# Smartcat (BL2100)

C-Programmable Single-Board Computer with Ethernet and Operator Interface

## User's Manual

019–0103 • 011115–A

# Smartcat (BL2100) User's Manual

Part Number 019-0103 • 011115–A • Printed in U.S.A.



## 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) 753-5141

www.zworld.com

# TABLE OF CONTENTS

# 1. INTRODUCTION

The BL2100 is a high-performance, C-programmable single-board computer that offers built-in digital and analog I/O combined with Ethernet connectivity in a compact form factor. A Rabbit 2000™ microprocessor operating at 22.1 MHz provides fast data processing. An optional plastic enclosure and LCD/keypad module are available, and may be wall-mounted.

## 1.1 BL2100 Description

The BL2100 is an advanced single-board computer that incorporates the powerful Rabbit 2000 microprocessor, flash memory, static RAM, digital I/O ports, A/D converter inputs, D/A converter outputs, RS-232/RS-485 serial ports, and a 10Base-T Ethernet port.

## 1.2 BL2100 Features

- 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.
- 40 digital I/O: 24 protected digital inputs and 16 high-current digital outputs provide sinking and sourcing outputs.
- 15 analog channels: eleven 12-bit A/D converter inputs, four 12-bit D/A converter 0–10 V outputs (selected models).
- One RJ-45 Ethernet port compliant with IEEE 802.3 standard for 10Base-T Ethernet protocol (selected models).
- Two Ethernet status LEDs (selected models).
- Four serial ports (2 RS-232 or 1 RS-232 with RTS/CTS, 1 RS-485, and 1 CMOS-compatible programming port).
- Battery-backed real-time clock.
- Watchdog supervisor.
- Optional backlit 122 × 32 graphic display/keypad module.
- Remote program downloading and debugging capability via RabbitLink.

Four BL2100 models are available. Their standard features are summarized in Table 1.

***Table 1. BL2100 Models***

| Feature | BL2100 | BL2110 | BL2120 | BL2130 |
|---|---|---|---|---|
| Microprocessor | Rabbit 2000 running at 22.1 MHz | | | |
| Static RAM | 128K | | | |
| Flash Memory | 256K | | | |
| RJ-45 Ethernet Connector, Filter Capacitors, and LEDs | Yes | | No | |
| A/D Converter Inputs (-10 V to + 10 V) | Yes | No | Yes | No |
| D/A Converter Outputs (0 V to +10 V) | Yes | No | Yes | No |
| RabbitCore Module Used | RCM2200 | | RCM2300 | |

Appendix A provides detailed specifications.

## 1.3 Optional Add-Ons

- Plastic enclosure (can be wall-mounted or panel-mounted) with LCD/keypad module that comprises a 122 × 32 LCD graphic display, 7-key keypad, and seven LEDs. The plastic enclosure consists of a base and a cover for an assembly made up of the BL2100 with the LCD/keypad module plugged in.
- Plastic enclosure base.
- LCD/keypad module.

One enclosure base is included with the Tool Kit.

Further details on these add-ons are provided in Appendix C and in Appendix D.

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.

## 1.4 Development and Evaluation Tools

### 1.4.1 Tool Kit

A Tool Kit contains the hardware essentials you will need to create and use your own BL2100 single-board computer. The items in the Tool Kit and their use are as follows.

- ***BL2100 User's Manual*** with schematics (this document).
- Programming cable, used to connect your PC serial port to the BL2100.
- AC adapter, used to power the BL2100. An AC adapter is supplied with tool kits sold in the North American market. If you are using another power supply, it must provide 9 to 36 V DC (13 to 36 V DC if you intend to use the full range of the D/A converter outputs).
- Demonstration Board with pushbutton switches and LEDs. The Demonstration Board can be hooked up to the BL2100 to demonstrate the I/O and the TCP/IP capabilities of the BL2100.
- Wire assembly to connect Demonstration Board to BL2100.
- Plastic enclosure base with mounting screws.
- Screwdriver.
- ***Rabbit 2000 Processor Easy Reference*** poster.
- Registration card.

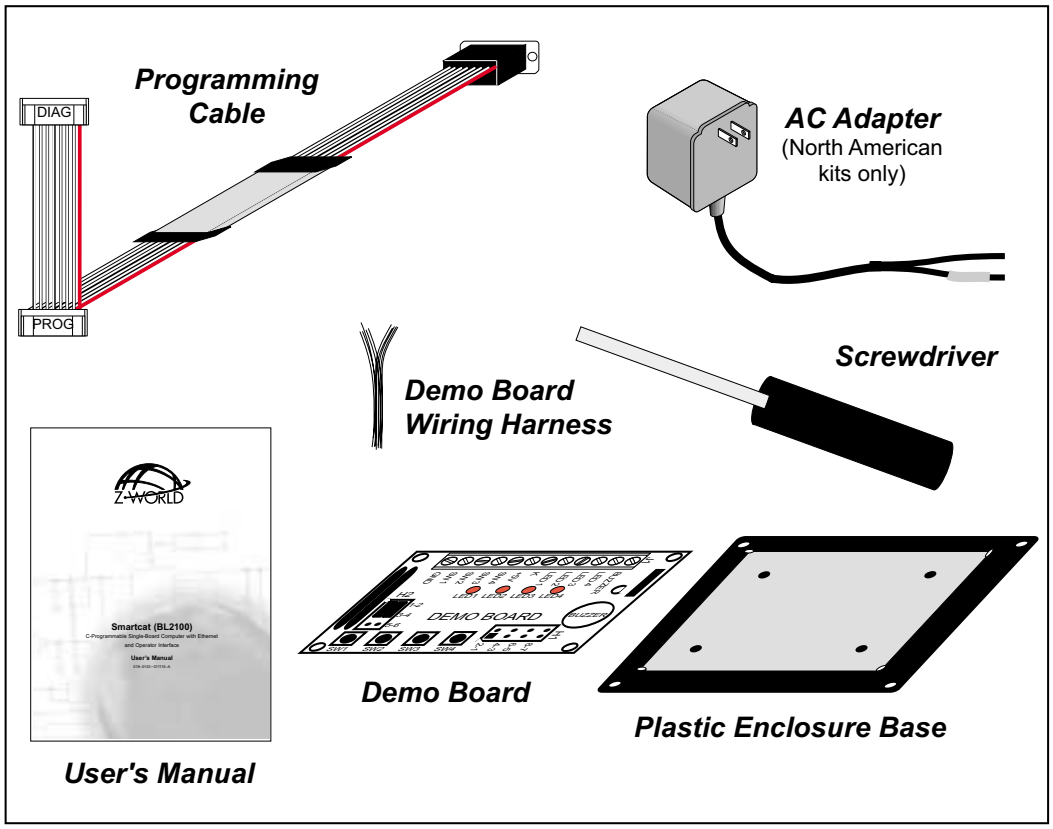



***Figure 1. BL2100 Tool Kit***

### 1.4.2 Software

The BL2100 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 BL2100. Software drivers for 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 BL2100. Basic Ethernet network connections are shown, and instructions for configuring the network parameters on the BL2100 are included.

## 2.1 BL2100 Connections

1. Remove the RabbitCore module from the BL2100 main board, and set the module aside. The module is removed to allow access to the mounting holes on the main BL2100 board, and will be plugged back in to the main board later.

   **NOTE:** If you are working with more than one BL2100 at a time, take care to keep the BL2100 main boards and their corresponding RabbitCore modules paired since the RabbitCore modules store calibration constants specific to the BL2100 main board to which they are plugged in.

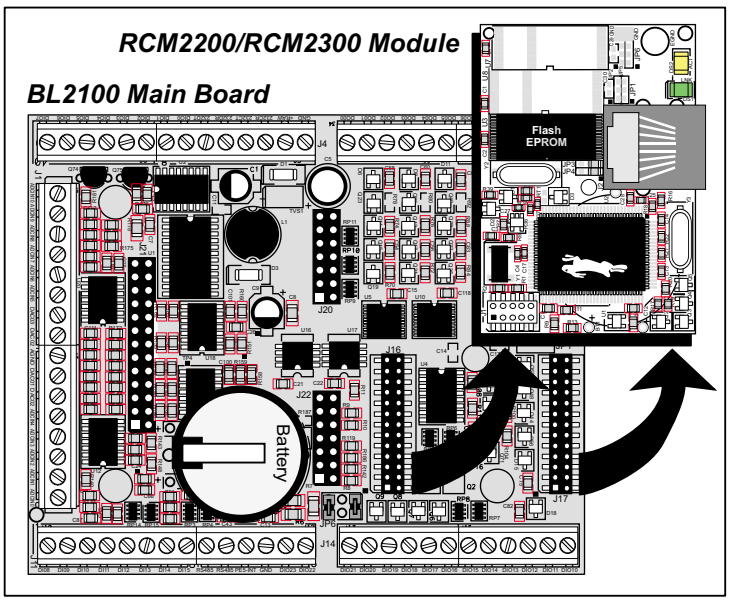



***Figure 2. Remove RabbitCore Module from BL2100 Main Board***

2. Attach the BL2100 main board to the plastic enclosure base.

Position the BL2100 main board over the plastic enclosure base as shown below in Figure 3. Attach the BL2100 to the base using the four 4-40 × ¼ screws supplied with the enclosure base.

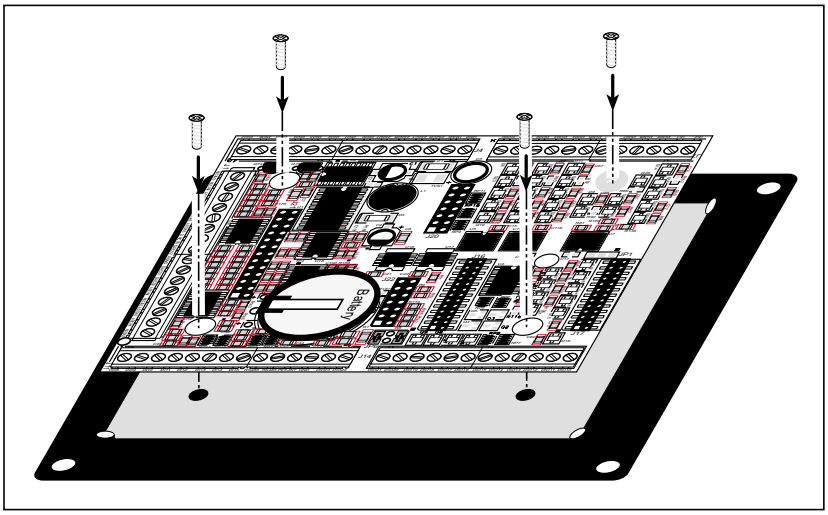


***Figure 3. Attach BL2100 Main Board to Plastic Enclosure Base***

The plastic enclosure base facilitates handling the BL2100 during development, and provides an attractive mounting alternative. Alternatively, you may wish to use standoffs to protect the components on the other side of the board. The plastic enclosure base is offered as a separate option when individual BL2100 boards are purchased.

> **NOTE:** Appendix D, “Plastic Enclosure,” provides additional information and specifications for the plastic enclosure.

3. Reconnect the RabbitCore module to headers J16 and J17 on the BL2100 main board it was removed from earlier as shown in Figure 4. Be careful to align the pins over the headers, and do not bend them as you press down to mate the module with the BL2100 main board.

**NOTE:** If you are working with more than one BL2100 at a time, take care to keep the BL2100 main boards and their corresponding RabbitCore modules paired since the RabbitCore modules store calibration constants specific to the BL2100 main board to which they are plugged in.

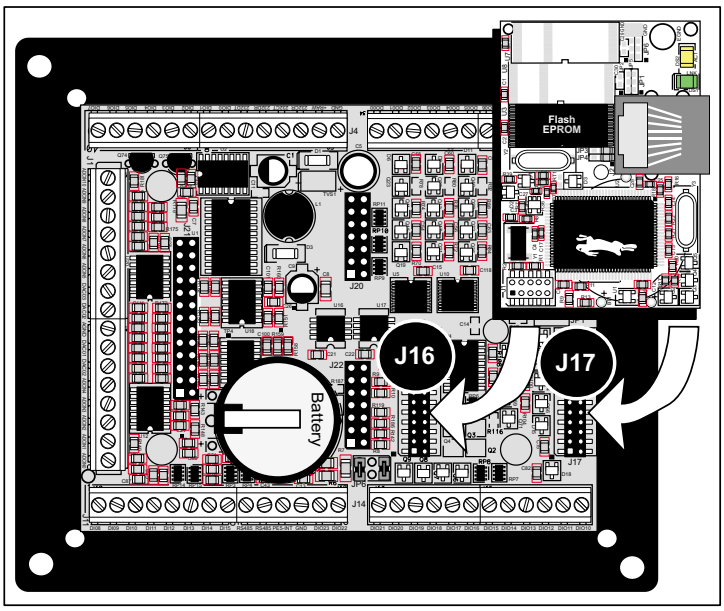



***Figure 4. Reconnect RabbitCore Module to BL2100 Main Board***

4. Connect the programming cable to download programs from your PC and to program and debug the BL2100.

Connect the 10-pin **PROG** connector of the programming cable to header J1 on the BL2100 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 F, "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.

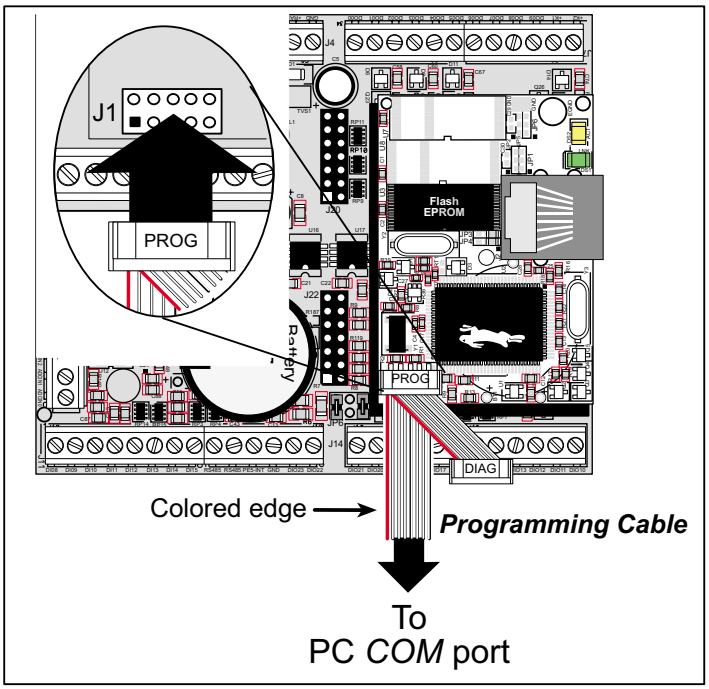



***Figure 5. Programming Cable Connections***

5. Connect the power supply.

Connect the bare ends of the power supply to the **+RAW** and **GND** positions on screw terminal header J4 as shown in Figure 6.

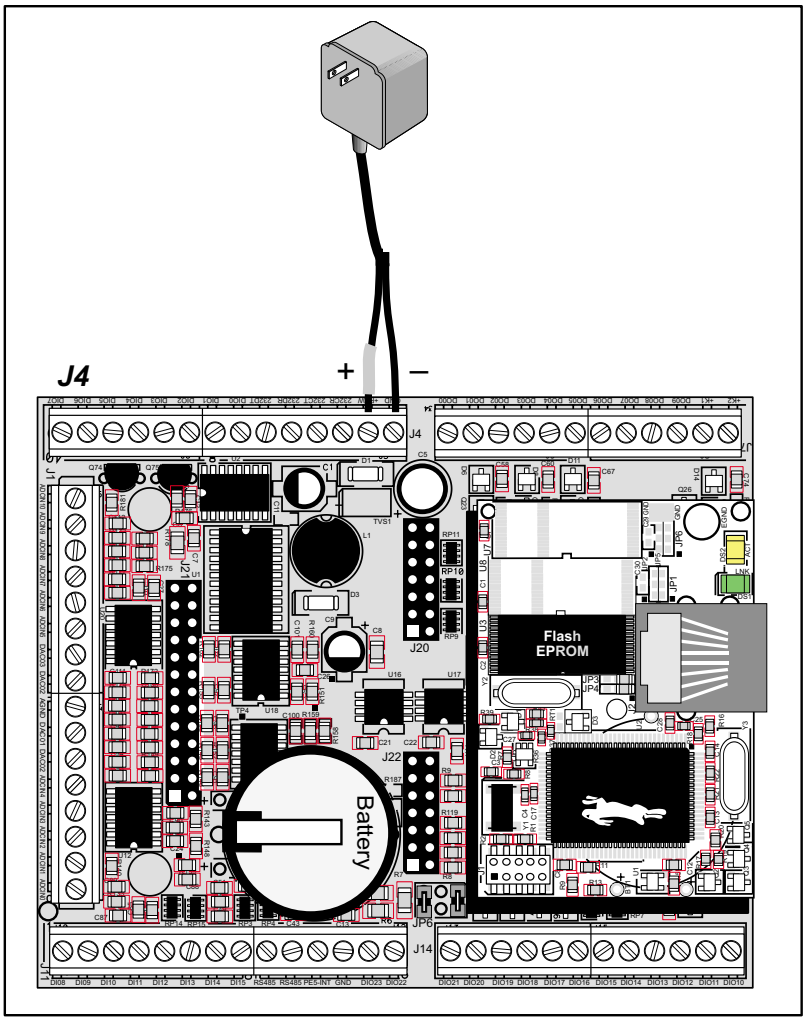



***Figure 6. Power Supply Connections***

6. Apply power.

Plug in the AC adapter. If you are using your own power supply, it must provide 9 V to 36 V DC—voltages outside this range could damage the BL2100.

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

## 2.2 Installing Dynamic C Premier

If you have not yet installed Dynamic C version 7.06 (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.

## 2.3 Starting Dynamic C

Once the BL2100 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 BL2100 and go through a sequence of steps to cold-boot the BL2100 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 BL2100 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 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.

## 2.4 PONG.C

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

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

This program 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.

## 2.5 Where Do I Go From Here?

If there are any problems at this point:

- Check the Z-World Technical Bulletin Board at http://www.zworld.com/support/bb/index.html.
- E-mail your questions to support@zworld.com.
- 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 BL2100 features and develop your own applications.

Chapter 3, "Subsystems," provides a description of the BL2100's features, Chapter 4, "Software," describes the Dynamic C software libraries and introduces some sample programs, and Chapter 5, "Using the TCP/IP Features," explains the TCP/IP features.

# 3. SUBSYSTEMS

Chapter 3 describes the principal subsystems for the BL2100.

- Digital I/O
- Serial Communication
- A/D Converter Inputs
- D/A Converter Outputs
- Analog Reference Voltage Circuit
- Memory
- External Interrupts

Figure 7 shows these Rabbit-based subsystems designed into the BL2100.

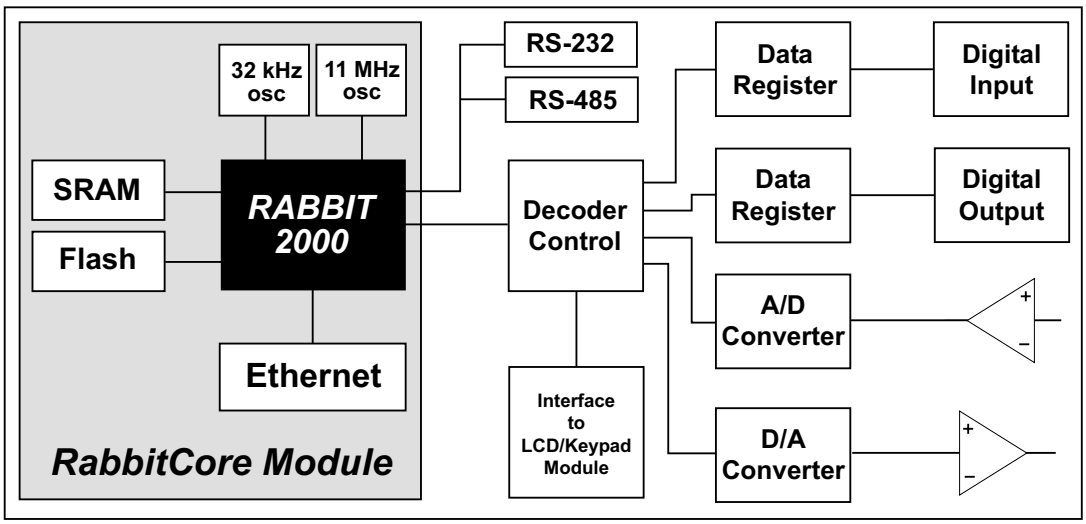



***Figure 7. BL2100 Subsystems***

## 3.1 BL2100 Pinouts

The BL2100 pinouts are shown in Figure 8.

