEERING MANAGER:		DWG NO.	
		D. MATTHEWS	12 DEC 01				
		SIGNATURES	DATE	SCALE	NONE	RELEASE DATE	12 DEC 01
				SHEET	1	OF	2





STUFFING TABLE

	CIRCUIT	PART	RCM2200	RCM2210	RCM2250
POWER TO VRAM SWITCH	WITH BATTERY BACKUP CIRCUITRY	R33	NOT INSTALLED	NOT INSTALLED	NOT INSTALLED
CS CONTROL SWITCH	WITH BATTERY BACKUP CIRCUITRY	R27	NOT INSTALLED	NOT INSTALLED	NOT INSTALLED
SRAM	MAIN	U5	128K SRAM	128K SRAM	512K SRAM
	SRAM SELECT	JP7	ZERO OHM ACROSS PINS 1-2	ZERO OHM ACROSS PINS 1-2	ZERO OHM ACROSS PINS 2-3
FLASH	FIRST	U3	256K FLASH	256K FLASH	256K FLASH
	FLASH MEMORY BANK SELECT	JP3	ZERO ohm ACROSS PINS 1-2	ZERO ohm ACROSS PINS 1-2	ZERO ohm ACROSS PINS 1-2
	FLASH TYPE	JP4	ZERO ohm ACROSS PINS 1-2	ZERO ohm ACROSS PINS 1-2	ZERO ohm ACROSS PINS 1-2
	CAPACITOR	C10	INSTALLED	INSTALLED	INSTALLED
	SECOND	U8	NOT INSTALLED	NOT INSTALLED	256K FLASH
	FLASH MEMORY BANK SELECT	JP2	NOT INSTALLED	NOT INSTALLED	ZERO ohm ACROSS PINS 1-2
	FLASH TYPE	JP1	NOT INSTALLED	NOT INSTALLED	ZERO ohm ACROSS PINS 1-2
	CAPACITOR	C30	NOT INSTALLED	NOT INSTALLED	INSTALLED
	THIRD	U7	NOT INSTALLED	NOT INSTALLED	NOT INSTALLED
	FLASH MEMORY BANK SELECT	JP5	NOT INSTALLED	NOT INSTALLED	NOT INSTALLED
	FLASH TYPE	JP6	NOT INSTALLED	NOT INSTALLED	NOT INSTALLED
	CAPACITOR	C29	NOT INSTALLED	NOT INSTALLED	NOT INSTALLED
ETHERNET	RJ-45 CONNECTOR WITH BUILT IN MAGNETICS	J2	INSTALLED	NOT INSTALLED	INSTALLED
	FILTER CAPACITORS	C18	INSTALLED	NOT INSTALLED	INSTALLED
		C19	INSTALLED	NOT INSTALLED	INSTALLED
		C23	INSTALLED	NOT INSTALLED	INSTALLED
		C24	INSTALLED	NOT INSTALLED	INSTALLED
	LEDS	DS1	INSTALLED	NOT INSTALLED	INSTALLED
		DS2	INSTALLED	NOT INSTALLED	INSTALLED
		R34	INSTALLED	NOT INSTALLED	INSTALLED
		R35	INSTALLED	NOT INSTALLED	INSTALLED
BATTERY	ON BOARD BATTERY	BT1	NOT INSTALLED	NOT INSTALLED	NOT INSTALLED