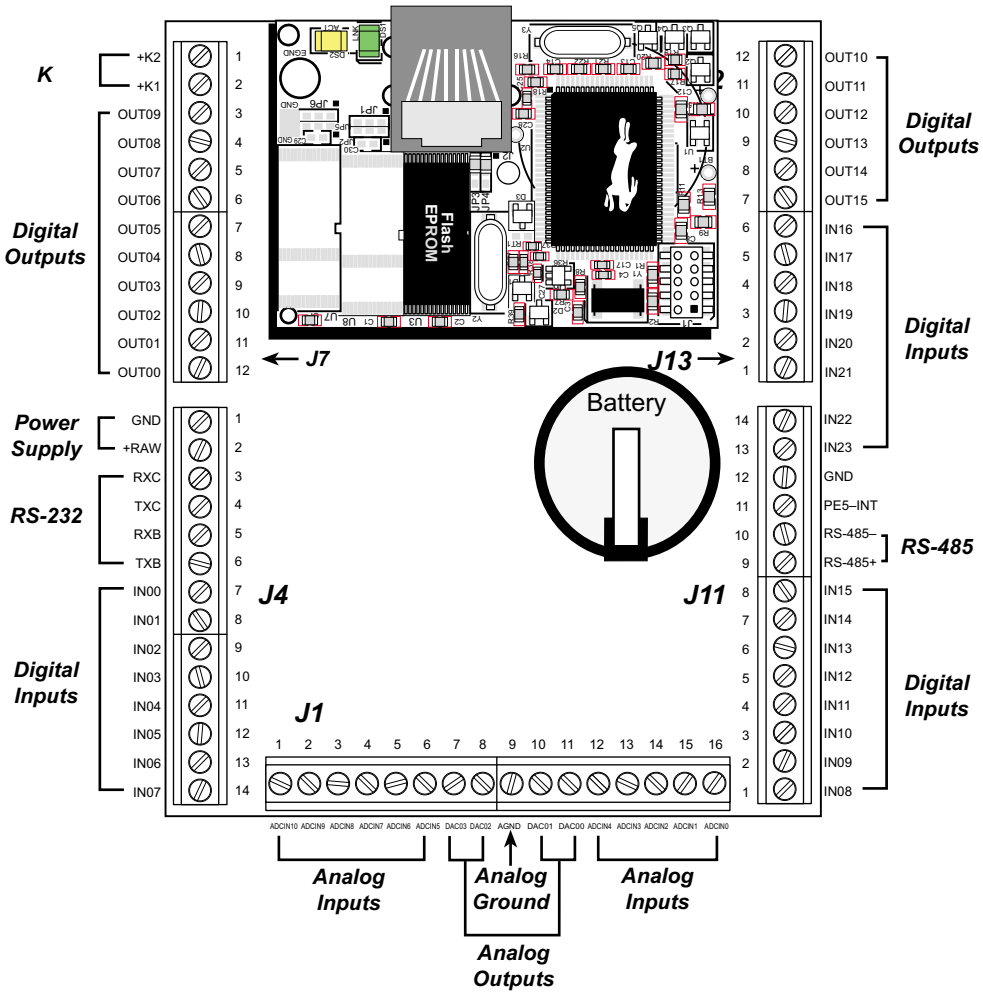



***Figure 8. BL2100 Pinouts***

**NOTE:** Screw-terminal header J1 and the associated analog I/O are not available on the BL2110 and the BL2130.

### 3.1.1 Headers and Screw Terminals

Standard BL2100 models are equipped with two 1 × 12 screw terminal strips (J7 and J13), and two 1 × 14 screw terminal strips (J4 and J11). The BL2100 and BL2110 also have the RJ-45 Ethernet jack and one 1 × 16 screw terminal strip (J1).

There is provision on the circuit board to accommodate 2 × 17, 2 × 20, and 2 × 25 male headers instead of the screw terminal strips supplied, and the male headers can be factory-installed by special request for volume orders. Contact your Z-World Sales Representative at +1(530)757-3737 for more information.

## 3.2 Digital I/O

### 3.2.1 Digital Inputs

The BL2100 has 24 digital inputs, IN00–IN23, each of which is protected over a range of –36 V to +36 V. The inputs are factory-configured to be pulled up to +5 V, but they can also be pulled up to +K2 or down to 0 V in banks of eight by changing a surface-mounted 0 Ω resistor as shown in Figure 9.

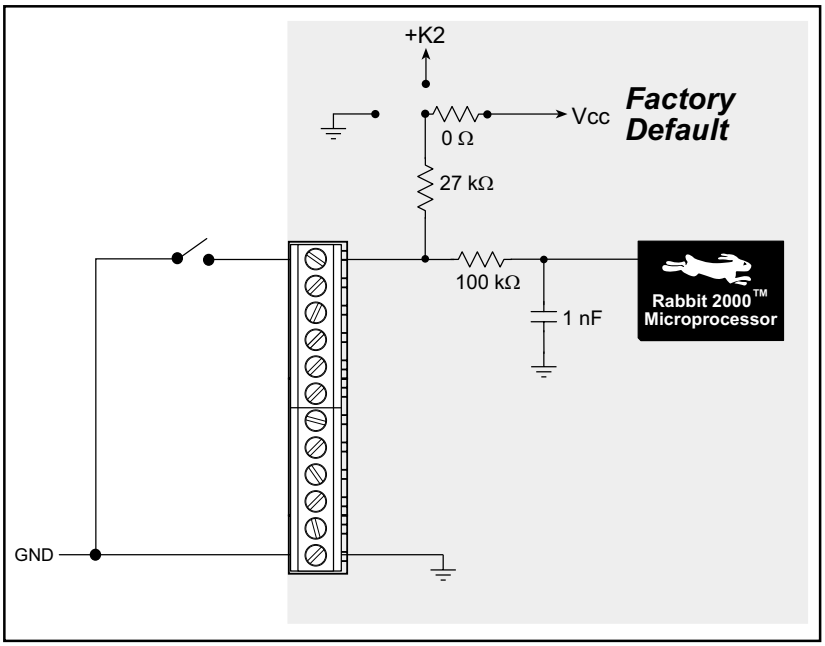



***Figure 9. BL2100 Digital Inputs [Pulled Up—Factory Default]***

**NOTE:** If the inputs are pulled up to +K2, the voltage range over which the digital inputs are protected changes to K2 – 36 V to +36 V.

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

IN16–IN23 can be factory-configured as outputs for users who prefer to have 16 inputs and 24 outputs.

### 3.2.2 Digital Outputs

The BL2100 has 16 digital outputs, OUT00–OUT15, which can each sink or source up to 200 mA. Figure 10 shows a wiring diagram for using the digital outputs in a sinking or a souring configuration.

All the digital outputs sink and source actively. They can be used as high-side drivers, low-side drivers, or as an H-bridge driver. When the BL2100 is first powered up or reset, all the outputs are disabled, that is, at a high-impedance status, until the **`digoutConfig`** software function call is made. The **`digoutConfig`** call sets the initial state of each digital output according to the configuration specified by the user, and enables the digital outputs to their initial status.

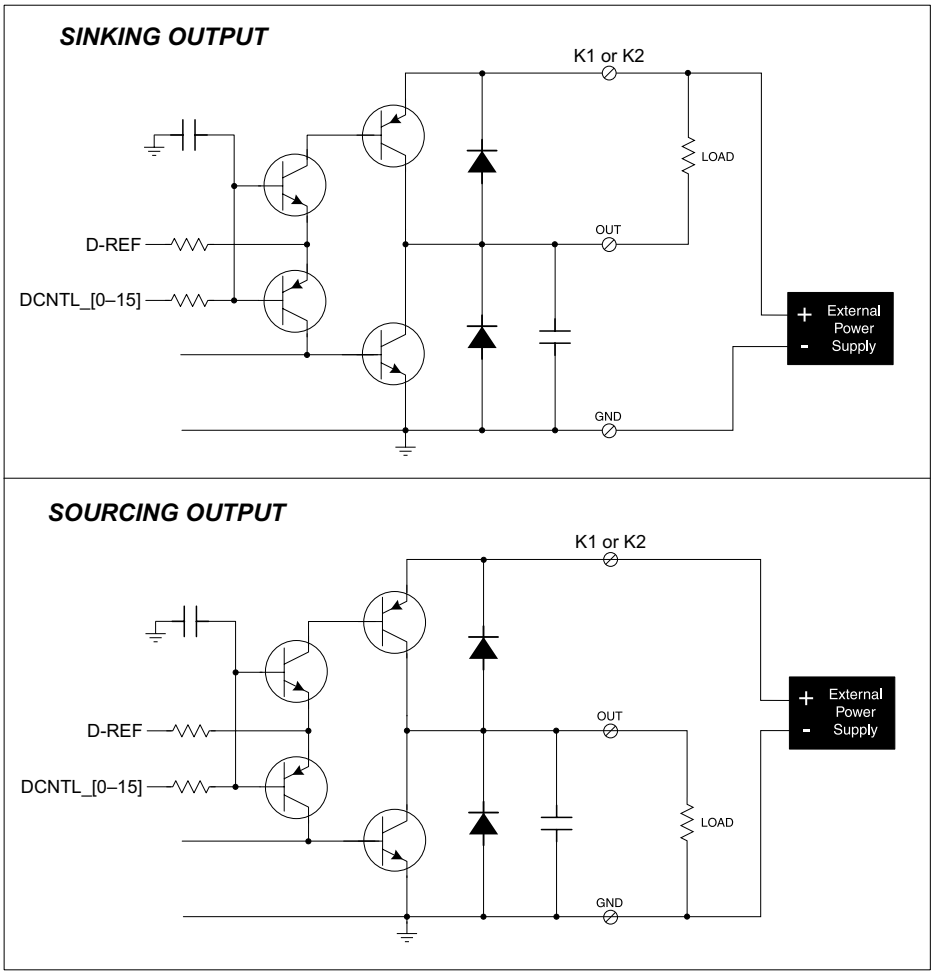



***Figure 10. BL2100 Digital Outputs***

OUT00–OUT07 are powered by to +K1, and OUT08–OUT15 are powered by +K2. K1 and K2 can each be up to 36 V. They don't have to be same.

All the sinking current, which could be up to 3.2 A, is returned through the GND pins. Be sure to use a suitably sized GND and keep the distance to the power supply as short as possible. Since there are two GND terminals (pin 1, screw-terminal header J4, and pin 12, screw-terminal header J11), it is highly recommend that you split the GND returns according to the two banks of digital outputs.

For the H bridge, which is shown in Figure 11, K1 and K2 ***should be the same*** if two digital outputs used for the H bridge are on different banks.

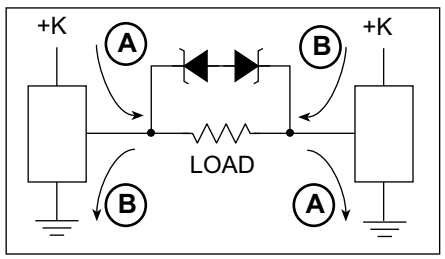



***Figure 11. H Bridge***

## 3.3 Serial Communication

The BL2100 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 2 summarizes the options.

***Table 2. 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 BL2100 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 BL2100 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 BL2100 uses an 11.0592 MHz crystal, which is doubled to 22.1184 MHz. At this frequency, the BL2100 supports standard asynchronous baud rates up to a maximum of 230,400 bps.

### 3.3.1 RS-232

The BL2100 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 BL2100's maximum baud rate for distances of up to 15 m.

### 3.3.2 RS-485

The BL2100 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 BL2100 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 12. Note that a common ground is recommended.

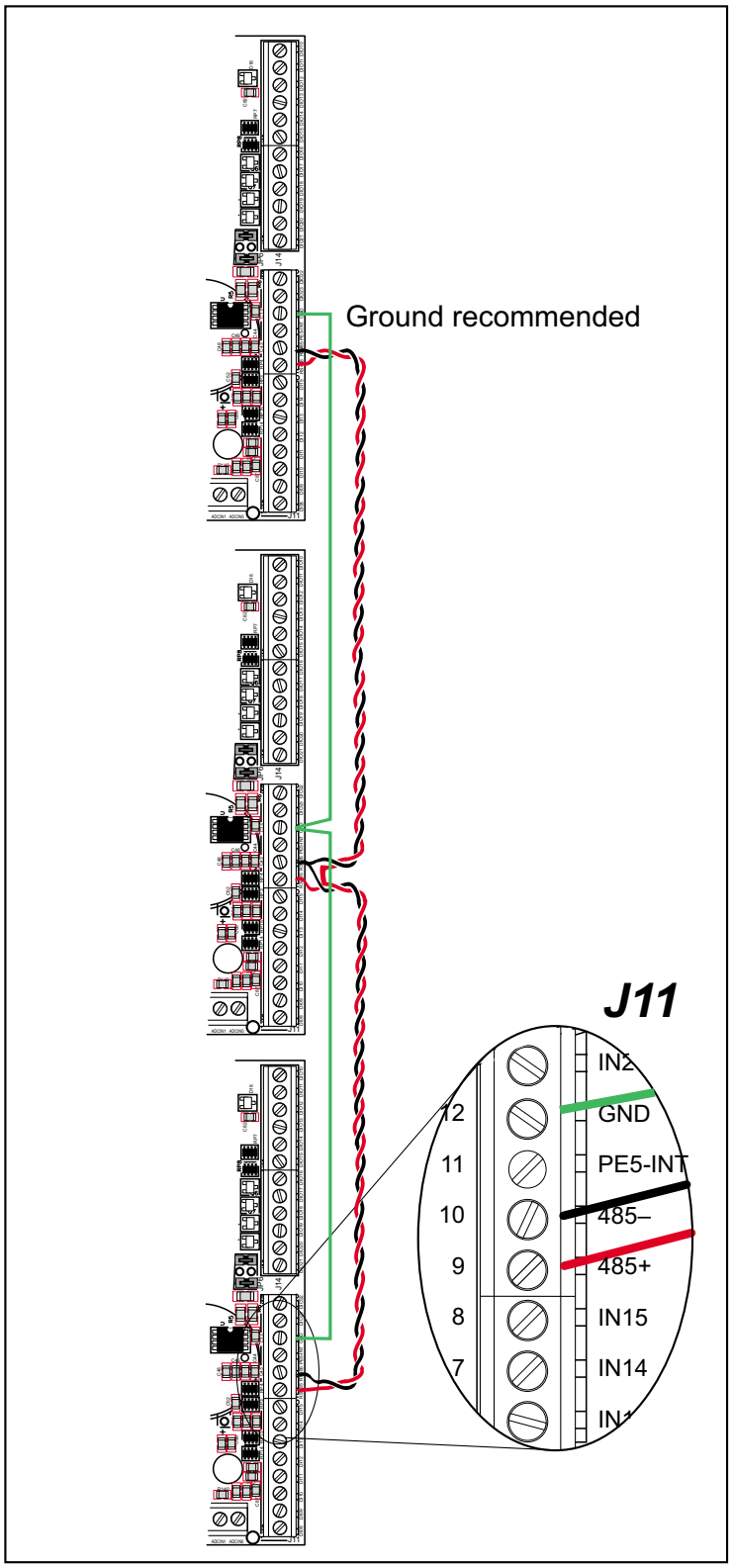



*Figure 12. BL2100 Multidrop Network*

The BL2100 comes with a 220 Ω termination resistor and two 681 Ω bias resistors installed and enabled with jumpers across pins 1–2 and 5–6 on header JP1, as shown in Figure 13.

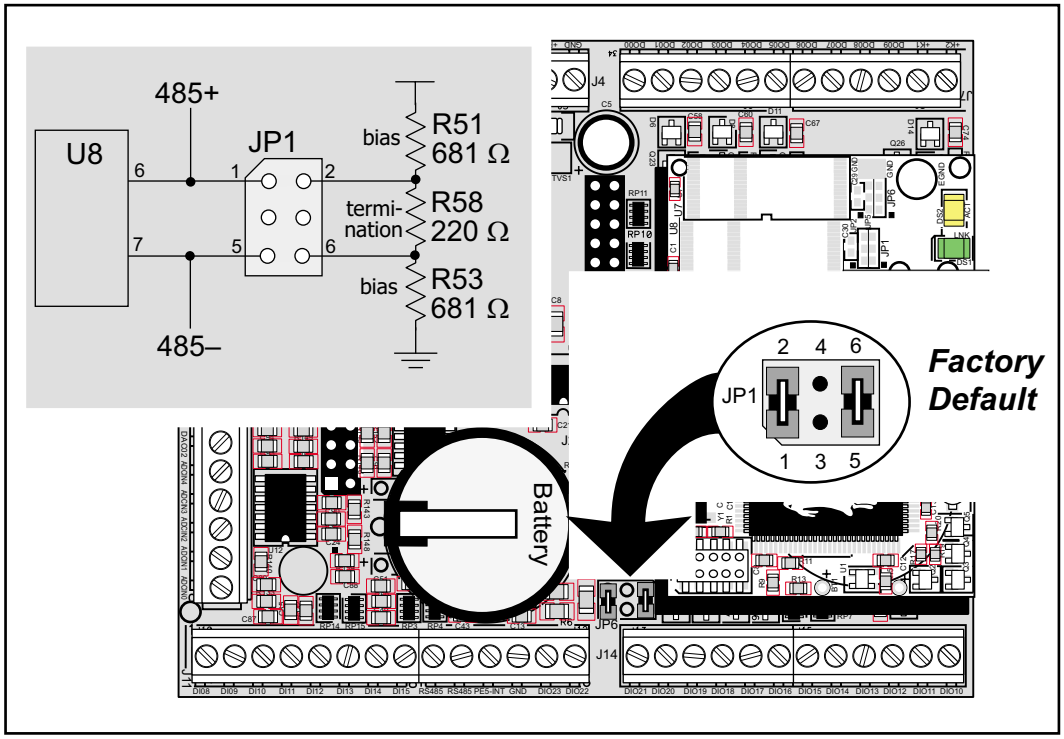



***Figure 13. RS-485 Termination and Bias Resistors***

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 BL2100 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 BL2100.

### 3.3.3 Programming Port

The RabbitCore module on the BL2100 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 BL2100 in a mode where the BL2100 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 BL2100 to start up in an external bootstrap mode. The BL2100 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

Figure 14 shows the pinout for the Ethernet port (J2 on the BL2100 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 14) are always absolute, and the RJ-45 connector will work properly with off-the-shelf Ethernet cables.

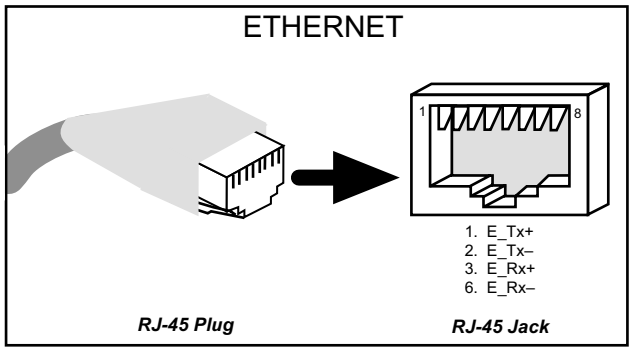



***Figure 14. RJ-45 Ethernet Port Pinout***

RJ-45 pinouts are sometimes numbered opposite to the way shown in Figure 14.

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 Ω resistor "jumper," R29, as shown in Figure 15.

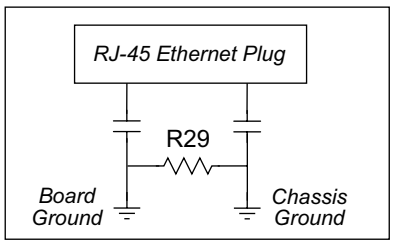



***Figure 15. Isolation Resistor R29***

The factory default is for the 0 Ω 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 A/D Converter Inputs

The single 14-channel A/D converter used in the BL2100 has a resolution of 12 bits (models BL2100 and BL2120 only). Eleven of the 14 channels are available externally, and three are used internally for the reference voltages: 4.096 V ($V_{ref}$), 2.048 V ($V_{ref}/2$), and Analog Ground. These internal voltages can be used to check the functioning of the A/D converter.

The A/D converter only measures voltages between 0 V and the applied reference voltage. Therefore, each external input has circuitry that provides scaling and buffering. All 11 external inputs are scaled and buffered to provide the user with an input impedance of 1 MΩ and a range of -10.24 V to +10.24 V.

Figure 16 shows the buffered A/D converter inputs.

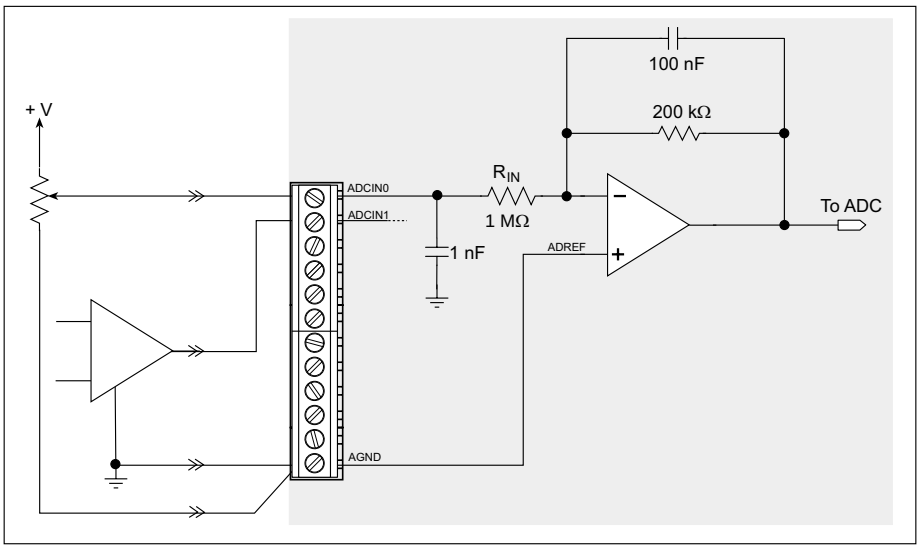



***Figure 16. Buffered A/D Converter Inputs***

The op-amp is powered from the +V supply. The 1 MΩ and 200 kΩ resistors set the gain (scale factor), which is 0.2 in this case. This results in a dynamic input range of 0.2 × 20.48 V or 4.096 V. The center point of this range is set by the 1.707 V reference voltage. With the reference set to 1.707 V, the center point is at 0 V and the input voltage can range from -10.24 V to +10.24 V. To maintain the best accuracy, the input range should be limited to -10.0 V to +10.0 V.

The A/D converter inputs are factory-calibrated and the calibration constants are stored in flash memory. You may calibrate the A/D converter inputs at a later time using the software functions described in Section 4.3.4, “A/D Converter Inputs.” The **`GETCALIB.C`** and the **`SAVECALIB.C`** sample programs in the Dynamic C **`SAMPLES\BL2100\Calib_Save_Retrieve`** folder illustrate how to retrieve and save calibration data.

## 3.5 D/A Converter Outputs

Only the BL2100 and the BL2120 models are stuffed with D/A converters. The D/A converter outputs are buffered and scaled to provide an output from 0 V to +10 V.

> **NOTE:** The D/A converter output voltage depends on the original power-supply voltage, +RAW, so if +RAW < 13 V, the maximum D/A converter output will be +RAW – 3 V.

Figure 17 shows the D/A converter outputs.

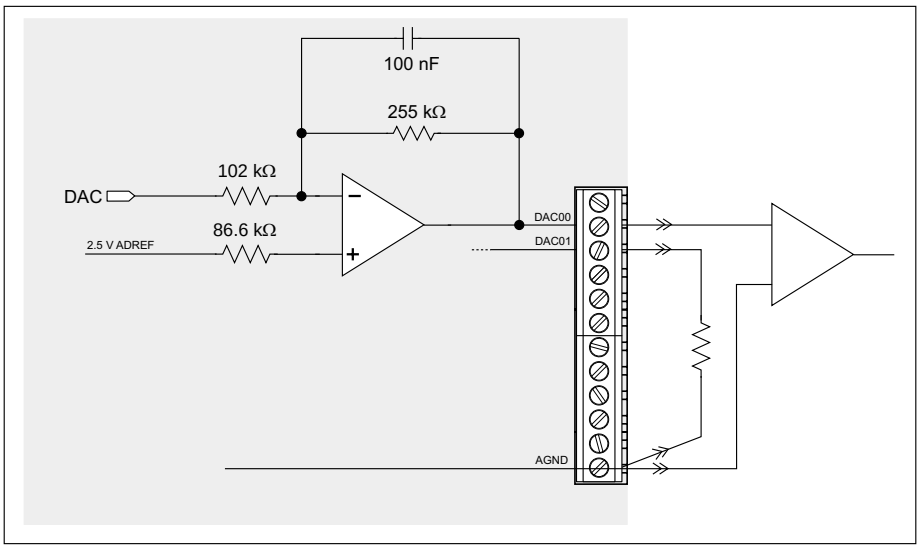



***Figure 17. D/A Converter Outputs***

To stay within the maximum power dissipation of the D/A converter circuit, the maximum D/A converter output current is 10 mA per channel for a power-supply voltage, +RAW, up to 15 V, and drops to 2 mA per channel for a power-supply voltage of 36 V.

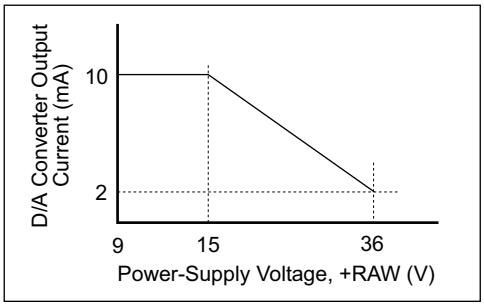



***Figure 18. Maximum D/A Converter Output Current vs. Power-Supply Voltage***

The D/A converter inputs are factory-calibrated and the calibration constants are stored in flash memory. You may calibrate the A/D converter inputs at a later time using the software functions described in Section 4.3.5, “D/A Converter Outputs.” The **`GETCALIB.C`** and the **`SAVECALIB.C`** sample programs in the Dynamic C **`SAMPLES\BL2100\Calib_Save_Retrieve`** folder illustrate how to retrieve and save calibration data.

## 3.6 Analog Reference Voltage Circuit

Figure 19 shows the analog voltage reference circuit.

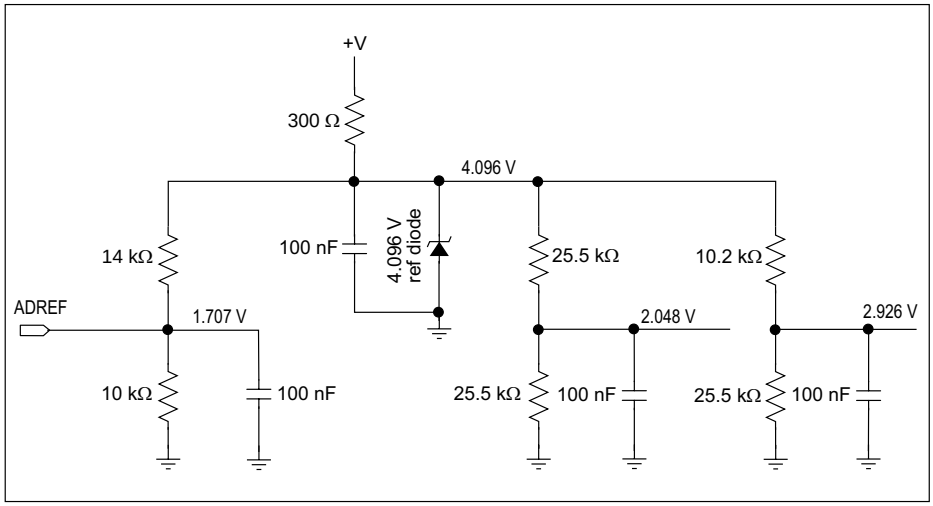



***Figure 19. Analog Reference Voltages***

This circuit generates the 4.096 V reference voltage, which is used by the A/D converter and by the D/A converters. This sets the operating range of the A/D converter and the D/A converters (0–10 V). To use the full accuracy of the A/D converter and the D/A converters, this voltage must be accurate to the same degree.

The reference zener diode in combination with the 300 Ω resistor form a shunt regulator. The 4.096 V reference voltage then feeds the A/D converter, the D/A converters, and the voltage divider composed of the 10 kΩ and the 14 kΩ resistors. The voltage divider generates a second reference voltage of 1.707 V to feed the four op-amps for the buffered A/D converter inputs.

The 2.048 V reference voltage is also used to generate the 2.5 V reference for D-REF used in the digital output circuit.

## 3.7 Memory

### 3.7.1 SRAM

The BL2100 module is designed to accept 128K to 512K of SRAM packaged in an SOIC case. The standard BL2100 modules come with 128K of SRAM.

### 3.7.2 Flash Memory

The BL2100 is also designed to accept 128K to 512K of flash memory packaged in a TSOP case. The standard BL2100 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 Ω 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.

## 3.8 External Interrupts

The BL2100 is already configured to support external interrupts on pin 11 of screw terminal header J11. The external interrupt circuit is shown in Figure 20.

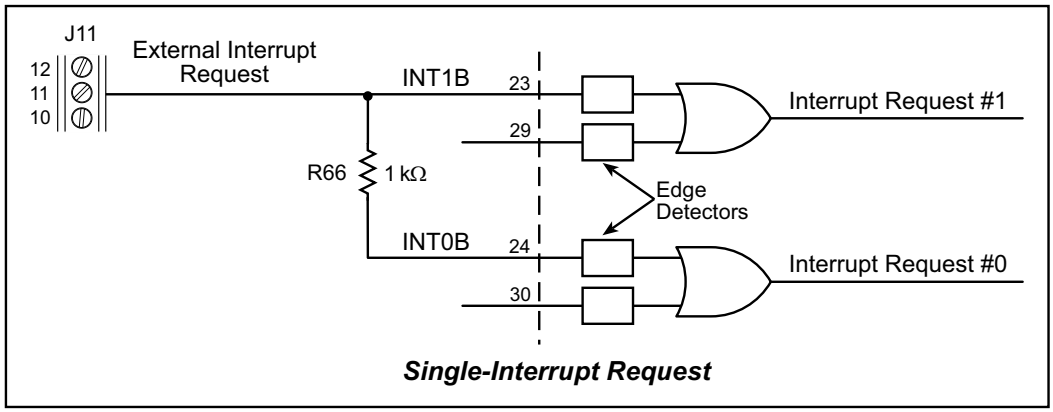



***Figure 20. Use of Rabbit 2000 External Interrupt***

**NOTE:** Refer to Technical Note TN301, ***Rabbit 2000 Microprocessor Interrupt Problem***, for more information. The modification given here for the BL2000 takes into account the workaround fix recommended in TN301.

In addition to its primary use as an external interrupt, pin 11 of screw terminal header J11 may also be used as a CMOS-level digital input or output, or to generate a PWM signal.

When using pin 11 as a CMOS-level digital input or output, use the standard Rabbit 2000 register function configuration for PE5 (on Parallel Port E) to set this pin up for your intended use. Be aware that there is no provision for protection against voltage spikes while PE5 is pulled up to Vcc with a 27 kΩ pull-up resistor.

The sample program **PWM.C** in the Dynamic C **SAMPLES/BL2100** directory illustrates how to use pin 11 of screw terminal header J11 to generate a PWM signal.

# 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 BL2100.

You have a choice of doing your software development in the flash memory or in the static RAM included on the BL2100. 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 BL2100 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 BL2100 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 BL2100 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 21.

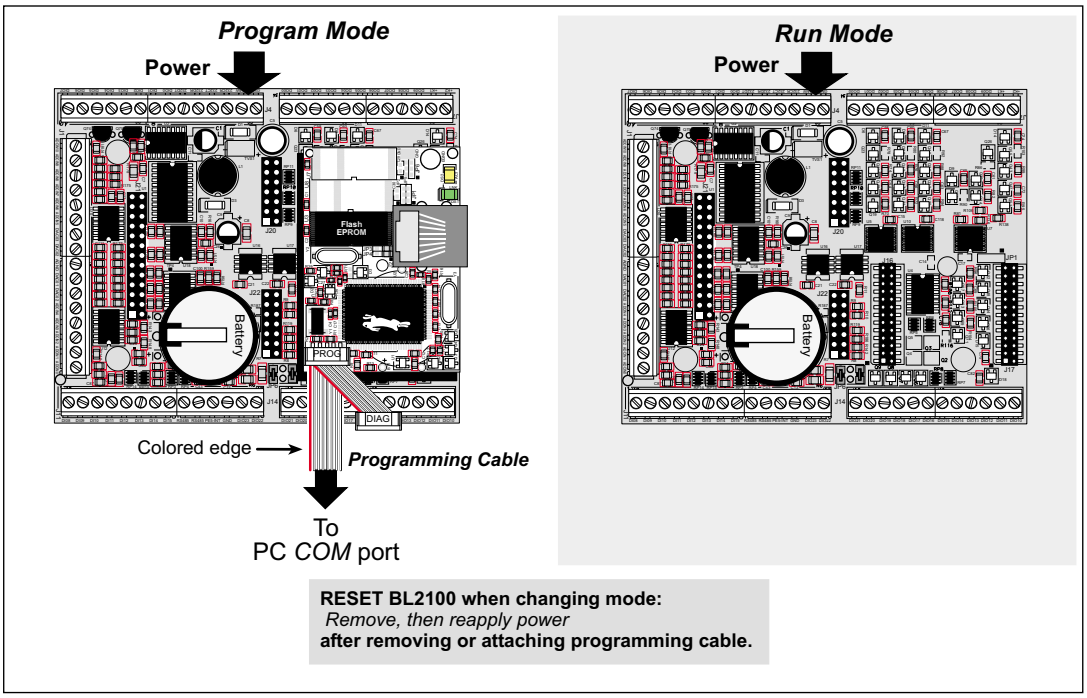



***Figure 21. BL2100 Program Mode and Run Mode Set-Up***

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

1. Disconnect the programming cable from header J1 of the BL2100 module.
2. Reset the BL2100 by unplugging the AC adapter, then plugging it back in.

The BL2100 is now ready to operate in the Run Mode.

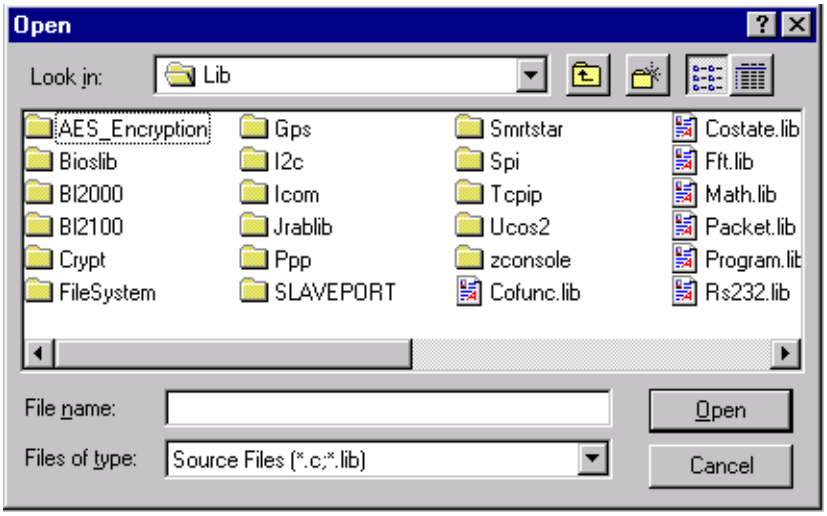
### 4.1.3 Detailed Instructions: Changing from Run Mode to Program Mode

1. Attach the programming cable to header J1 of the BL2100 module.
2. Reset the BL2100 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 BL2100 is now ready to operate in the Program Mode.

## 4.2 BL2100 Libraries

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




Two library directories provide libraries of function calls that are used to develop applications for the BL2100.

- **BL2100**—libraries associated with features specific to the BL2100. The functions in the **BL21xx.LIB** library are described in Section 4.3, "BL2100 Function APIs."
- **TCPIP**—libraries specific to using TCP/IP functions on the BL2100.

Two other library directories provide libraries of function calls that are used to develop applications for the optional BL2100 LCD/keypad module.

- **DISPLAYS\GRAPHIC**—libraries associated with LCD display.
- **KEYPADS**–libraries associated with the keypad.

The LCD/keypad module functions are described in Section C.6. Other generic functions applicable to all devices based on the Rabbit 2000 microprocessor are described in the ***Dynamic C Premier User's Manual***.

## 4.3 BL2100 Function APIs

### 4.3.1 Board Initialization

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

Call this function at the beginning of your program. This function initializes the system I/O ports and loads all the A/D converter and D/A converter calibration constants from flash memory into SRAM for use by your program. If the LCD/keypad module is installed, this function will turn off LED DS1 to indicate that the initialization was successful.

The ports are initialized according to Table A-4 in Appendix A-5.

**SEE ALSO**

**`digOut, digIn, serMode, anaOut, anaIn, anaInDriver, anaOutDriver`**

### 4.3.2 Digital I/O

## void digOutConfig(unsigned int outputMode);

Each of the BL2100 high-current outputs (OUT00–OUT15) has the capability of being configured in software as either sinking or sourcing using the **digOutConfig** function. Execute **digOutConfig** at the start of your application to initially set OUT00–OUT15 to be OFF for the type of circuit that you have, either sinking or sourcing.

To properly set the high-current outputs, you will need to decide for each channel whether the output is sinking or sourcing. The **digOutConfig** function will then ensure that each output remains OFF when the digital output control interface is initialized. The individual high-current outputs remaininactivated until you activate the desired output driver(s)/channel(s) using **digOut**.

**NOTE:** The **brdInit** function must be executed before calling **digOutConfig**.

**NOTE:** You must execute the **digOutConfig** function to set the high-current drivers to be either sinking or sourcing. A runtime error will occur in **digOut** if **digOutConfig** has not executed.

**NOTE:** The extra digital outputs resulting from the reconfiguration of IN16–IN23 as digital outputs are sinking outputs only and cannot be configured with **digOutConfig**.

#### PARAMETER

**outputMode** is a 16-bit parameter where each bit corresponds to one of the following high current outputs.

Bit 15 = high-current output channel OUT15
Bit 14 = high-current output channel OUT14
Bit 13 = high-current output channel OUT13
Bit 12 = high-current output channel OUT12
Bit 11 = high-current output channel OUT11
Bit 10 = high-current output channel OUT10
Bit 9 = high-current output channel OUT09
Bit 8 = high-current output channel OUT08
Bit 7 = high-current output channel OUT07
Bit 6 = high-current output channel OUT06
Bit 5 = high-current output channel OUT05
Bit 4 = high-current output channel OUT04
Bit 3 = high-current output channel OUT03
Bit 2 = high-current output channel OUT02
Bit 1 = high-current output channel OUT01
Bit 0 = high-current output channel OUT00

The high-current outputs can be configured to be sinking or sourcing outputs by setting the corresponding bit to an 0 or 1: 0 = sinking, 1 = sourcing.

#### RETURN VALUE

None.

#### SEE ALSO

**brdInit, digOut**

#### EXAMPLE

```
outputMode = 0x0ff1; // Outputs OUT15–OUT12 = Sinking
                     // Outputs OUT11–OUT08 = Sourcing
                     // Outputs OUT07–OUT04 = Sourcing
                     // Outputs OUT03–OUT01 = Sinking
                     // Output OUT00 = Sourcing
```

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

Sets the state of a digital output (OUT00–OUT15).

Remember to call the **brdInit** and the **digOutConfig** functions before executing this function.

A runtime error will occur for the following conditions:

1. **channel** or **value** out of range.
2. **brdInit** or **digOutConfig** was not executed before executing **digOut**.

#### PARAMETERS

**channel** is the output channel number (0–15, 0–23 if IN16–IN23 are configured as digital outputs).

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

#### SEE ALSO

**brdInit, digIn, digOutConfig**

## int digIn(int channel);

Reads the state of an input channel.

A run-time error will occur for the following conditions:

1. **channel** out of range.
2. **brdInit** was not executed before executing **digIn**.

#### PARAMETER

**channel** is the input channel number (0–23)

#### RETURN VALUE

The state of the input (0 or 1).

#### SEE ALSO

**brdInit, digOut**

### 4.3.3 Serial Communication

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 BL2100.

## int serMode(int mode);

User interface to set up BL2100 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**

## void ser485Tx(void);

Sets pin 3 (DE) high to enable the RS-485 transmitter.

**SEE ALSO**

**serMode, ser485Rx**

## void ser485Rx(void);

Resets pin 3 (DE) low to disable the RS-485 transmitter.

**SEE ALSO**

**serMode, ser485Tx, serCflowcontrolOn, serCflowcontrolOff**

### 4.3.4 A/D Converter Inputs

The functions in this section apply only to the BL2100 and the BL2120 models.

```
int anaInCalib(int channel, int value1,
   float volts1, int value2, float volts2);
```

Calibrates the response of the A/D converter channel as a linear function using the two conversion points provided. Gain and offset constants are calculated and placed into global table **_adcCalib**.

**PARAMETERS**

**channel** is the A/D converter input channel (0–10).

**value1** is the first A/D converter channel value (0–4095).

**volts1** is the voltage corresponding to the first A/D converter channel value (-10 V to +10 V).

**value2** is the second A/D converter channel value (0–4095).

**volts2** is the voltage corresponding to the second A/D converter channel value (-10 V to +10 V).

**RETURN VALUE**

0 if successful.

-1 if not able to make calibration constants.

**SEE ALSO**

**anaIn, anaInVolts, brdInit**

## int anaInDriver(unsigned char cmd, char len);

Reads the voltage of an analog input channel by serially clocking out an 8-bit command to the A/D converter. The driver has been designed for the Texas Instruments TLC2543 A/D converter used on the BL2100 and the BL2120.

### PARAMETERS

**cmd** is formatted as follows.

**TLC2543 commands**

D7–D4

Channel 0–10
Channel 11 = ($V_{ref+}$ - $V_{ref-}$)/2
Channel 12 = $V_{ref-}$
Channel 13 = $V_{ref+}$
Channel 14 = software powerdown

D3–D2

Output data length:
01—8 bits
00—12 bits (normally used as default)
11—16 bits (not supported by driver)

D1

Output data format
0—MSB first
1—LSB first (not supported by driver)

D0

Mode of operation
0—Unipolar (normally used as default)
1—Bipolar

**len** is the output data length:

0 = 12-bit mode
1 = 8-bit mode

### RETURN VALUE

A value corresponding to the voltage on the A/D converter input channel, which will be:

0–4095 for 12-bit A/D conversions
0–255 for 8-bit A/D conversions

### SEE ALSO

**anaIn, anaInVolts, brdInit**

### EXAMPLE

Look at the sample programs in **SAMPLES\BL2100\ADC**.

## int anaIn(unsigned int channel);

Reads the state of an A/D converter input channel.

**PARAMETER**

**channel** is the A/D converter input channel (0–10) to read.

**RETURN VALUE**

A value corresponding to the voltage on the analog input channel (0–4095).