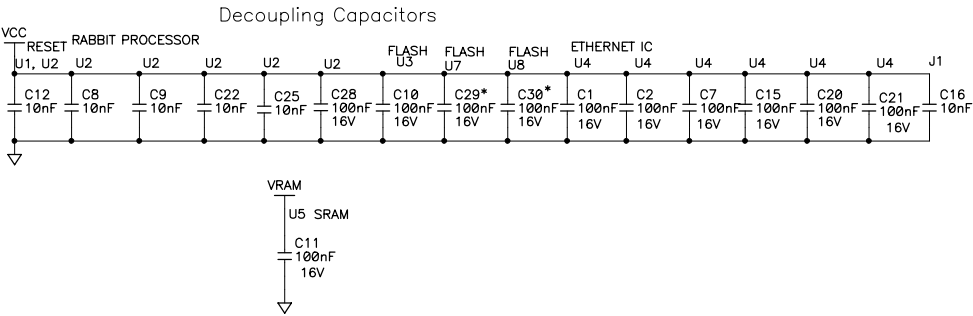
REVISION HISTORY

REVISION APPROVAL

REV	ECO	DESCRIPTION OF CHANGE	PROJECT ENGINEER	APPROVAL DATE	DOCUMENT CONTROL	APPROVAL DATE
A	E11399	INITIAL RELEASE	RJH	3/22/01	KIS	3/22/01
B	E11646	ADD CAP TO THE LEFT OF R17 (PAGE 2). ADD REFERENCE TO NEW RCM2210.	RJH	9/21/01	KIS	8/29/01
C	E11570	REMOVED R12, ADDED R42,R43	RJH	9/24/01	KIS	9/21/01
D	E11787	CORRECT STUFFING TABLE SPEC FOR RCM2210 AT U5.				

TABLE A


REF DES	DEVICE	DEVICE VOLTAGE INFORMATION			DEVICE: FILTER CAP REF DES(s)
		GND	VCC	VRAM	
U1	ETC811L	1	4		C12
U2	RABBIT 2000	2,27,39 52,77,89	3,28,53, 78,92		C8,C9,C12,C22,C25,C28
U3	FLASH	24	8		C10
U8	FLASH	24	8		C30
U7	FLASH	24	8		C29
U4	RTL8019AS	14,28,44 52,83,86	6,17,47 57,70,89		C1,C2,C7,C15,C20,C21
U5	SRAM 128K X 8	16		32	C11