**SEE ALSO**

**anaInVolts, anaInCalib, anaInfast, brdInit**

## float anaInVolts(unsigned int channel);

Reads the state of an A/D converter input channel and uses the previously set calibration constants to convert it to volts.

**PARAMETER**

**channel** is the A/D converter input channel (0–10).

**RETURN VALUE**

A voltage value corresponding to the voltage on the analog input channel.

**SEE ALSO**

**anaIn, anaInCalib, brdInit**

## int anaInEERd(unsigned int channel);

Reads the calibration constants, gain, and offset from the simulated EEPROM in flash memory (located in reserved user block memory area 0x1C00–0x1FFF).

**PARAMETER**

**channel** is the A/D converter input channel (0–10).

**RETURN VALUE**

0 if successful.

-1 if address is invalid or out of range.

**SEE ALSO**

**anaInEEWr, brdInit**

## int anaInEEWr(unsigned int channel);

Writes the calibration constants, gain, and offset to the simulated EEPROM in flash memory (located in reserved user block memory area 0x1C00–0x1FFF).

### PARAMETER

**channel** is the A/D converter input channel (0–10) for which the calibration constants will be read.

### RETURN VALUE

0 if successful.

-1 if address is invalid or out of range.

### SEE ALSO

**anaInEERd, brdInit**

### 4.3.5 D/A Converter Outputs

The functions in this section apply only to the BL2100 and the BL2120 models.

```
int anaOutCalib(int channel, int value1,
   float volts1, int value2, float volts2);
```

Calibrates the response of the D/A converter channel desired as a linear function using the two conversion points provided. Gain and offset constants are calculated and placed into global table **_dacCalib**.

**PARAMETERS**

**channel** is the D/A converter output channel (0–3).

**value1** is the first D/A converter value (0–4095).

**volts1** is the voltage corresponding to the first D/A converter value (0 V to +10 V).

**value2** is the second D/A converter value (0–4095).

**volts2** is the voltage corresponding to the second D/A converter value (0 V to +10 V).

**RETURN VALUE**

0 if sucessful.

-1 if not able to make calibration constants.

**SEE ALSO**

**anaOut, anaOutVolts, brdInit**

```
void anaOutDriver(int power_control,
   int speed_control, int channel,
   unsigned int rawcount);
```

Sets the voltage of a D/A converter output channel by serially clocking in 16 bits to a D/A converter using the following format:

D15, D12

Register R1, Register R0

00—Write data to DAC OUTB
01—Write data to buffer
10—Write data to DAC OUTA
11—Reserved

D14

Speed control
0—slow
1—fast (default)

D13

Power control
0—normal (default)
1—powerdown

D11–D0

Data bits, MSB–LSB (0–4095)

**PARAMETERS**

**`power_control`** is the D/A converter power control option (0—normal (default) or 1—powerdown).

When the power-down mode is selected, the only other parameter that is used is the D/A converter channel (**`channel`**). The values of the other parameters are not considered.

Two D/A converter channels are affected when putting a D/A converter output in powerdown or normal mode.

Powerdown Mode:

When **`power_control`** equals 1 and **`channel`** is 0 or 1, then both D/A converter channels 0 and 1 are put in powerdown mode (channels 2 and 3 not affected).

When **`power_control`** equals 1 and **`channel`** is 2 or 3, then both D/A converter channels 2 and 3 are put in powerdown mode (channels 0 and 1 not affected).

Normal Mode:

When **`power_control`** equals 1 and **`channel`** is 0 or 1, then both D/A converter channels 0 and 1 are put in normal mode. (channels 2 and 3 not affected).

When **`power_control`** equals 1 and **`channel`** is 2 or 3, then both D/A converter channels 2 and 3 are put in normal mode (channels 0 and 1 not affected).

**speed_control** is the D/A converter power control option (0—slow or 1—fast (default)).

| Mode | Speed vs. Power Dissipation |
|---|---|
| 0—slow | 12 µs access vs. 1 mA |
| 1—fast (default) | 3 µs access vs. 2.3 mA |

Test conditions from TI's data sheet (TLV5618A D/A converter) for the speed-control option:
- No load.
- All inputs are at GND or VDD.
- D/A converter latch = 0x800.

**channel** is the D/A converter output channel to write (0–3).

**rawcount** is the data value corresponding to the desired voltage on the analog output channel (0–4095).

#### RETURN VALUE

None

#### SEE ALSO

**anaOut, anaOutVolts, anaOutCalib**

## void anaOut(unsigned int channel, unsigned int rawcount);

Sets the voltage of a D/A converter output channel.

#### PARAMETERS

**channel** is the D/A converter output channel (0–3).

**rawcount** is a data value corresponding to the voltage desired on the output channel (0–4095).

#### RETURN VALUE

0 if sucessful.

-1 if **rawcount** is more than 4095.

#### SEE ALSO

**anaOutDriver, anaOutVolts, anaOutCalib**

## void anaOutVolts(unsigned int ch, float voltage);

Sets the voltage of a D/A converter output channel by using the previously set calibration constants to calculate the correct data values.

#### PARAMETERS

**channel** is the D/A converter output channel (0–3).

**voltage** is the voltage desired on the output channel.

#### RETURN VALUE

None.

#### SEE ALSO

**anaOut, anaOutCalib, brdInit**

## int anaOutEERd(unsigned int channel);

Reads the calibration constants, gain, and offset from the simulated EEPROM in flash memory (located in reserved user block memory area 0x1C00–0x1FFF).

**PARAMETER**

**channel** is the D/A converter output channel (0–3).

**RETURN VALUE**

0 if successful.

-1 if address or range is invalid.

**SEE ALSO**

**anaOutEEWr, brdInit**

## int anaOutEEWr(unsigned int channel);

Writes the calibration constants, gain, and offset to the simulated EEPROM in flash memory (located in reserved user block memory area 0x1C00–0x1FFF).

**PARAMETER**

**channel** is the D/A converter output channel (0–3).

**RETURN VALUE**

0 if successful.

-1 if address or range is invalid.

**SEE ALSO**

**anaOutEERd, brdInit**

### 4.3.6 TCP/IP Function APIs

The **TCPIP** directory contains libraries with generic TCP/IP functions for the BL2100 and the BL2110.

- **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***.

## 4.4 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 **BL2100** folder provides sample programs specific to the BL2100. 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 BL2100 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, "BL2100 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."

### 4.4.1 Digital I/O

The following sample programs are found in the **IO** subdirectory in **SAMPLES\BL2100**.

- **DIGIN.C**—Demonstrates the use of the digital inputs. Using the Demonstration Board, you can see an input channel toggle from HIGH to LOW when pressing a pushbutton on the Demonstration Board. See Appendix D for hookup instructions for the Demonstration Board.
- **DIGOUT.C**—Demonstrates the use of the high-current outputs configured as either sinking or sourcing outputs. Using the Demonstration Board, you can see an LED toggle on/off via a high-current output. See Appendix D for hookup instructions for the Demonstration Board.
- **PWM.C**—Demonstrates the use of Timer B to generate a PWM signal on PE5-INT located on screw terminal header J11. The program generates a 42 Hz PWM signal with the duty cycle adjustable from 1 to 99%.

### 4.4.2 Serial Communication

The following sample programs are found in the **RS232** subdirectory in **SAMPLES\BL2100**.

- **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\BL2100**.

- **MASTER.C**—This program demonstrates a simple RS-485 transmission of lower case letters to a slave BL2100. The slave will send back converted upper case letters back to the master BL2100 and display them in the **STDIO** window. Use **SLAVE.C** to program the slave BL2100.
- **SLAVE.C**—This program demonstrates a simple RS-485 transmission of lower case letters to a slave BL2100. The slave will send back converted upper case letters back to the master BL2100 and display them in the **STDIO** window. Use **MASTER.C** to program the master BL2100.

### 4.4.3 A/D Converter Inputs

The following sample programs are found in the **ADC** subdirectory in **SAMPLES\BL2100**.

- **AD_CALIB.C**—Demonstrates how to recalibrate an A/D converter channel using two knownvoltages to generate two coefficients, gain and offset, which are rewritten into the user block data area. The voltage that is being monitored is displayed continuously. Make sure that you don't exceed the voltage range of the A/D converter input channel.

  **NOTE:** This sample program will overwrite the calibration constants set at the factory.

- **AD1.C**—Demonstrates how to access the A/D internal test voltages in both the TLC2543 and TLC1543 A/D converter chips. The program reads the A/D internal voltages and then uses the **STDIO** window to display the RAW data.
- **AD2.C**—Demonstrates how to access the A/D channels using the **anaInVolt** function. The program uses the **STDIO** window to display the voltage that is being monitored.
- **AD3.C**—Demonstrates how to access the A/D converter channels with the low-level A/D driver. The program uses the **STDIO** window to display the voltage that is being monitored on all the A/D channels using the low-level A/D driver.
- **AD4.C**—Demonstrates how how to use the A/D converter channels with the low-level A/D driver. The program uses the **STDIO** window to display the voltage (average of 10 samples) that is being monitored on all the A/D converter channels using the low-level A/D driver.

### 4.4.4 D/A Converter Outputs

The following sample programs are found in the **DAC** subdirectory in **SAMPLES\BL2100**.

- **DACAL.C**—This program demonstrates how to recalibrate an D/A converter channel using two known voltages, and defines the two coefficients, gain and offset, that will be rewritten into the D/A converter's EEPROM simulated in flash memory.

  **NOTE:** This sample program will overwrite the calibration constants set at the factory.

- **DAOUT1.C**—This program outputs a voltage that can be read with a voltmeter. The output voltage is computed using the calibration constants that are read from the EEPROM simulated in flash memory.

- **DAOUT2.C**—This program demonstrates the use of both the D/A and the A/D converters. The user selects both the D/A converter and A/D channel to be used, then sets the D/A converter output voltage to be read by the A/D channel. All activity will be displayed in the **STDIO** window.

### 4.4.5 Using Calibration Constants

The following sample programs are found in the **Calib_Save_Retrieve** subdirectory in **SAMPLES\BL2100**. Note that both sample programs prompt you to use a serial number for the BL2100. This serial number can be any 5-digit number of your choice, and will be unique to a particular BL2100. Do ***not*** use the MAC address on the bar code label of the RabbitCore module attached to the BL2100 since you may at some later time use that particular RabbitCore module on another BL2100, and the previously saved calibration data would no longer apply.

- **GETCALIB.C**—This program demonstrates how to retrieve your analog calibration data to rewrite it back to the simulated EEPROM in flash with using a serial utility such as Tera Term.

  **NOTE:** Calibration data must be saved previously in a file by the sample program **SAVECALIB.C**.

- **SAVECALIB.C**—This program demonstrates how to save your analog calibration coefficients using a serial port and a PC serial utility such as Tera Term.

  **NOTE:** Use the sample program **GETCALIB.C** to retrieve the data and rewrite it to the single-board computer.

### 4.4.6 TCP/IP Sample Programs

TCP/IP sample programs are described in Chapter 5.

### 4.4.7 LCD/Keypad Module Sample Programs

Sample programs for the LCD/keypad module are described in Section C.7.

# 5. USING THE TCP/IP FEATURES

Chapter 5 discusses using the TCP/IP features on the BL2100 and BL2110 boards. The TCP/IP feature is ***not*** available on BL2120 and BL2130 versions.

## 5.1 TCP/IP Connections

Before proceeding you will need to have the following items.

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

The Ethernet cables and Ethernet hub are available from Z-World in a TCP/IP tool kit. More information is available at 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 BL2100 to a PC that at least has a 10Base-T Ethernet card.

If you have Ethernet access, use a straight through Ethernet cable to establish an Ethernet connection to the BL2100 from an Ethernet hub. These connections are shown in Figure 22.

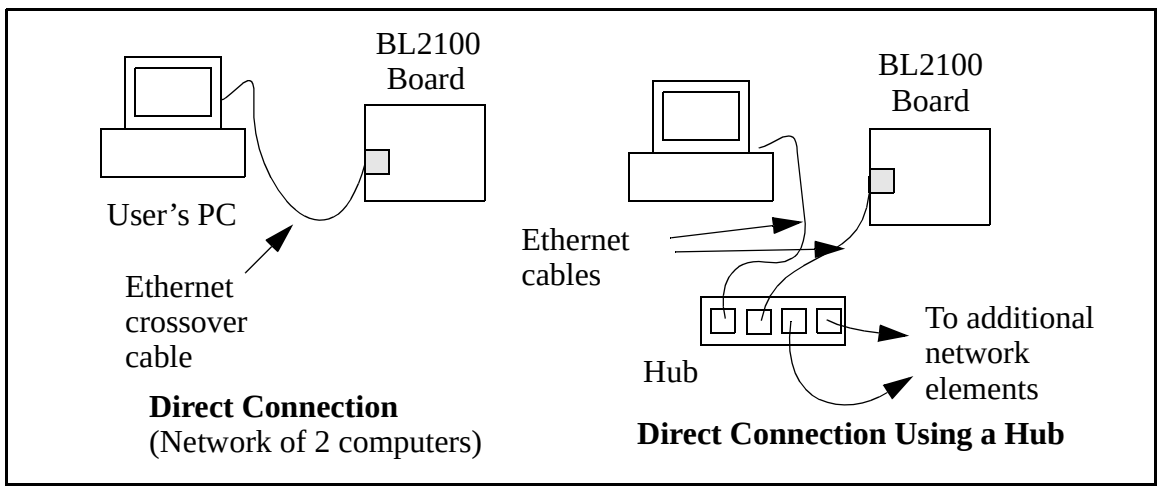



***Figure 22. Ethernet Connections***

The PC running Dynamic C through the serial programming port on the BL2100 does not need to be the PC with the Ethernet card.

3. Apply Power

Plug in the AC adapter. The BL2100 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.

When working with the BL2100, the green **LNK** light is on when a program is running and the board is properly connected either to an Ethernet hub or to an active Ethernet card. The 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 BL2100 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 BL2100:

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

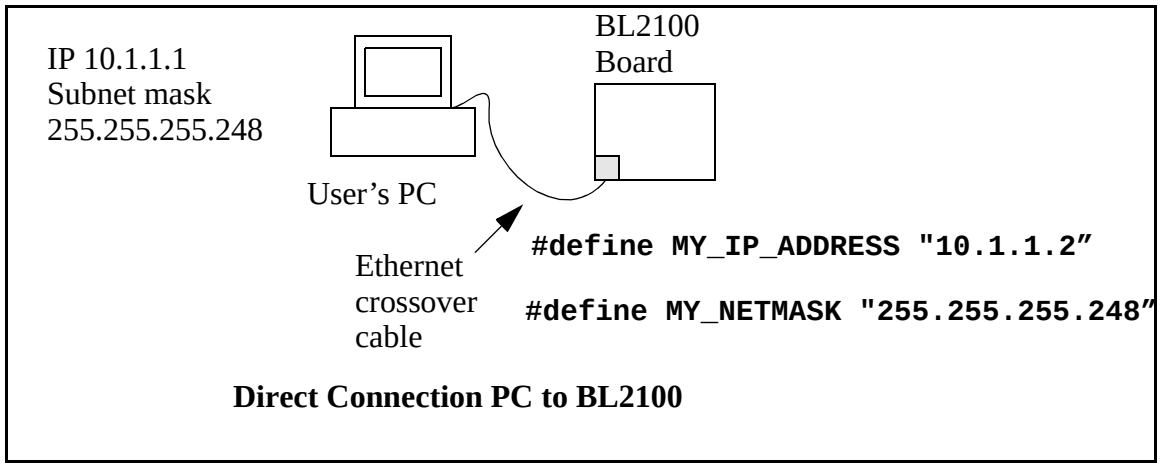
In this case, the gateway is not used and is commented out. The IP address of the board is defined to be 10.1.1.2. The IP address of your PC can be defined as 10.1.1.1.

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

When your computer is connected directly to the BL2100 via an Ethernet connection, you need to assign an IP address to your computer. To assign the PC the address 10.1.1.1 with the subnetmask 255.255.255.248 under Windows 98, do the following.

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

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




**Direct Connection PC to BL2100**

### 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 BL2100's RJ-45 Ethernet connector. When the program starts running, the green **LNK** light on the BL2100 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 BL2100 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\BL2100\TCPIP**) demonstrates how to make the BL2100 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 **SMTP.C** (**SAMPLES\BL2100\TCPIP**) allows you to send an E-mail when a switch on the Demonstartion Board is pressed. Follow the instructions included with the sample program.

The sample program **TELNET.C** (**SAMPLES\BL2100\TCPIP**) allows you to communicate with the BL2100 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.

## 5.3 Where Do I Go From Here?

If there are any problems at this point:

- Check the Z-World Technical Bulletin Board at http://www.zworld.com/support/bb/index.html.
- E-mail your questions to support@zworld.com.
- 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.

# APPENDIX A. SPECIFICATIONS

Appendix A provides the specifications for the BL2100 and describes the conformal coating.

## A.1 Electrical and Mechanical Specifications

Figure A-1 shows the mechanical dimensions for the BL2100.

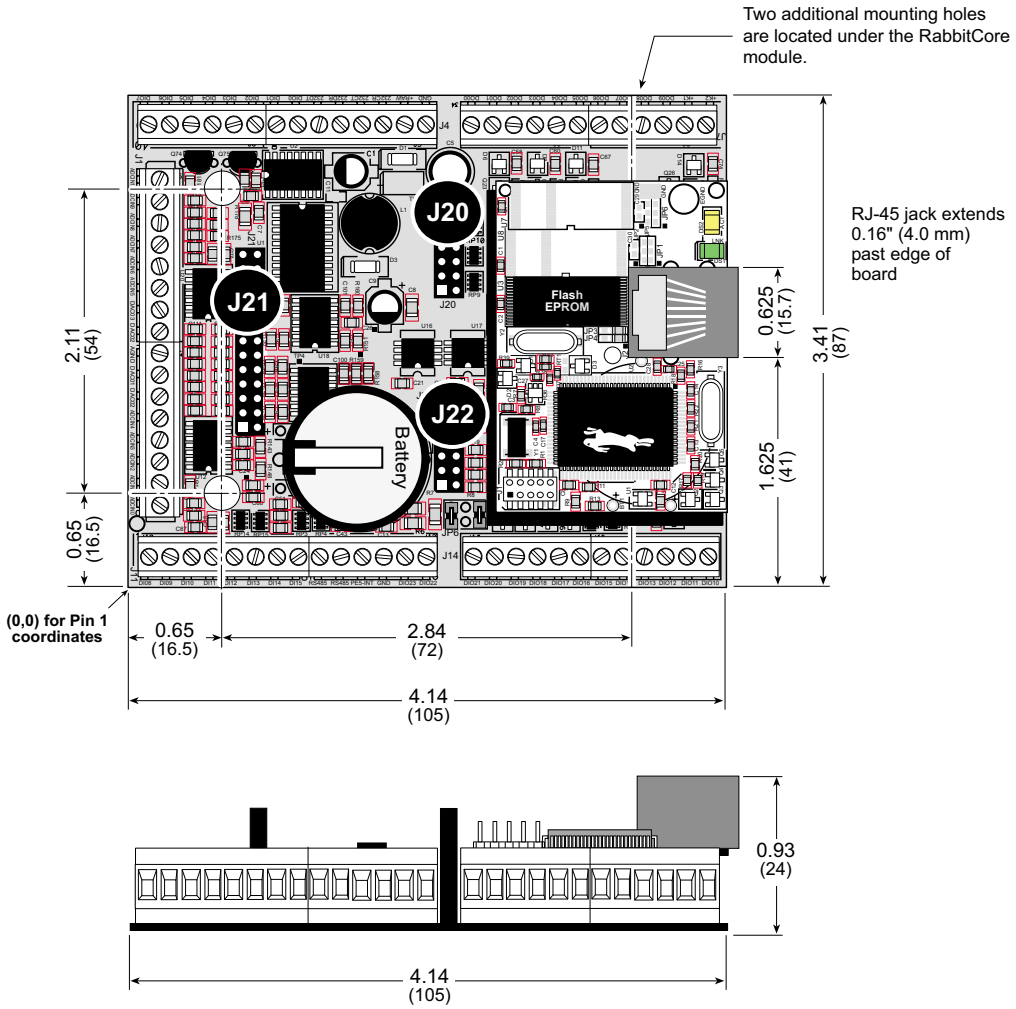



***Figure A-1. BL2100 Dimensions***

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

Table A-1 provides the pin 1 locations for the BL2100 headers used to mount the LCD/keypad module as viewed in Figure A-1.

***Table A-1. BL2100 LCD/Keypad Header Pin 1 Locations***

| Header | Pin 1 (x,y) Coordinates (inches) |
|---|---|
| J20 | (2.170, 2.055 |
| J21 | (0.795, 1.105) |
| J22 | (2.170, 0.705) |

It is recommended that you allow for an “exclusion zone” of 0.25" (6 mm) around the BL2100 in all directions when the BL2100 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. An “exclusion zone” of 0.12" (3 mm) is recommended below the BL2100. Figure A-2 shows this “exclusion zone.”

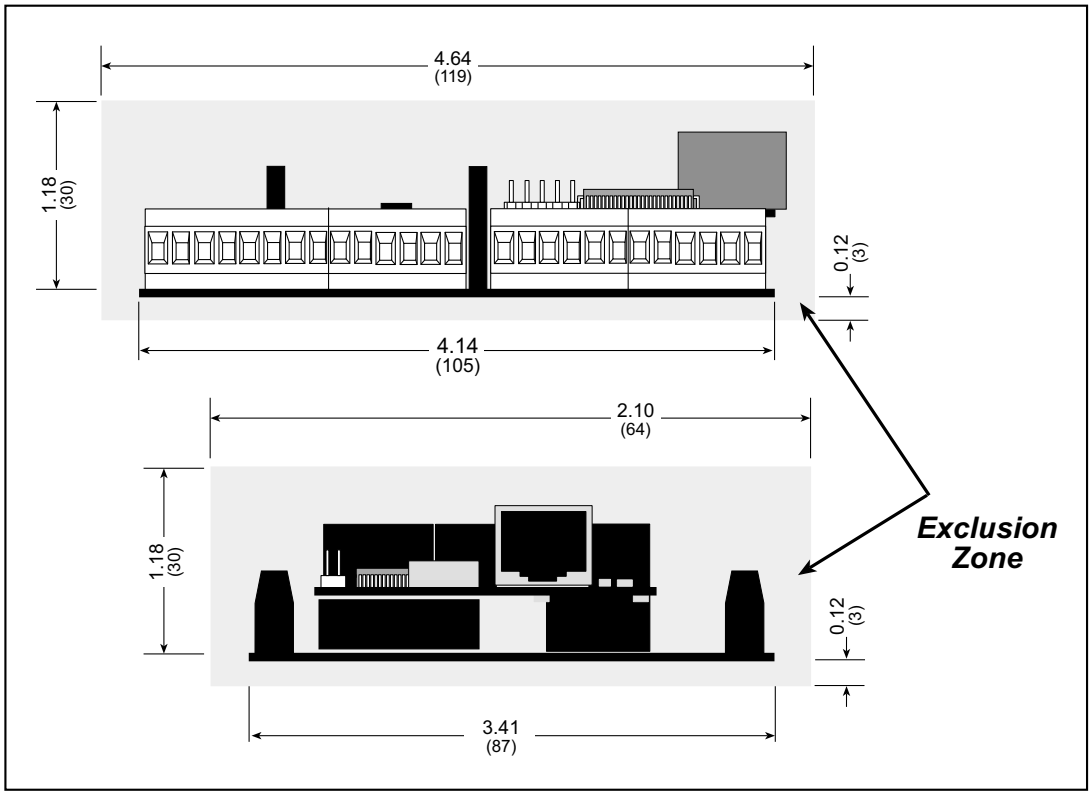



***Figure A-2. BL2100 “Exclusion Zone”***

Table A-2 lists the electrical, mechanical, and environmental specifications for the BL2100 without the optional LCD/keypad module plugged in. Appendix C provides specifications for the LCD/keypad.

### *Table A-2. BL2100 Specifications*

| Parameter | Specification |
|---|---|
| Board Size | 3.41" × 4.14" × 0.93"<br>(87 mm × 105 mm × 24 mm) |
| Connectors | one RJ-45 (Ethernet) (BL2100 and BL2110)<br>one 2 × 5, 2 mm pitch (serial programming port)<br>five 0.9 mm × 0.5 screw-terminal connector strips (accept 14–30 AWG or 0.05–1.5 mm² wire) |
| Ethernet Interface<br>(BL2100 and BL2110) | Direct connection to 10Base-T Ethernet networks via RJ-45 connection |
| Temperature | –40°C to +70°C |
| Humidity | 5% to 95%, noncondensing |
| External Input Voltage | 9 V to 36 V DC* |
| Power Consumption | 1.5 W maximum |
| Digital I/O | 40 digital I/O:<br>24 inputs: hardware-configurable pull-up or pull-down, ± 36 V DC, switching threshold 2.4 V typical<br>16 outputs: software actively toggles sinking and sourcing, +36 V DC, 200 mA maximum per channel |
| Analog Inputs (BL2100 and BL2120) | Eleven 12-bit A/D converter inputs, ± 10 V DC, 1 MΩ input impedance<br>Sampling rate: 4.3 kHz |
| Analog Outputs (BL2100 and BL2120) | Four 12-bit D/A converter outputs, 0 V to 10.0 V DC, 10 mA max. per output channel<br>Update rate: 10 kHz |
| 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 cascadable |
| Serial Ports | 4 serial ports:<br>• two RS-232 or one RS-232 (with CTS/RTS)<br>• one RS-485, onboard network termination and bias resistors<br>• one 5 V CMOS-compatible programming port |
| Serial Rate | Maximum standard asynchronous 230,400 bps |

*Table A-2. BL2100 Specifications (continued)*

| Parameter | Specification |
|---|---|
| Watchdog/Supervisor | Yes |
| Time/Date Clock | Yes |
| Backup Battery | Yes: Panasonic CR2330 or equivalent 3 V lithium coin type, 265 mA·h standard using onboard battery holder; optional 3 V, 950 mA·h solder-in battery available |

* 13 V to 36 V DC supply voltage required to support full 0–10 V DC output range of D/A converter

## A.2 Conformal Coating

The areas around the crystal oscillator and the battery backup circuit on the BL2100 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.

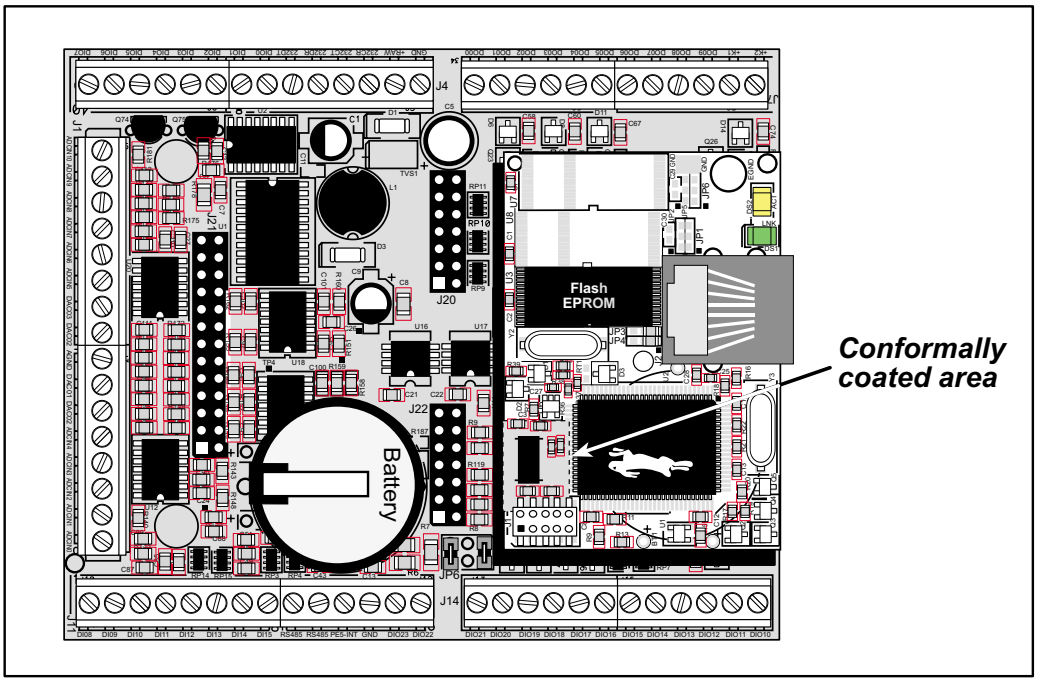



*Figure A-3. BL2100 Areas Receiving Conformal Coating*

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 BL2100 options via jumpers.

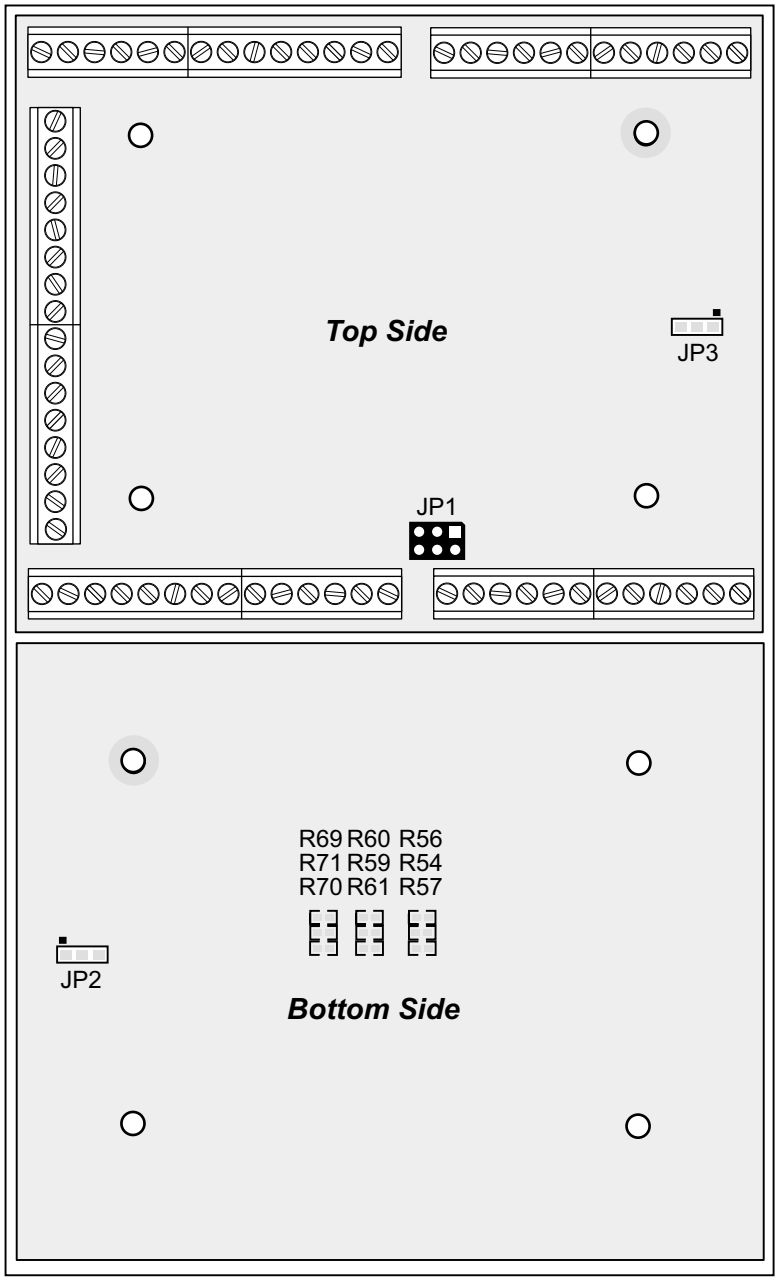



***Figure A-4. Location of BL2100 Configurable Positions***

Table A-3 lists the configuration options.

***Table A-3. BL2100 Jumper Configurations***

| Header | Description | Pins Connected | | Factory Default |
|---|---|---|---|---|
| JP1 | RS-485 Bias and Termination Resistors | 1–2<br>5–6 | Bias and termination resistors connected | ✕ |
| | | 1–3<br>4–6 | Bias and termination resistors ***not*** connected* | |
| JP2 | Software I/O Configuration Option | 1–2 | Standard | ✕ |
| | | 2–3 | Custom (IN16–IN23 are configured as digital sinking outputs) | |
| JP3 | Analog Circuit Option | 1–2 | Installed | BL2100<br>BL2120 |
| | | 2–3 | Not installed | BL2110<br>BL2130 |
| — | IN00–IN07 | R56 | Pulled up to Vcc | ✕ |
| | | R57 | Pulled up to +K2 | |
| | | R54 | Pulled down | |
| — | IN08–IN15 | R60 | Pulled up to Vcc | ✕ |
| | | R61 | Pulled up to +K2 | |
| | | R59 | Pulled down | |
| — | IN16–IN23 | R69 | Pulled up to Vcc | ✕ |
| | | R70 | Pulled up to +K2 | |
| | | R71 | Pulled down | |

* Although pins 1–3 and 4–6 of header JP1 are shown “jumpered” 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.

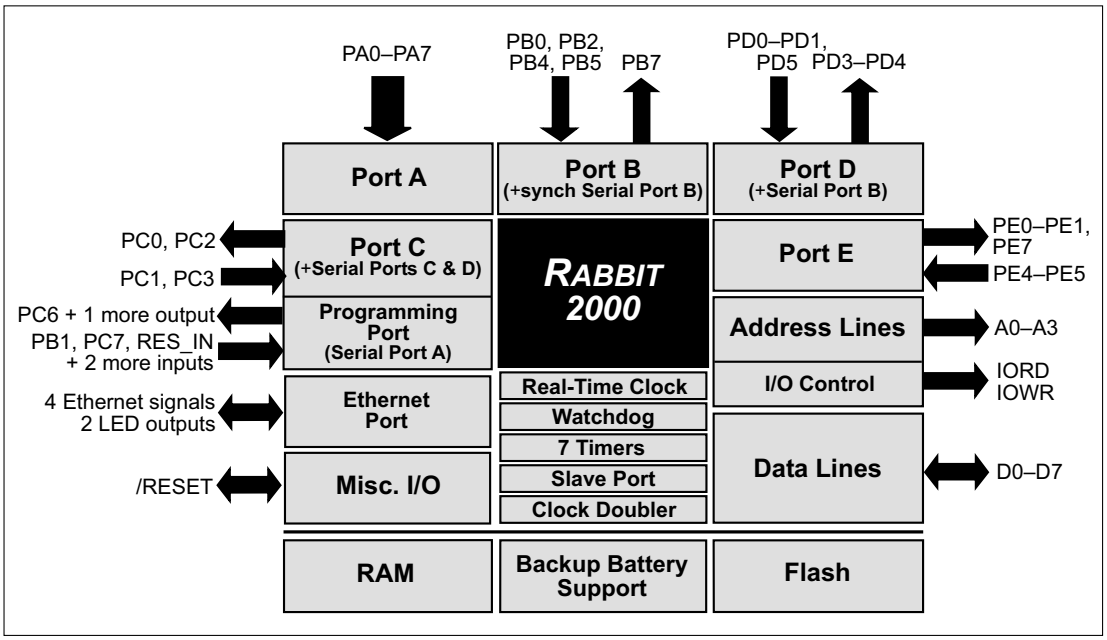



***Figure A-5. BL2100 Rabbit-Based Subsystems***

Table A-4 lists the Rabbit 2000 parallel ports and their use in the BL2100.

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

| Port | I/O | Signal | Output Function State |
|---|---|---|---|
| PA0 | Input | IN16 | Pulled up |
| PA1 | Input | IN17 | Pulled up |
| PA2 | Input | IN18 | Pulled up |
| PA3 | Input | IN19 | Pulled up |
| PA4 | Input | IN20 | Pulled up |
| PA5 | Input | IN21 | Pulled up |
| PA6 | Input | IN22 | Pulled up |
| PA7 | Input | IN23 | Pulled up |
| PB0 | Input | DAC_ADC_SDO | Pulled up |
| PB1 | Input | Not Used | Pulled up |
| PB2 | Input | ADC_EOC | Driven by A/D converter |
| PB3 | Input | Not Used | Pulled up |

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

| Port | I/O | Signal | | Output Function State |
|---|---|---|---|---|
| PB4 | Input | I/O Configuration Option (header JP2) | | 1 = standard (JP2:1–2)<br>0 = custom* (JP2:2–3) |
| PB5 | Input | Analog Circuit Option (header JP3) | | 1 = BL2100/BL2120 (JP3:1–2)<br>0 = BL2110/BL2130 (JP3:2–3) |
| PB6 | Output | Not Used | | Off |
| PB7 | Output | DAC_ADC_SDI | | Inactive high |
| 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 **sock_init** |
| PC5 | Input | TPOUT+ (Realtek INT0) | | Pulled up |
| PC6 | Output | TXA Programming Port | Serial Port A | Inactive high |
| PC7 | Input | RXA Programming Port | | Inactive high |
| PD0 | Input | Realtek CLK | | Initialized by **sock_init** |
| PD1 | Input | Realtek SDO | | Initialized by **sock_init** |
| PD2 | Output | Not used | | Inactive high |
| PD3 | Output | DAC CLK Line | | Inactive high |
| PD4 | Output | ATXB RS-232 | Serial Port B | Inactive high |
| PD5 | Input | ARXB RS-232 | | Inactive high |
| PD6 | Output | Not used | | Inactive high |
| PD7 | Output | Not used | | Inactive high |
| PE0 | Output | Digital I/O strobe | | Inactive high |
| PE1 | Output | External I/O enable | | Inactive high |
| PE2 | N/A | Realtek IORB strobe | | Initialized by **sock_init** |
| PE3 | N/A | Realtek SDI line | | Initialized by **sock_init** |
| PE4 | Input | INT0B | | Tied to PE5 by 1 kΩ resistor |
| PE5 | Input | INT1B | | User interrupt input† |
| PE6 | N/A | Realtek IOWB strobe | | Initialized by **sock_init** |
| PE7 | Output | LCD_KEYPAD strobe | | Inactive high |

* IN16–IN23 are sinking outputs in this custom configuration

† PE5 is driven by PE4 if the interrupt is not being used.

## A.5 I/O Address Assignments

Table A-5 lists the external I/O addresses for the digital inputs and outputs.

***Table A-5. Digital I/O Addresses***

| External Address | Name | Function |
|---|---|---|
| 0000 | **DIPA** | Digital inputs IN00–07, read only |
| 0001 | **DOPA** | Digital outputs OUT00–OUT07, write only |
| 0002 | **DIPB** | Digital inputs IN08–15, read only |
| 0003 | **DOPB** | Digital outputs OUT08–OUT15, write only |

PE1 serves as a system-enable control. When PE1 is high or in a high-impedance status, all BL2100 outputs are disabled (digital outputs and analog outputs are disabled, and RS-485 is at listen status).

PE0 is configured as a strobe and is used for digital inputs, digital outputs, and the control register. The control register is located at 0xx4–0xx7, write only. The function of each bit is listed in Table A-6.

***Table A-6. Control Register Bit Map (External 0x0004–0x007)***

| Bit | Name | Function |
|---|---|---|
| 0 | **485_SEND** | RS-485 send/receive |
| 1 | **DO_CS0** | Digital output 0–08, enable low active |
| 2 | **DO_CS1** | Digital output 09–16, enable low active |
| 3 | Not used | Not used |
| 4 | **AO_CS** | Analog output 00–04, enable low active |
| 5 | **DAC_CS0** | Chip select for analog ch 00 and 01 |
| 6 | **DAC_CS1** | Chip select for analog ch 02 and 03 |
| 7 | **ADC_CS** | Chip select for A/D converter |

PA0–PA7 are used with IN16–IN23, which may be reconfigured as sinking digital outputs OUT16–OUT23 by installing/removing components as reflected in the schematic.

All analog inputs and outputs are accessed by a series connection. PD3 is served as a clock line while PB0 and PB7 are used for data in and data out, respectively.

PD4 and PD5 are used for RS-485 communication. The direction of the communication is controlled by the control register. PC0, PC1, and PC2, PC3 are used for RS-232 communication. They can be used separately as two 3-wire RS-232, or they may be combined to work as a 5-wire RS-232 port.

# APPENDIX B. POWER SUPPLY

Appendix B describes the power circuitry provided on the BL2100.

## B.1 Power Supplies

Power is supplied to the BL2100 via pins 1 and 2 of screw-terminal header J4. The BL2100 is protected against reverse polarity by a diode at D1 as shown in Figure B-1.

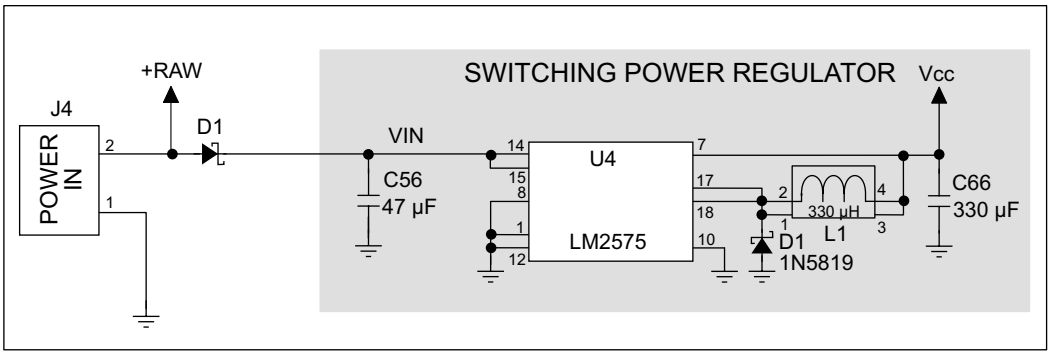



***Figure B-1. BL2100 Power Supply***

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 BL2100 logic circuits. Vcc is not accessible to the user.

The digital ground and the analog ground share a single split ground plane on the board, with the analog ground connected at a single point to the digital ground by a 0 Ω resistor (R29). This is done to minimize digital noise in the analog circuits and to eliminate the possibility of ground loops. External connections to analog ground are made on screw-terminal header J1, and external connections to digital ground are made on screw-terminal headers J4 and J11.

### B.1.1 Power for Analog Circuits

Power to the analog circuits is provided by way of a two-stage low-pass filter, which isolates the analog section from digital noise generated by the other components. The analog power voltage +V powers the op-amp for the buffered A/D converter inputs, the A/D converter, the D/A converter, and the 4.096 V reference circuit. The maximum current draw on +V is less than 10 mA. +V is not accessible to the user.