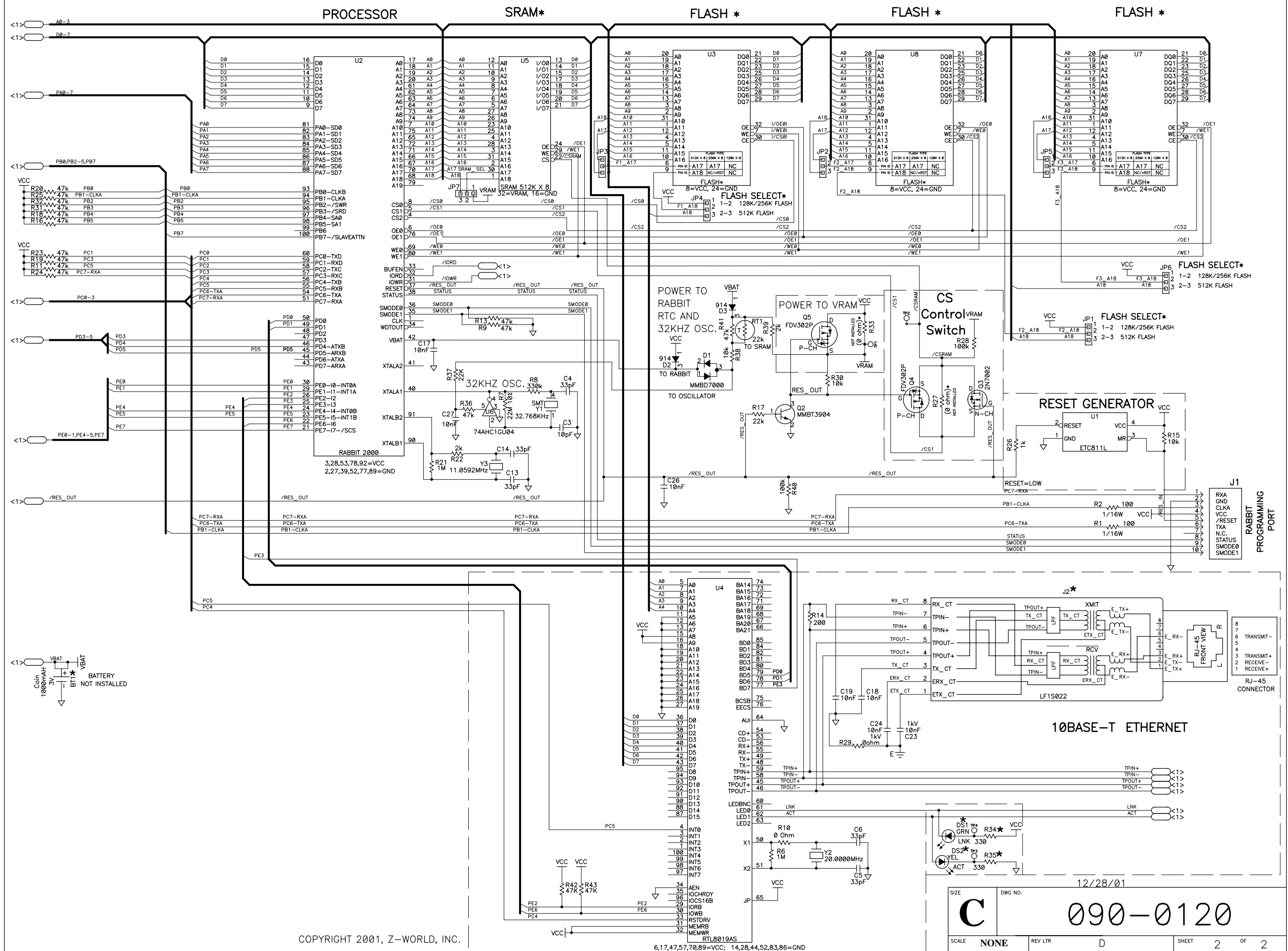


- NOTES: UNLESS OTHERWISE SPECIFIED;
1. ALL RESISTOR VALUES ARE IN OHMS, 1/16W, 5%
  2. ALL CAPACITORS ARE 50VDC OR HIGHER.
  3. THE ORIGATION SOURCE OF A VOLTAGE IS REPRESENTED BY (  $V_{CC}$  ), AND ALL REFERENCES TO THAT VOLTAGE ARE REPRESENTED BY (  $V_{CC}$  ).
  4. R27, R33, & BT1 NOT NORMALLY STUFFED.
  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.

Copyright 2001, Z-World, Inc.

12/28/01

APPEND THE FOLLOWING DOCUMENTS WHEN CHANGING THIS DOCUMENT:		DRAWING CONTENT:		TITLE SCHEMATIC DIAGRAM MICRO ETHERNET RABBITCORES RCM2200 SERIES		 2900 SPAFFORD ST. DAVIS, CA 95616 530 - 757-4616	
		DRAWN BY: (INITIAL RELEASE)					
	RJH	3/15/01					
	REVISED BY: K.SCHALLER	12/28/01					
		APPROVALS: INITIAL RELEASE		SIZE <b>C</b>		DWG NO.  <b>090-0120</b>	
	PROJECT ENGINEER: RICHARD HESS	3/22/01					
	ENGINEERING MANAGER: R.MATTHEWS	3/22/01					
		SIGNATURES	DATE	SCALE NONE	RELEASE DATE	SHEET 1	OF 2



REVISION HISTORY			REVISION APPROVAL			
REV	ECO	DESCRIPTION OF CHANGE	PROJECT ENGINEER	APPROVAL DATE	DOCUMENT CONTROL	APPROVAL DATE
A	E11400	INITIAL RELEASE	RJH	6/04/01	KIS	6/04/01
B	E11691	NEW BATTERY BACKUP CIRCUITRY, BATTERY, J2, AND J3	RJH	6/04/01	KIS	6/04/01
C	E11787	REMOVE C16 REFERENCE TO DECOUPLING U3				

STUFFING TABLE


	CIRCUIT	PART	RCM2300
POWER TO VRAM SWITCH	WITH BATTERY BACKUP CIRCUITRY	R33	NOT INSTALLED
CS CONTROL SWITCH	WITH BATTERY BACKUP CIRCUITRY	R27	NOT INSTALLED
FLASH	MAIN	U3	256K FLASH
	FLASH SELECT	JP1	ZERO ohm ACROSS PINS 1–2
	FLASH TYPE	JP2	ZERO ohm ACROSS PINS 1–2
LED	RED LED CIRCUIT	R29	NOT INSTALLED
		DS1	NOT INSTALLED
BATTERY	ON BOARD BATTERY	BT1	NOT INSTALLED
EXTRA CONN.	2MM THROUGH HOLE CONNECTORS	J2	NOT INSTALLED
		J3	NOT INSTALLED