## B.2 Batteries and External Battery Connections

The SRAM and the real-time clock have 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 BL2100 is powered or not. When the BL2100 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 will come from the battery. The reset generator circuit controls the source of power by way of its **/RESET** output signal.

A replaceable 265 mA·h lithium battery provides power to the real-time clock and SRAM when external power is removed from the circuit board. The drain on the battery is typically less than 10 µA when there is no external power applied to the BL2100, and so the expected shelf life of the battery is

$$\frac{265\ \text{mA}\cdot\text{h}}{10\ \mu\text{A}} = 3.0\ \text{years.}$$

The drain on the battery is typically less than 4 µA when external power ***is*** applied, and so the expected BL2100 battery in-service life is

$$\frac{265\ \text{mA}\cdot\text{h}}{4\ \mu\text{A}} = 7.5\ \text{years.}$$

A long-life 950 mA·h solder-in battery is also provided for in the board layout.

### B.2.1 Replacing the Backup Battery

The battery is user-replaceable, and is fitted in a battery holder. To replace the battery, lift up on the spring clip and slide out the old battery. Use only a Panasonic CR2330 or equivalent replacement battery, and insert it into the battery holder with the + side facing up.

**NOTE:** The SRAM contents and the real-time clock settings will be lost if the battery is replaced with no power applied to the BL2100. Excercise care if you replace the battery while external power is applied to the BL2100.

**CAUTION:** There is an explosion danger if the battery is short-circuited, recharged, or replaced incorrectly. Replace the battery only with the same type or an equivalent type recommended by the battery manufacturer. Dispose of used batteries according to the battery manufacturer's instructions.

### B.2.2 Battery-Backup Circuit

Figure B-2 shows the battery-backup circuit  located on the BL2100 module.

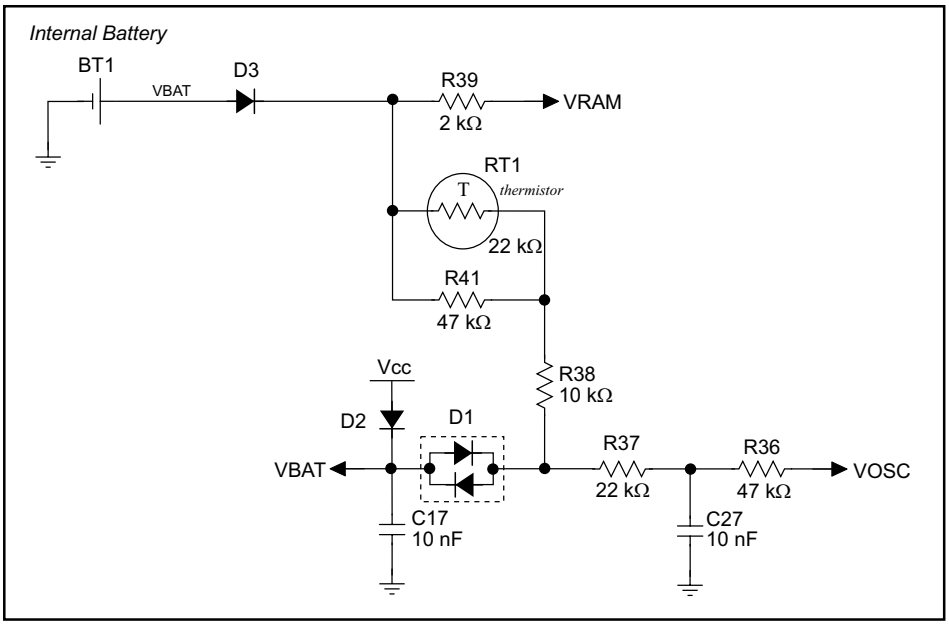



***Figure B-2. BL2100 Backup Battery Circuit***

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 BL2100.

### B.2.3 Power to VRAM Switch

The VRAM switch on the BL2100 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.

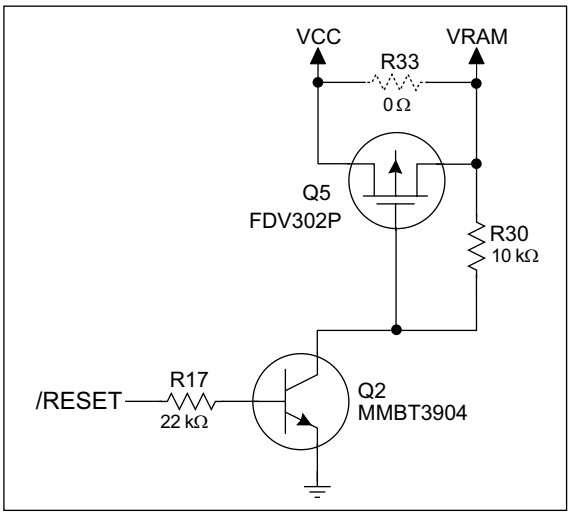



***Figure B-3. VRAM Switch***

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 BL2100 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 BL2100 ***is*** in reset, the **/RESET** line will go low. This turns off Q2 and Q5, providing an isolation between Vcc and VRAM.

### B.2.4 Reset Generator

The BL2100 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 BL2100 module.

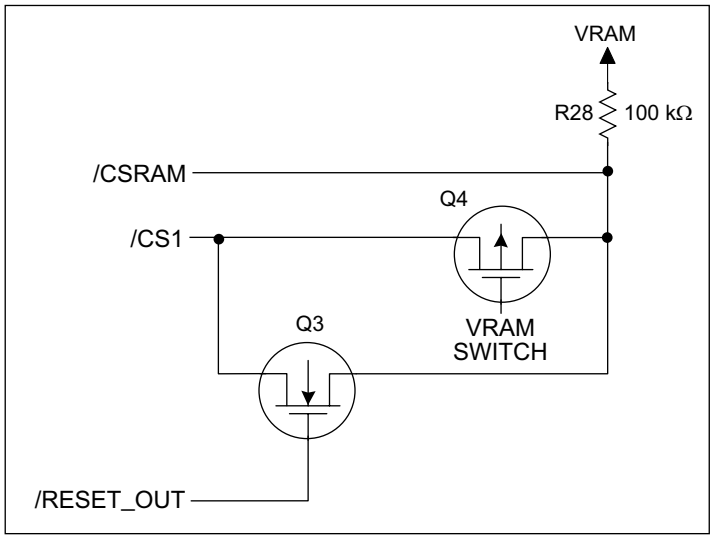



***Figure B-4. Chip Select Circuit***

The current drain on the battery in a battery-backed circuit must be kept at a minimum. When the BL2100 is not powered, the battery keeps the SRAM memory contents and the real-time clock (RTC) going. The SRAM has a powerdown mode that greatly reduces power consumption. This powerdown mode is activated by raising the chip select (CS) signal line. Normally the SRAM requires Vcc to operate. However, only 2 V is required for data retention in powerdown mode. Thus, when power is removed from the circuit, the battery voltage needs to be provided to both the SRAM power pin and to the CS signal line. The CS control circuit accomplishes this task for the 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. LCD/KEYPAD MODULE

An optional LCD/keypad is available for the BL2100. Appendix C describes the LCD/keypad and provides the software APIs to make full use of the LCD/keypad.

## C.1 Specifications

Table C-1 lists the electrical, mechanical, and environmental specifications for the LCD/keypad moduel.

***Table C-1. LCD/Keypad Specifications***

| Parameter | Specification |
|---|---|
| Board Size | 2.60" × 3.00" × 0.75"<br>(66 mm × 76 mm × 19 mm) |
| Temperature | Operating Range: 0°C to +50°C<br>Storage Range: –40°C to +85°C |
| Humidity | 5% to 95%, noncondensing |
| Power Consumption | 1.5 W maximum |
| Connections | Connects to high-rise header sockets on BL2100 |
| LCD Panel Size | 122 × 32 graphic display |
| Keypad | 7-key keypad |
| LEDs | Seven user-programmable LEDs |

## C.2 Mounting LCD/Keypad Module on the BL2100

Finish making any connections involving the analog I/O on screw-terminal header J1 before you install the LCD/keypad module since the LCD/keypad module will block access to the screws on screw-terminal header J1.

Install the LCD/keypad module on header sockets J20, J21, and J22 of the BL2100 main board as shown in Figure C-1. Be careful to align the pins over the headers, and do not bend them as you press down to mate the LCD/keypad module with the BL2100 main board.

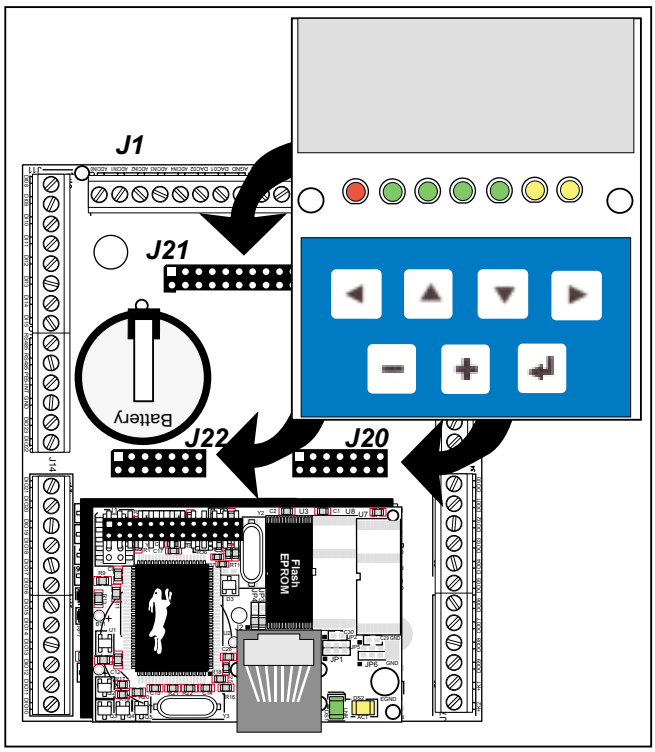



***Figure C-1. Install LCD/Keypad Module on BL2100 Main Board***

## C.3 Keypad Labeling

The keypad may be labeled according to your needs. A template is provided in Figure C-2 to allow you to design your own keypad label insert.

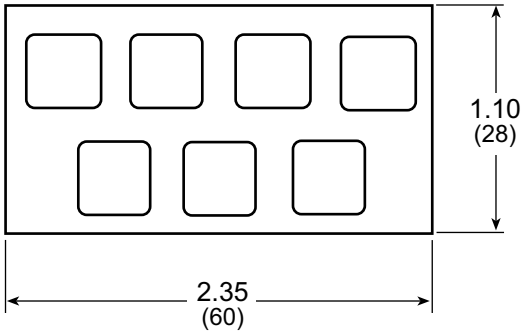



***Figure C-2. Keypad Template***

To replace the keypad legend, remove the old legend and insert your new legend prepared according to the template in Figure C-2. The keypad legend is located under the blue keypad matte, and is accessible from the left only as shown in Figure C-3.

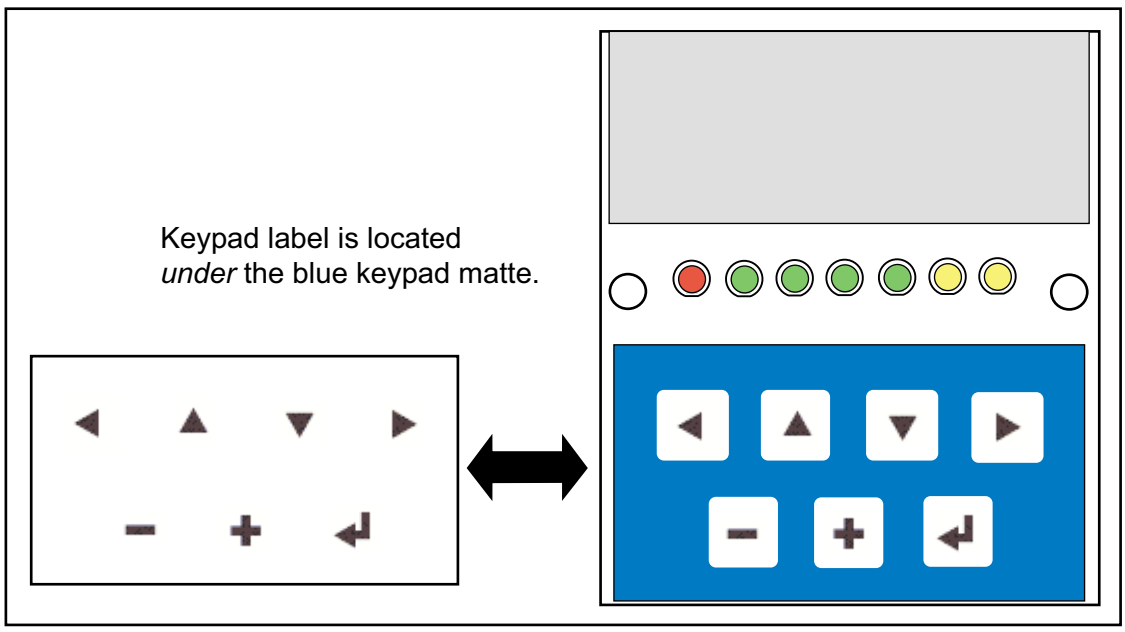



***Figure C-3. Removing and Inserting Keypad Label***

## C.4 Header Pinouts

Figure C-4 shows the pinouts for the LCD/keypad module.

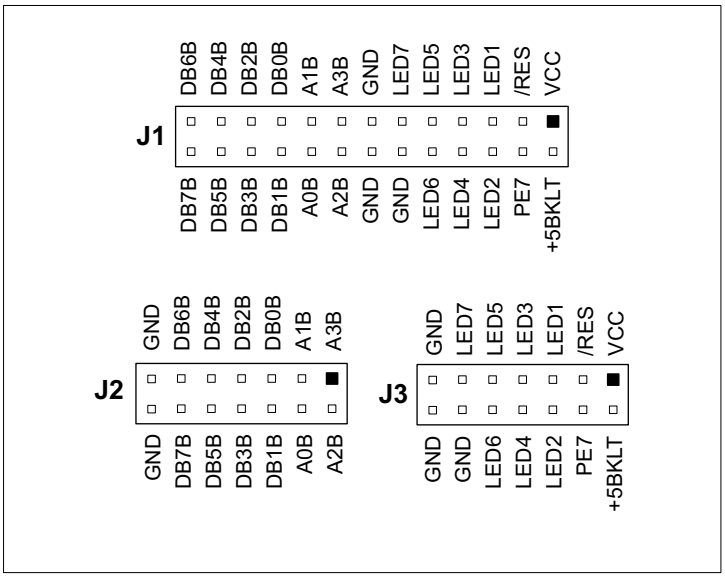



***Figure C-4. LCD/Keypad Module Pinouts***

### C.4.1 I/O Address Assignments

The LCD and keypad on the LCD/keypad module are addressed by the PE7 strobe as explained in Table C-2.

***Table C-2. LCD/Keypad Module Address Assignment***

| Address | Function |
|---|---|
| Exx0–Exx7 | LCD control |
| Exx8 | LED enable |
| Exx9 | Not used |
| ExxA | 7-key keypad |
| ExxB (bits 0–6) | 7-LED driver |
| ExxB (bit 7) | LCD backlight on/off |
| ExxC–ExxF | Not used |

## C.5 Programming Cable Tips

Once the LCD/keypad module is in place on the BL2100, it is not possible to remove or attach the programming cable to/from the BL2100 programming port. You will have to remove, or at least lift up, the LCD/keypad module while you connect or disconnect the programming cable.

While you are developing your application, you may wish to connect or disconnect the programming cable when resetting the BL2100 and switching between the Program Mode and the Run Mode. To avoid the inconvenience of removing and replacing the LCD/keypad module each time, the programming cable may be disconnected/reconnected at the RS-232/CMOS level converter in the middle of the programming cable.

1. Peel back plastic shrink wrap as shown in Figure C-5.

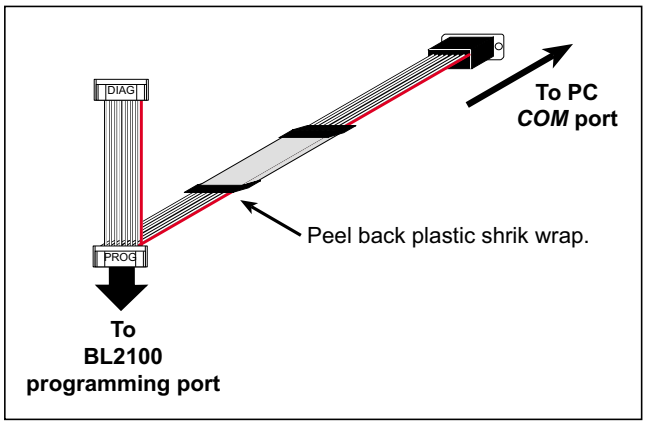



***Figure C-5. Peel Back Plastic Shrink Wrap***

2. Disconnect the programming cable at RS-2332/CMOS level converter board.

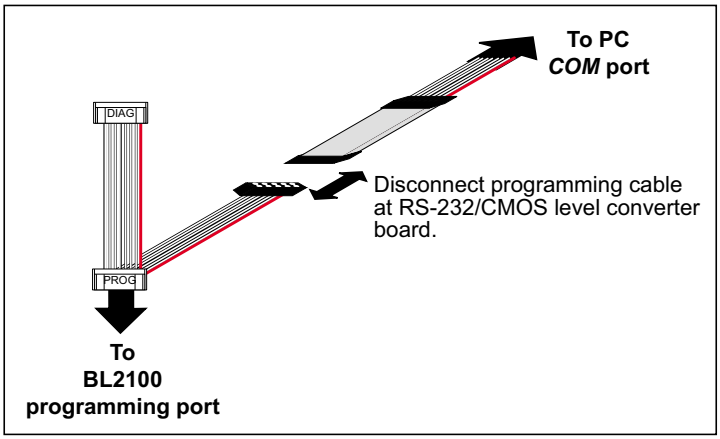



***Figure C-6. Disconnect Programming Cable***

3. Line up the colored edges of the programming cable when reconnecting the programming cable. Reconnect the programming cable as shown in Figure C-7, being careful to align the pins with the jack

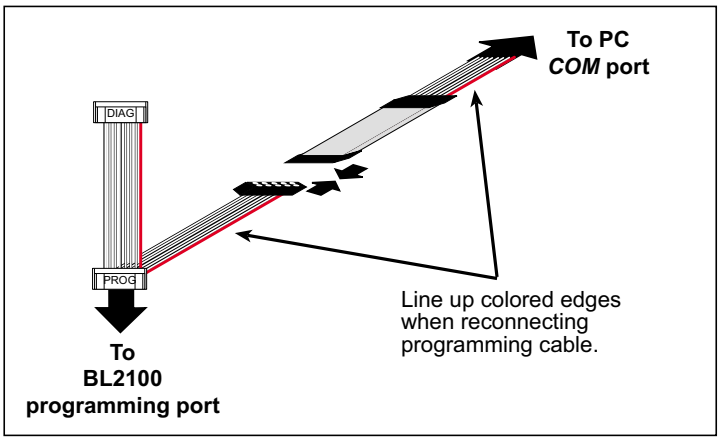



***Figure C-7. Reconnect Programming Cable***

Once you have finished programming the LCD/keypad module, you should disconnect the programming cable from the BL2100 programming port, remembering to first remove, or at least lift up, the LCD/keypad module, disconnect the programming cable, and finally firmly mount the LCD/keypad module back on the BL2100 main board.

## C.6 LCD/Keypad Module Function APIs

### C.6.1 LEDs

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

One function is available to control the LEDs, and can be found in the **BL21XX.LIB** library.

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

LED on/off control. This function will only work when the LCD/keypad module is installed on the BL2100.

**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**

### C.6.2 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 backlight 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 BL2100).

**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.

**bmWidth** 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

**glPrinf**

## 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, doprnt**

## 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, TextCursorLocation**

### C.6.3 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 × 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 × 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 followingfor 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**

## C.7 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.

The following sample programs, found in the **TCPIP** subdirectory in **SAMPLES/LCD_Keypad/122x32_1x7**, are targeted at the Ethernet-enabled versions of the BL2100, the BL2100 and the BL2110. Remember to configure the IP address, net-mask, and gateway as indicated in the sample programs.

- **MBOXDEMO.C**—This program implements a web server that allows 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 BL2100 and a PC (or other device with a Web Browser) to an Ethernet. If you connect the PC and the BL2100 directly, be sure to use a crossover Ethernet cable; strait-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 BL2100 and the other single-board computer to an Ethernet. If you connect the them directly, be sure to use a crossover Ethernet cable; strait-through Ethernet cables and a hub may be used instead.

# APPENDIX D. PLASTIC ENCLOSURE

The plastic enclosure provides a secure way to protect your BL2100. The enclosure itself may be mounted on any flat surface.

The complete plastic enclosure consists of a base and a cover. The base alone is a convenient surface on which to mount the BL2100, and also provides a means to mount the BL2100 on any flat surface. The base and cover are sold together with an LCD/keypad module that plugs into the main BL2100 board.

Appendix D describes how to mount the BL2100 and the LCD/keypad inside the plastic enclosure, and provides details on mounting the assembly.

## D.1 Assembly Instructions

1. Remove the RabbitCore module from the BL2100 main board, and set the module aside. The module will be plugged back in to the main board later.

   **NOTE:** If you are working with more than one BL2100 at a time, take care to keep the BL2100 main boards and their corresponding RabbitCore modules paired since the RabbitCore modules store calibration constants specific to the BL2100 main board to which they are plugged in.

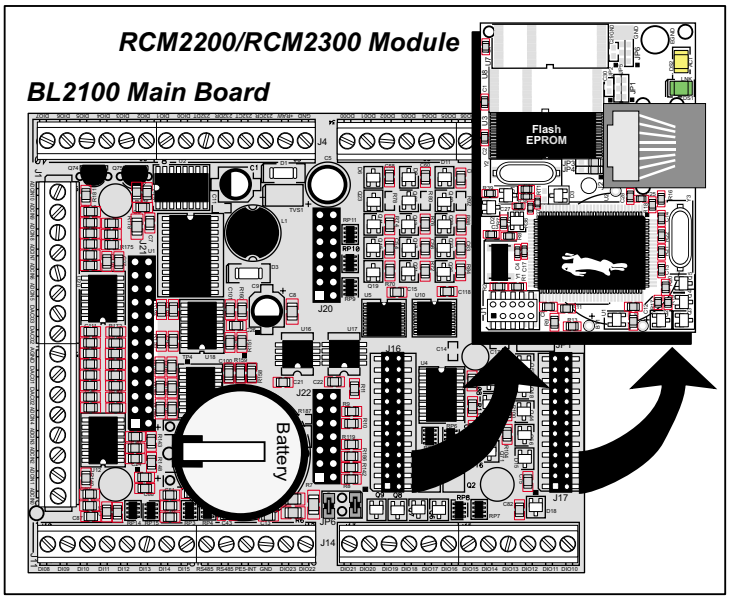



***Figure D-1. Remove RCM2200 Module from BL2100 Main Board***

2. Attach the BL2100 main board to the plastic enclosure base.

Position the BL2100 main board over the plastic enclosure base as shown below in Figure D-2. Attach the BL2100 to the base using the four 4-40 × ¼ screws supplied with the enclosure base.

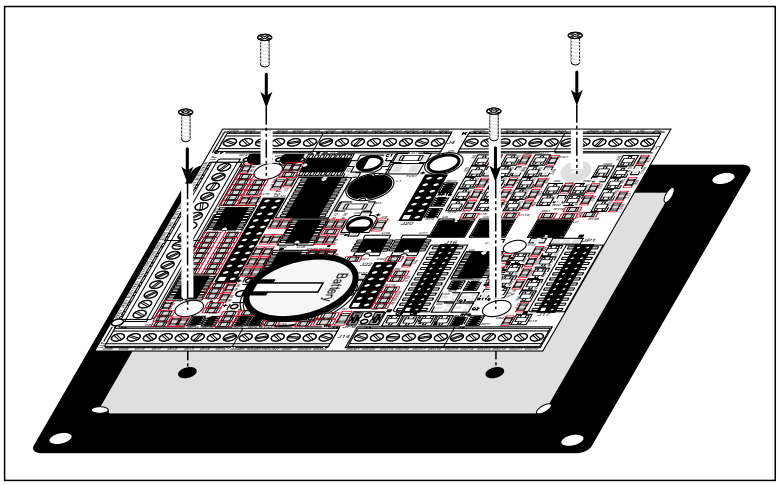


***Figure D-2. Attach BL2100 to Plastic Enclosure Base***

3. Reconnect the RabbitCore module to headers J16 and J17 on the BL2100 main board as shown in Figure D-3. Be careful to align the pins over the headers, and do not bend them as you press down to mate the module with the BL2100 main board.

   **NOTE:** If you are working with more than one BL2100 at a time, take care to keep the BL2100 main boards and their corresponding RabbitCore modules paired since the RabbitCore modules store calibration constants specific to the BL2100 main board to which they are plugged in.

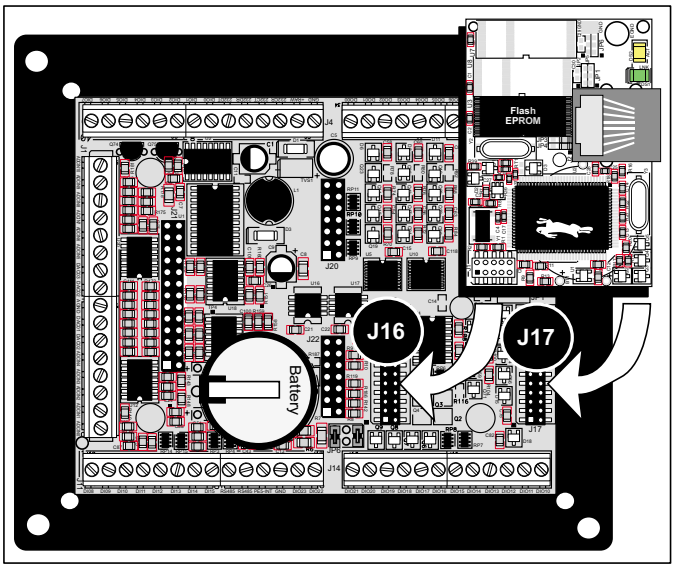



***Figure D-3. Reconnect Module to BL2100 Main Board***

4. Install the LCD/keypad module (optional) as shown in Figure D-4. Be careful to align the pins over the headers, and do not bend them as you press down to mate the LCD/keypad module with the BL2100 main board.

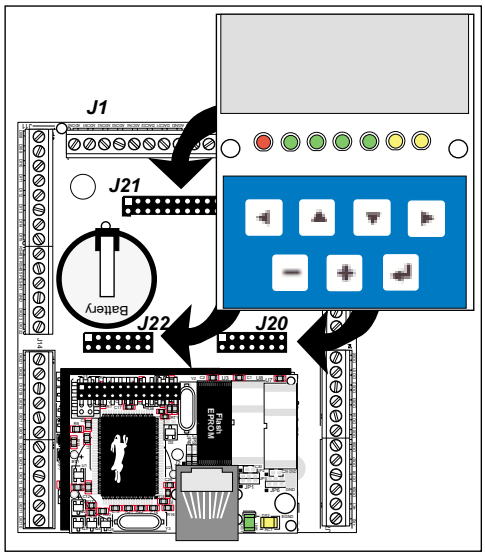



***Figure D-4. Install LCD/Keypad Module on BL2100 Main Board***

5. Mount plastic enclosure (optional).

Use four #10 screws to attach the plastic enclosure at the four outer corner mounting holes to the surface on which it will be mounted. This step applies to production versions of BL2100 units once development has been completed.

6. Attach the enclosure cover to the base.

Position the cover over the plastic enclosure base as shown below in Figure D-5. Attach the cover to the base using the four 4-40 × 7/8 screws supplied.

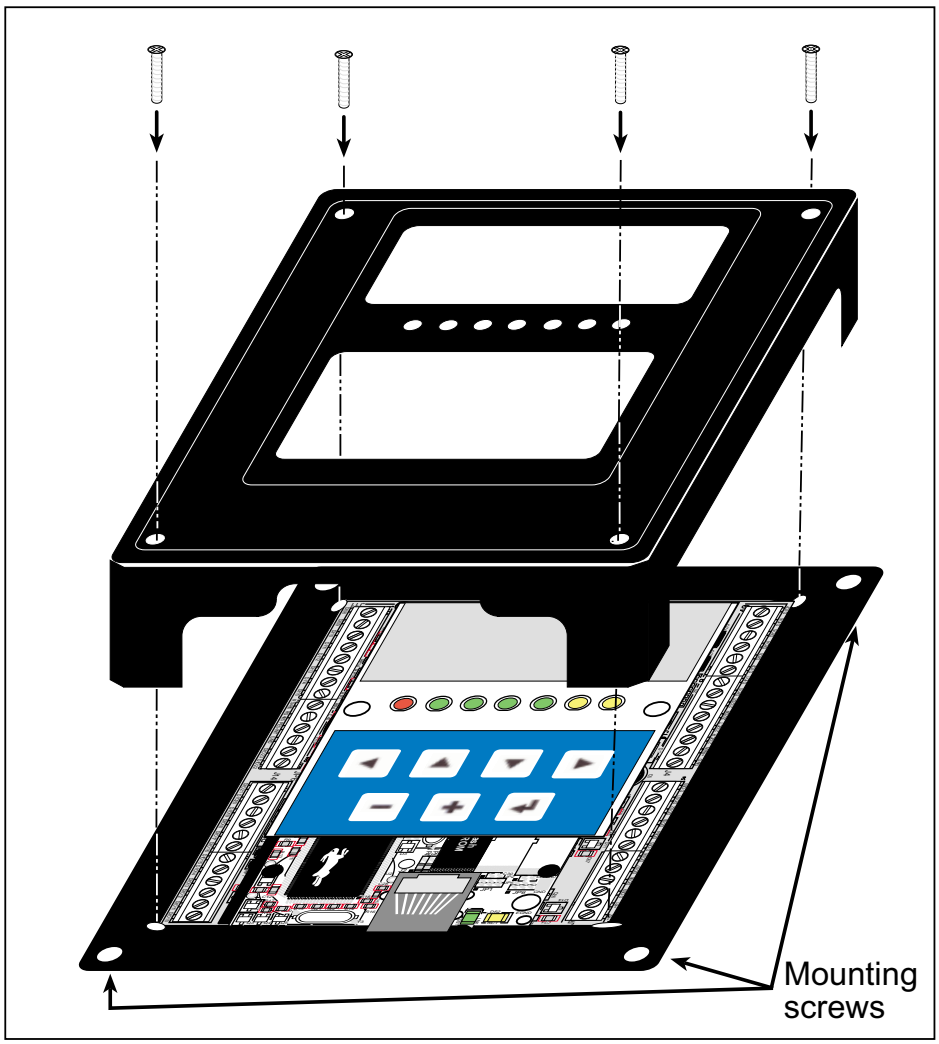



***Figure D-5. Attach Enclosure Top***

## D.2 Dimensions

Figure D-6 shows the dimensions for the plastic enclosure.

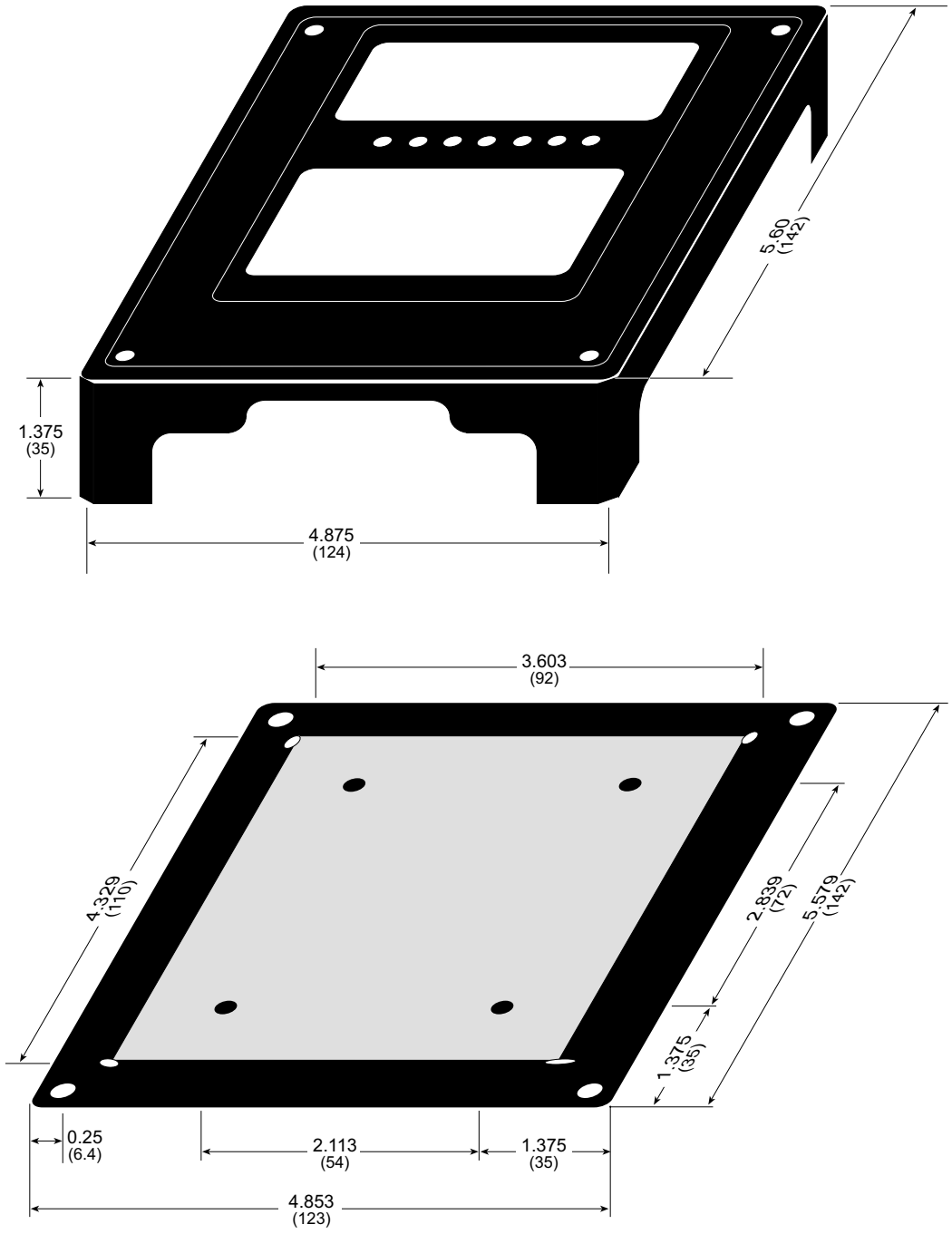



***Figure D-6. Plastic Enclosure Dimensions***

When fully assembled with the BL2100 and the LCD/keypad module installed, the total height of the plastic enclosure will be 1.5" (38 mm).

# APPENDIX E. DEMONSTRATION BOARD

Appendix E shows how to connect the Demonstration Board to the BL2100.

## E.1 Connecting Demonstration Board

Before running sample programs based on the Demonstration Board, you will have to connect the Demonstration Board from the BL2100 Tool Kit to the BL2100 board. Proceed as follows.

1. Use the wires included in the BL2100 Tool Kit to connect header J1 on the Demonstration Board to screw-terminal headers J4 and J7 on the BL2100. The connections are shown in Figure E-1 for sample program **`DIGIN.C`** and for sample program **`SMTP.C`**, in Figure E-2 for sample program **`DIGOUT.C`**, and in Figure E-3 for sample program **`SSI.C`**.
2. Make sure that your BL2100 is connected to your PC and that the power supply is connected to the BL2100 and plugged in as described in Chapter 2, “Getting Started.”

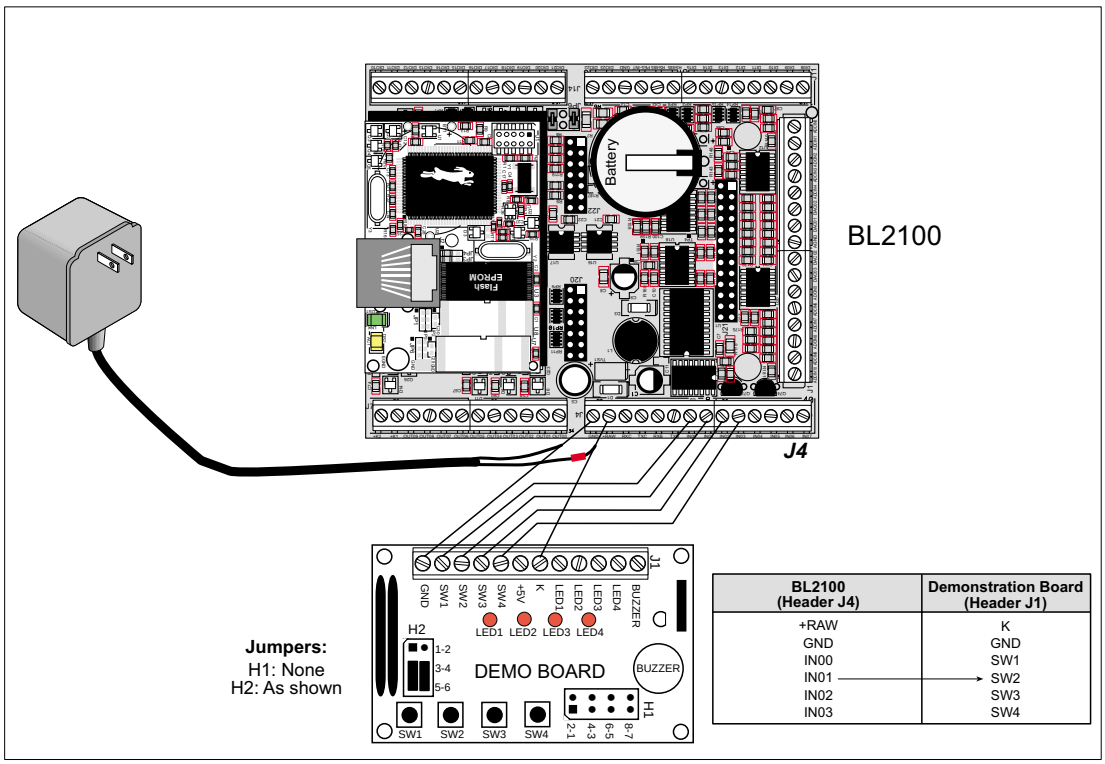



***Figure E-1. General Digital Input Connections Between BL2100 and Demonstration Board***

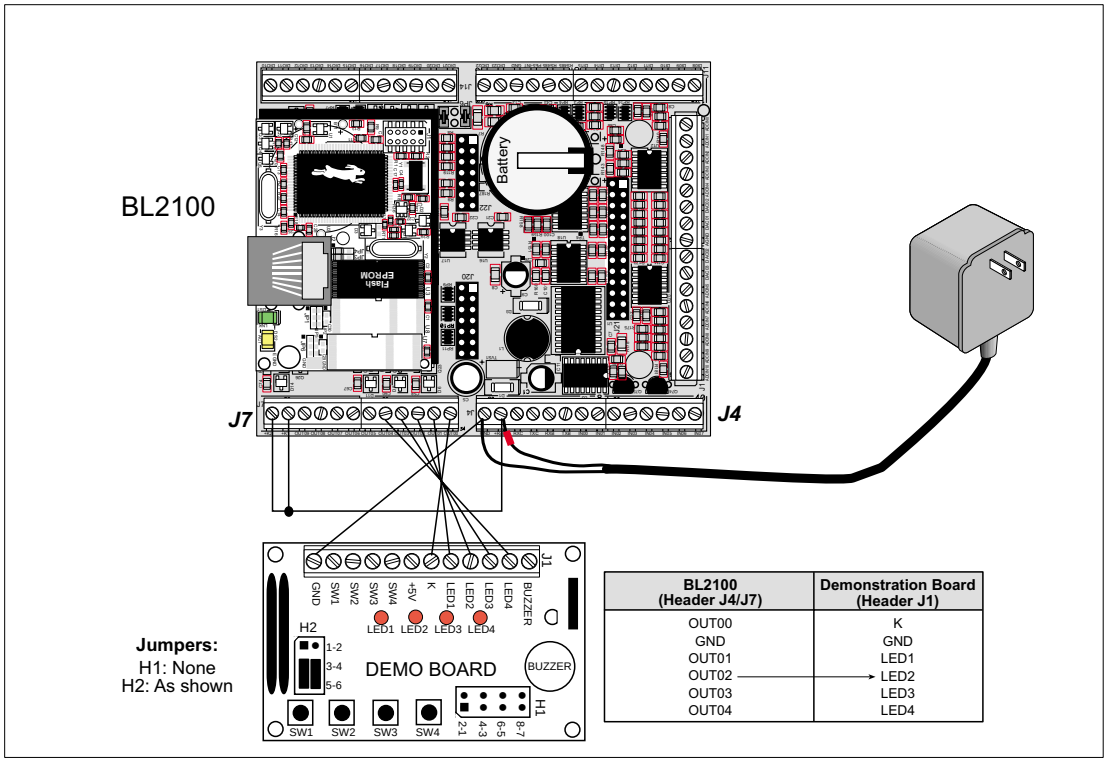



***Figure E-2. Digital Output Connections Between BL2100 and Demonstration Board***

**NOTE:** +K1 and +K2 on screw-terminal header J7 must be connected to +RAW on screw-terminal header J4 as shown in Figure E-2.