TABLE A

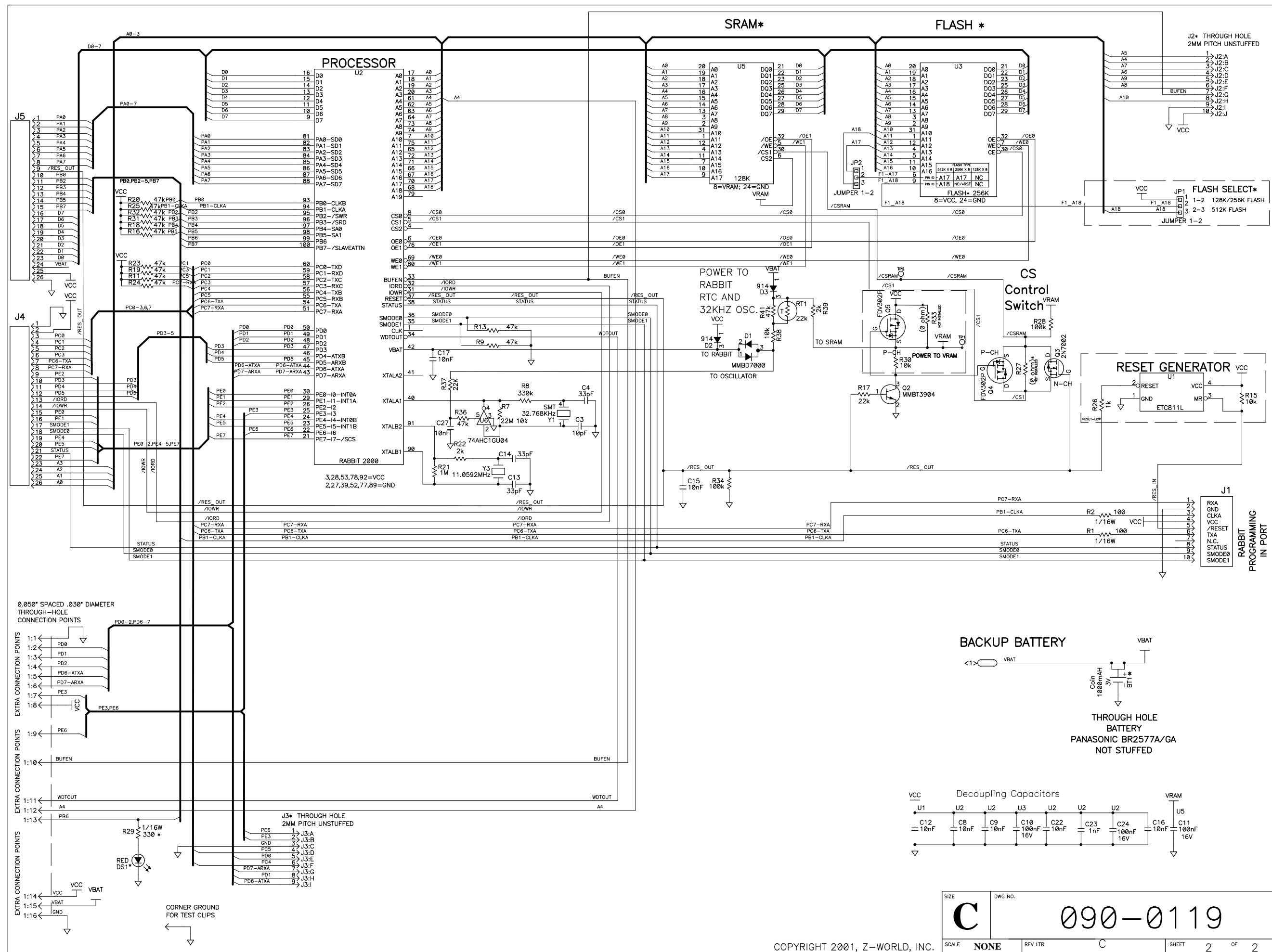
REF DES	DEVICE	DEVICE VOLTAGE INFORMATION				DEVICE: FILTER CAP REF DES(s)
		GND	VCC	VRAM	NO CONNECTS	
U1	ETC811L					C12
U2	RABBIT 2000	2,27,39 52,77,89	3,28,53, 78,92			C8,C9,C10,22,23
U3	FLASH	24	8			C16
U5	SRAM 128K X 8	24		8		C11

- NOTES: UNLESS OTHERWISE SPECIFIED;
- ALL RESISTOR VALUES ARE IN OHMS, 1/16W, 5%
  - ALL CAPACITORS ARE 50VDC OR HIGHER.
  - THE ORIGINATION SOURCE OF A VOLTAGE IS REPRESENTED BY (  $\overset{VCC}{\uparrow}$  ), AND ALL REFERENCES TO THAT VOLTAGE ARE REPRESENTED BY (  $\overset{VCC}{\downarrow}$  ).
  - R27, R33, J2, J3, BT1, DS1, & R29 NOT NORMALLY STUFFED.
  - 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.
  - JP1 AND JP2 ARE JUMPERED POSITION 1 TO POSITION 2 BY FACTORY DEFAULT.

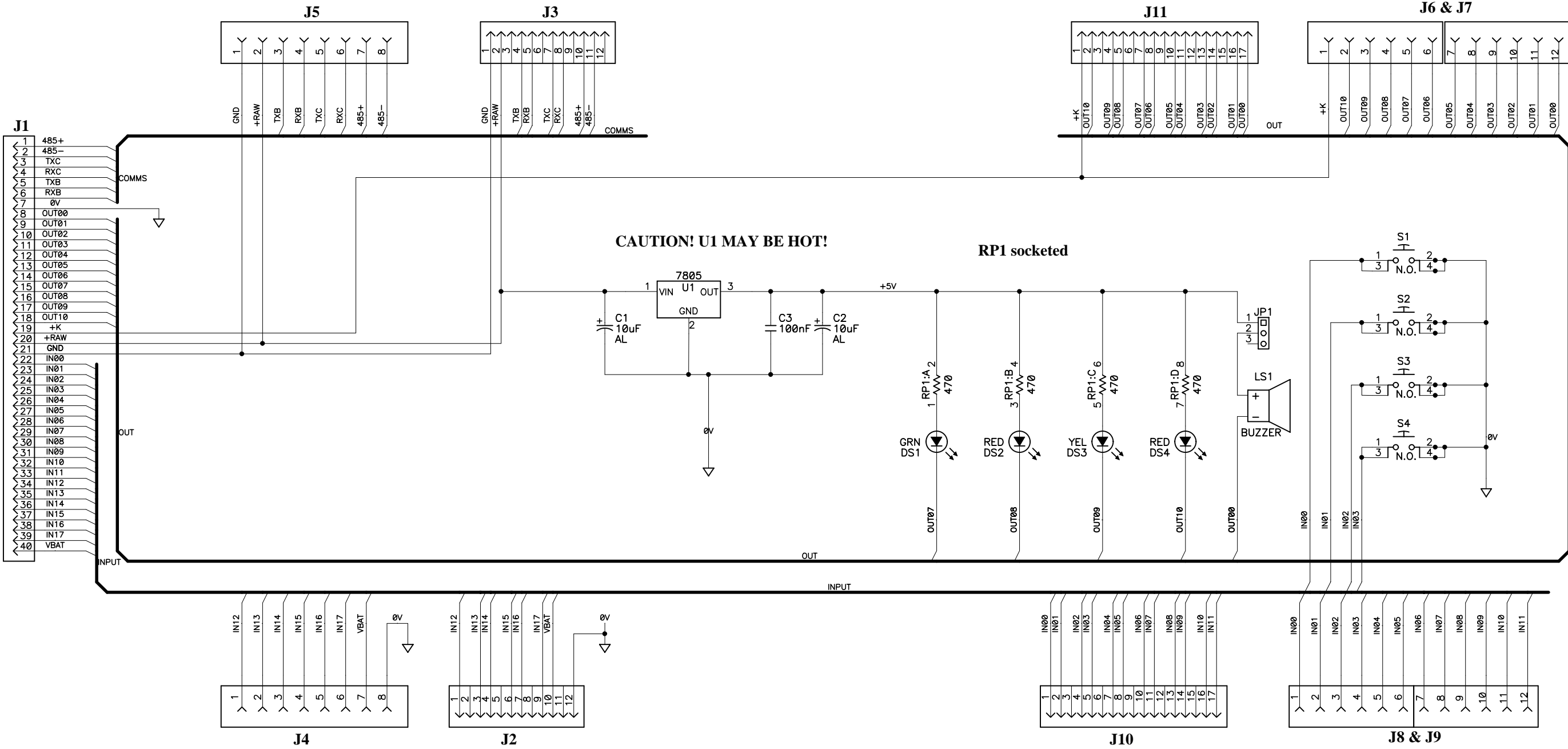
Copyright 2001, Z–World, Inc.


APPEND THE FOLLOWING DOCUMENTS WHEN CHANGING THIS DOCUMENT:		DRAWING CONTENT:		TITLE		 2900 SPAFFORD ST. DAVIS, CA 95616 530 - 757-4616	
		DRAWN BY: (INITIAL RELEASE)					
		RJH	3/22/01	SCHEMATIC DIAGRAM RCM2300		C090–0119	
		REVISED BY:					
		K.SCHALLER	12/27/01				
		APPROVALS: INITIAL RELEASE		SIZE		DWG NO.	
		PROJECT ENGINEER:					
		ENGINEERING MANAGER:		SCALE		RELEASE DATE	
		SIGNATURES	DATE	NONE		SHEET 1 OF 2	



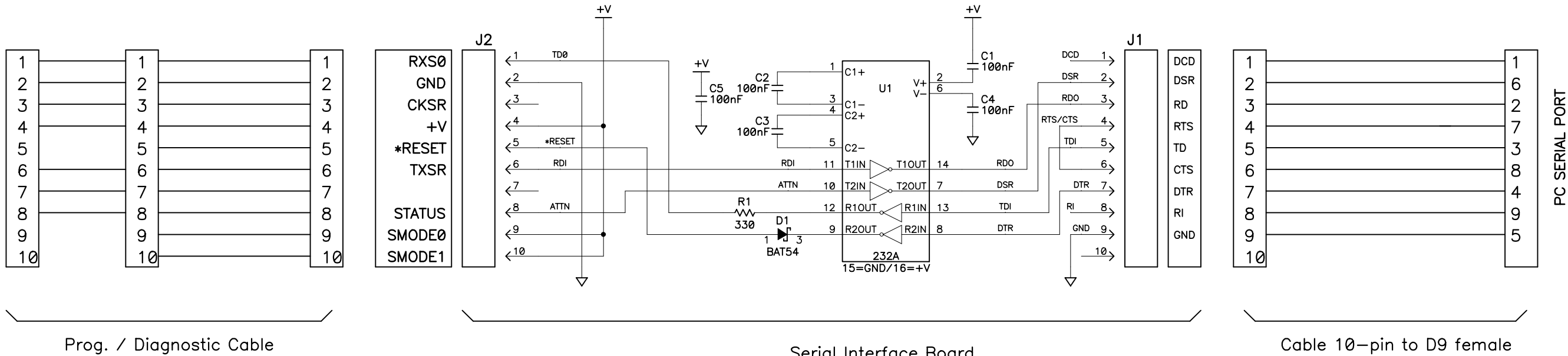


REVISION HISTORY			REVISION APPROVAL			
REV	ECO	DESCRIPTION	PROJECT ENGINEER	APPROVAL DATE	DOCUMENT CONTROL	APPROVAL DATE
A	E11770	INITIAL RELEASE				




APPEND THE FOLLOWING DOCUMENTS WHEN CHANGING THIS DOCUMENT:		DRAWING CONTENT:		TITLE  SCHEMATIC DIAGRAM OP6800  PROTO BOARD		  2900 SPAFFORD ST. DAVIS, CA 95616 530 - 757 - 4616				
		DRAWN BY: (INITIAL RELEASE) Eric Peak	11 Dec 2001							
		REVISED BY:		SIZE  <b>B</b>				DWG NO.  090-0140		
		APPROVALS: INITIAL RELEASE								
		PROJECT ENGINEER:		SCALE  NONE				RELEASE DATE		
		ENGINEERING MANAGER:								
		SIGNATURES		DATE		SHEET		1	OF	1

REVISION HISTORY				REVISION APPROVAL			
REV	ECO	DESCRIPTION		PROJECT ENGINEER	APPROVAL DATE	DOCUMENT CONTROL	APPROVAL DATE
A	E11523	INITIAL RELEASE OF SCHEMATIC		EP	5/14/01	KIS	5/14/01
B	E11691	CORRECT DE9 PINOUT					



NOTES: UNLESS OTHERWISE SPECIFIED;  
1. ALL RESISTOR VALUES ARE IN OHMS, 1/10W, 5%  
2. ALL CAPACITORS ARE 50VDC OR HIGHER.  
3. THE ORIGATION 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:		TITLE		<div> <b>2900 SPAFFORD ST. DAVIS, CA 95616 530 - 757 - 4616</b></div>			
		DRAWN BY: (INITIAL RELEASE)		SCHEMATIC DIAGRAM RABBIT PROG. CABLE					
		REVISD BY:							
		APPROVALS: INITIAL RELEASE							
		PROJECT ENGINEER:		SIZE <b>B</b>	DWG NO. <b>090-0128</b>				
		ENGINEERING MANAGER:							
		SIGNATURES		DATE		SCALE <b>NONE</b>	RELEASE DATE	SHEET <b>1</b>	OF <b>1</b>