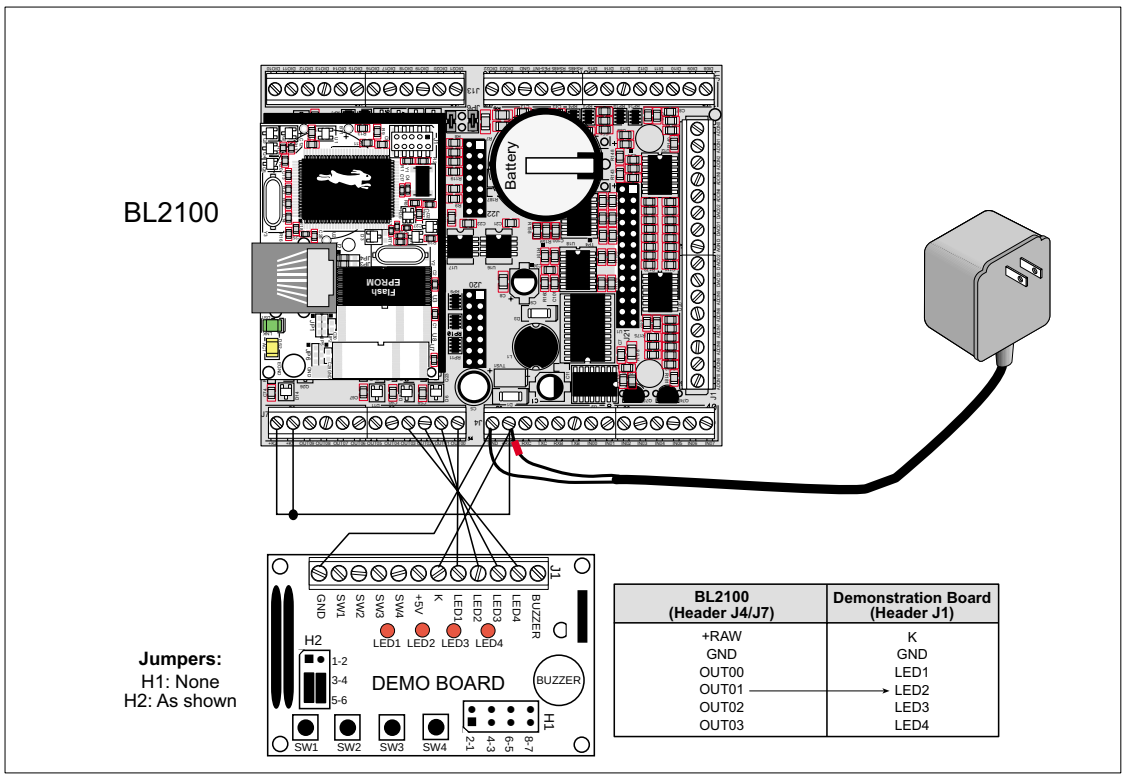



***Figure E-3.* `SSI.C` *Connections Between BL2100 and Demonstration Board***

**NOTE:** +K1 and +K2 on screw-terminal header J7 must be connected to +RAW on screw-terminal header J4 as shown in Figure E-3.

# APPENDIX F. PROGRAMMING CABLE

Appendix F 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 BL2100 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 BL2100 system while it is running.

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

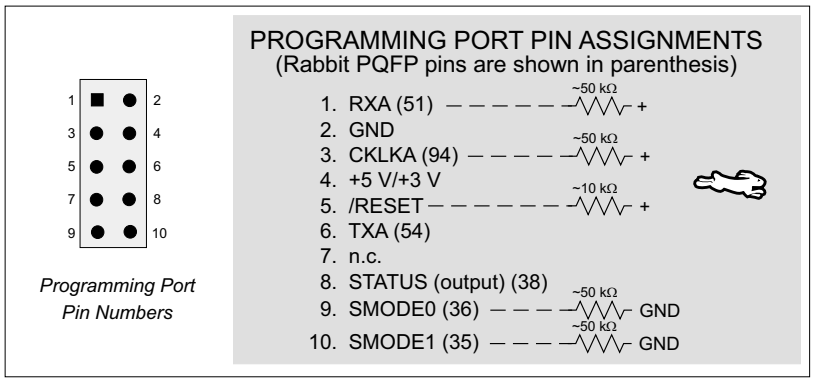



***Figure F-1. Programming Port Pin Assignments***

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.

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

***Table F-1. BL2100 Programming Port Configurations***

| Pin | Pin Name | Default Use | Alternate Use | Notes |
|---|---|---|---|---|
| 1 | RXA | Serial Port A | PC6—Input | |
| 2 | GND | | | |
| 3 | CLKA | | PB1—Bitwise or parallel programmable input | |
| 4 | VCC | | | |
| 5 | RESET | | | Connected to reset generator U1 on BL2100 module |
| 6 | TXA | Serial Port A | PC7—Output | |
| 8 | STATUS | | Output | |
| 9 | SMODE0 | | Input | Must be low when BL2100 boots up |
| 10 | SMODE1 | | Input | Must be low when BL2100 boots up |

# INDEX

## O



## P



## R



## S

## T

# SCHEMATICS

**090-0124 BL2100 Schematic**

**090-0120 RCM2200 Module Schematic**

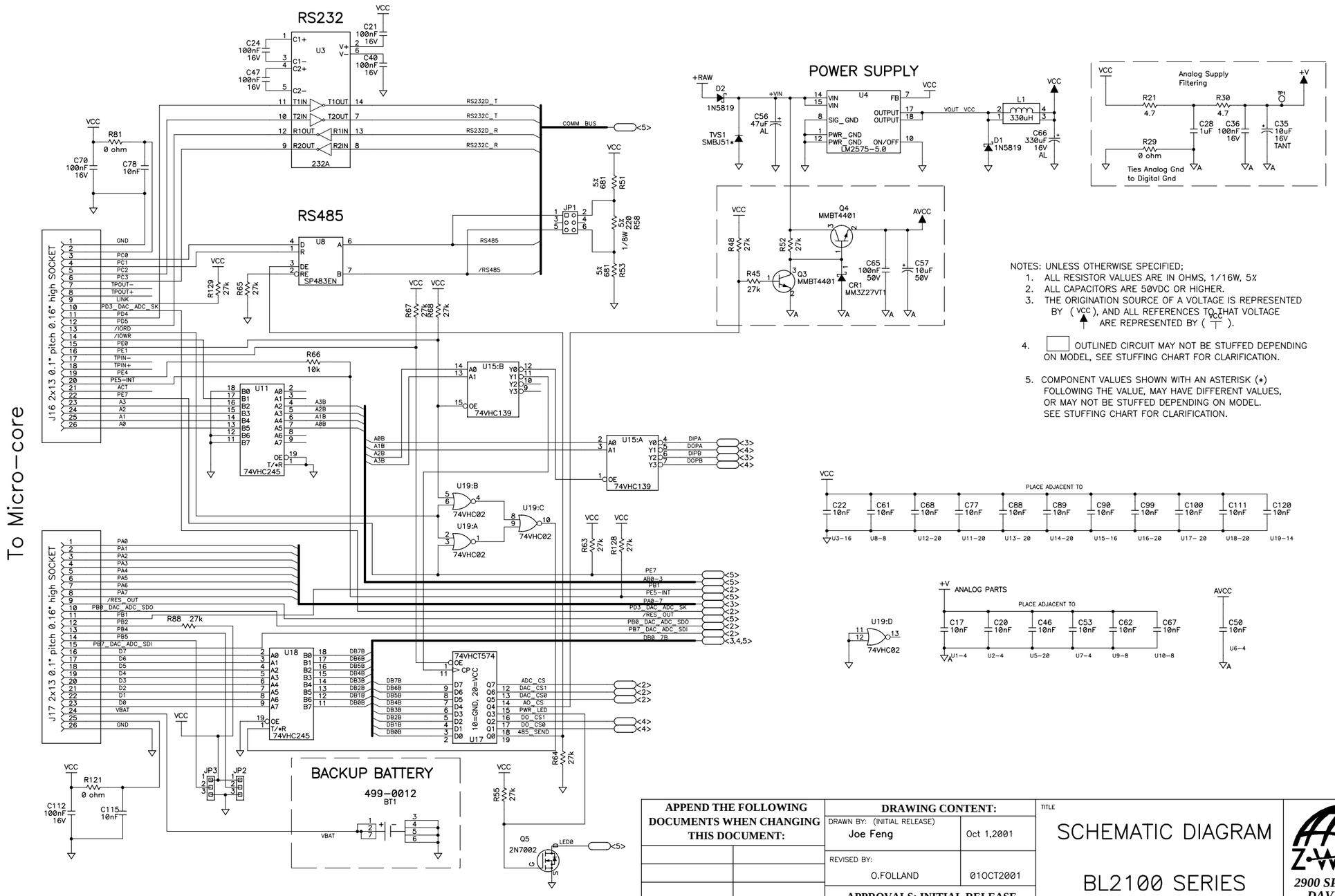
**090-0119 RCM2300 Module Schematic**

**090-0042 Demonstration Board Schematic**

**090-0125 LCD/Keypad Module Schematic**

**090-0128 Programming Cable Schematic**

| REVISION HISTORY | | | REVISION APPROVAL | | | |
|---|---|---|---|---|---|---|
| REV | ECO | DESCRIPTION OF CHANGE | PROJECT ENGINEER | APPROVAL DATE | DOCUMENT CONTROL | APPROVAL DATE |
| A | E11714 | INITIAL RELEASE | | | | |




NOTES: UNLESS OTHERWISE SPECIFIED;

1. ALL RESISTOR VALUES ARE IN OHMS, 1/16W, 5%
2. ALL CAPACITORS ARE 50VDC OR HIGHER.
3. THE ORIGINATION SOURCE OF A VOLTAGE IS REPRESENTED BY ( VCC ), AND ALL REFERENCES TO THAT VOLTAGE ARE REPRESENTED BY ( VCC ).
4. [ ] OUTLINED CIRCUIT MAY NOT BE STUFFED DEPENDING ON MODEL, SEE STUFFING CHART FOR CLARIFICATION.
5. COMPONENT VALUES SHOWN WITH AN ASTERISK (*) FOLLOWING THE VALUE, MAY HAVE DIFFERENT VALUES, OR MAY NOT BE STUFFED DEPENDING ON MODEL. SEE STUFFING CHART FOR CLARIFICATION.

| APPEND THE FOLLOWING DOCUMENTS WHEN CHANGING THIS DOCUMENT: | |
|---|---|
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |

| DRAWING CONTENT: | |
|---|---|
| DRAWN BY: (INITIAL RELEASE) Joe Feng | Oct 1,2001 |
| REVISED BY: O.FOLLAND | 01OCT2001 |
| **APPROVALS: INITIAL RELEASE** | |
| PROJECT ENGINEER: Joe Feng | Oct 1,2001 |
| ENGINEERING MANAGER: | |
| **SIGNATURES** | **DATE** |

TITLE

# SCHEMATIC DIAGRAM

## BL2100 SERIES


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


SIZE C | DWG NO. 090-0124

SCALE NONE | RELEASE DATE | SHEET 1 OF 6

AI_BUS
<5>

ADC00
ADC01
ADC02
ADC03
ADC04
ADC05
ADC06
ADC07
ADC08
ADC09
ADC10

ADREF

R6 1.00M 1%
R2 1.00M 1%
R13 1.00M 1%
R12 1.00M 1%
R11 1.00M 1%
R10 1.00M 1%
R9 1.00M 1%
R8 1.00M 1%
R5 1.00M 1%
R4 1.00M 1%
R3 1.00M 1%

C6 1nF
C1 1nF
C12 1nF
C11 1nF
C10 1nF
C9 1nF
C8 1nF
C7 1nF
C5 1nF
C4 1nF
C3 1nF

C14 100pF
C23 100pF
C18 100pF
C19 100pF
C52 100pF
C33 100pF
C51 100pF
C32 100pF
C16 100pF
C13 100pF
C15 100pF

R16 200K
R20 200K
R22 200K
R23 200K
R41 200K
R28 200K
R42 200K
R27 200K
R17 200K
R15 200K
R14 200K

U2:D
U2:A
U2:C
U2:B
U7:A
U7:D
U7:B
U7:C
U1:A
U1:D
U1:C
LMC660

U5
TLC2543
AIN0 DATA_OUT
AIN1 EOC
AIN2
AIN3
AIN4 I/O_CLK
AIN5 DATA_IN
AIN6 CS
AIN7
AIN8 REF+
AIN9
AIN10 REF−

16 PB0_DAC_ADC_SDO
19 PB1
18 PD3_DAC_ADC_SK
17 PB7_DAC_ADC_SDI
15 ADC_CS
14 4P096_VREF
13

TP4

ANALOG TO DIGITAL
11 CHANNELS
−10 V TO + 10 V

VCC
R112 27k
R104 27k
R117 27k

<1> DAC_CS0
<1> DAC_CS1
<1> ADC_CS
<1> PB7_DAC_ADC_SDI
<1> PD3_DAC_ADC_SK
<1> PB1
<1> PB0_DAC_ADC_SDO

U9
U10
DIN
SCLK
/CS
REF
OUTA
OUTB
TLV5618

R47 102K 1/16W 1%
R40 255K 1%
R35 86.6K 1%
R25 102K 1/16W 1%
R26 255K 1%
R33 86.6K 1%
R31 102K 1/16W 1%
R24 255K 1%
R32 86.6K 1%
R46 102K 1/16W 1%
R39 255K 1%
R34 86.6K 1%

C49 100pF
C31 100pF
C30 100pF
C48 100pF

U6:A
U6:D
U6:C
U6:B
LM324

DAC0
DAC1
DAC2
DAC3

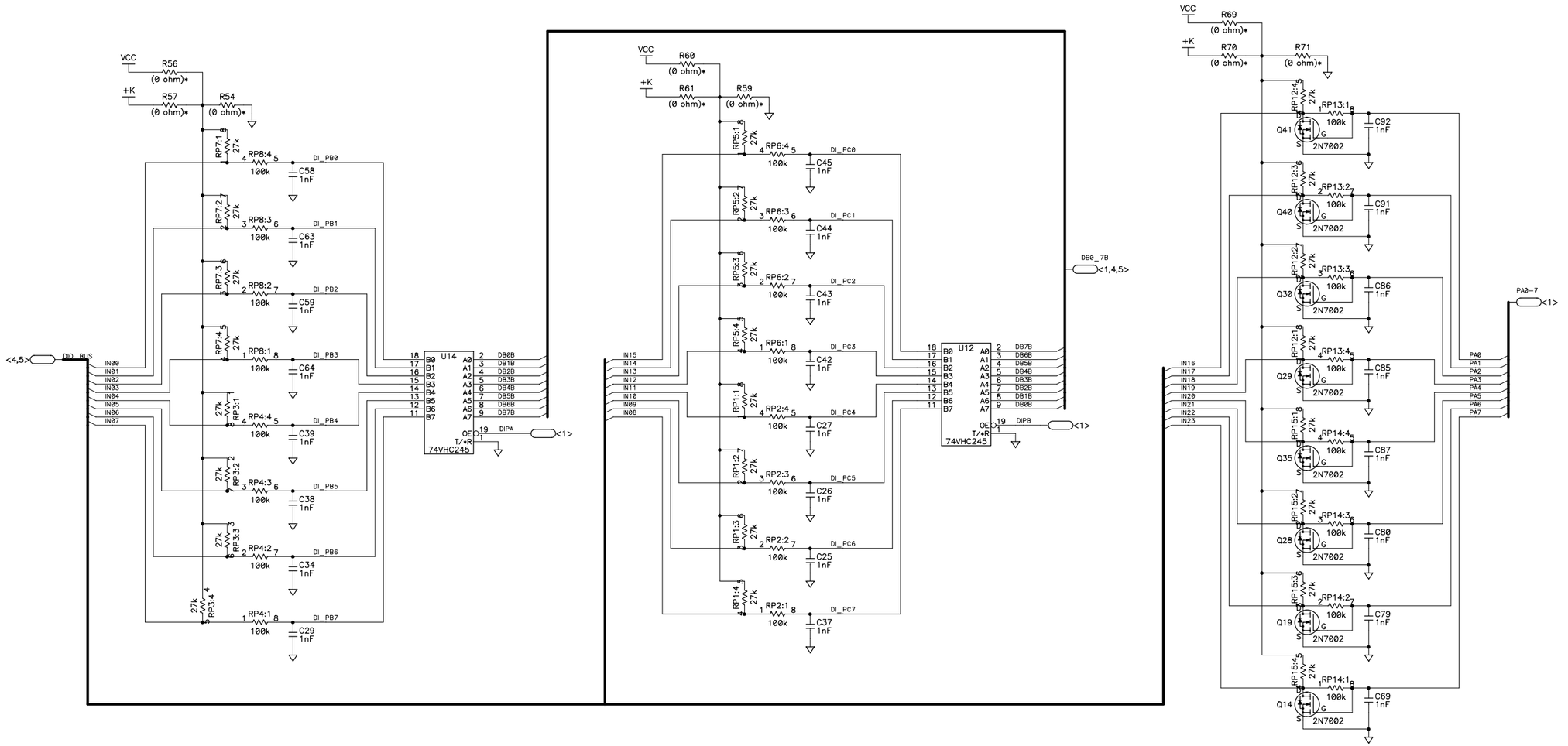
AO_BUS
<5>

+V
R49 390 ohm 1/16W
C55 100nF 16V
VR1 LM4040-4.096
4P096_VREF
R38 14.0K
R43 10.0K
C54 100nF 16V
R44 25.5K
R50 25.5K
C60 100nF 16V
R37 10.2K
R36 25.5K
C41 100nF 16V

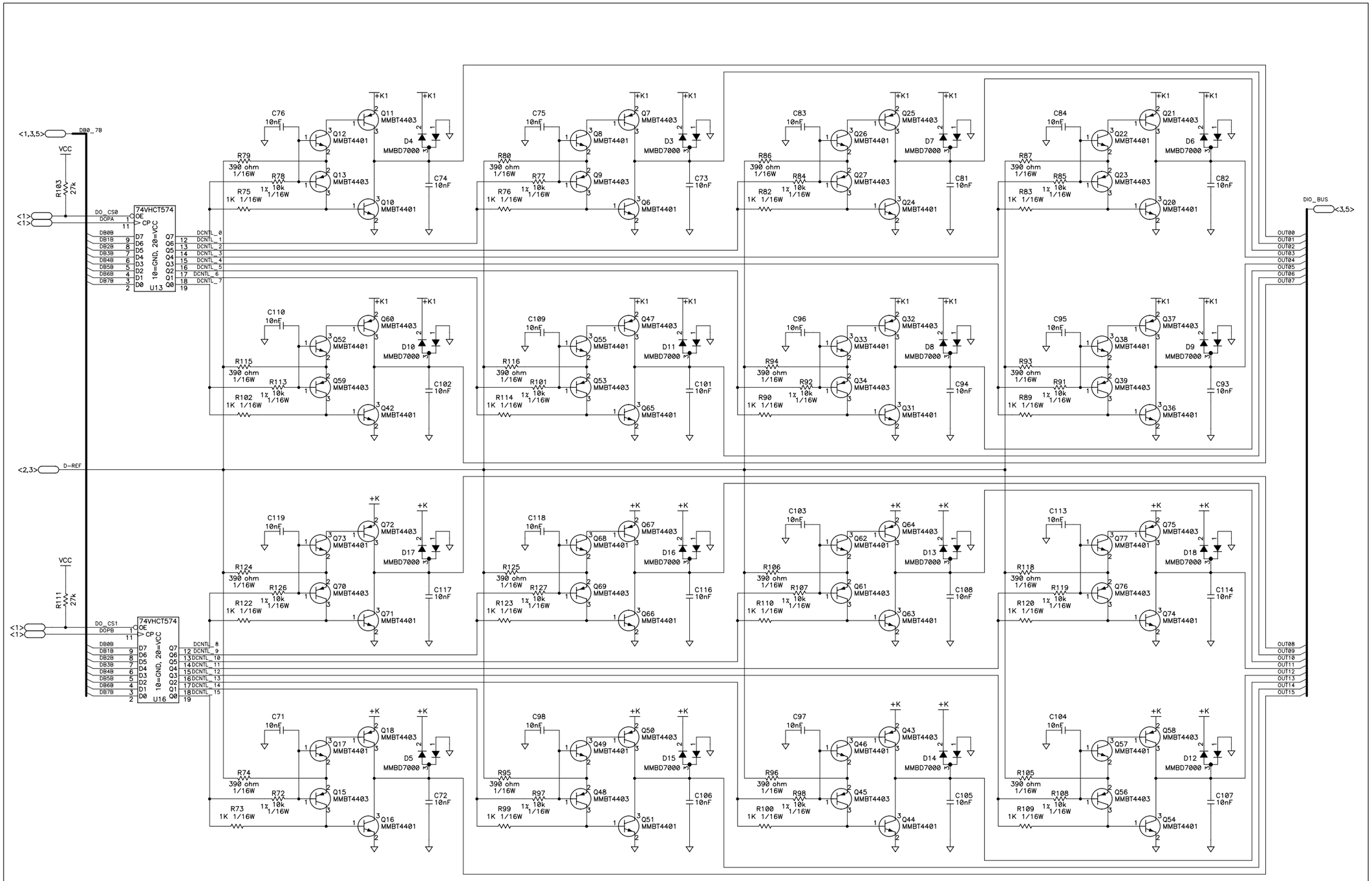
DIGITAL TO ANALOG
4 CHANNELS
0 TO +10 V

R7 25.5K
C2 10nF
R19 27k
VCC
Q1
MPSA14
MPSA63
Q2
R18 4.7
D−REF
<4>
U1:B
2P048_VREF
R1 102K 1%

| SIZE | DWG NO. | |
|---|---|---|
| C | 090−0124 | |
| SCALE NONE | REV LTR A | SHEET 2 OF 6 |




| SIZE | DWG NO. | |
|---|---|---|
| C | 090-0124 | |
| SCALE NONE | REV LTR A | SHEET 3 OF 6 |




| SIZE | DWG NO. | |
|---|---|---|
| C | 090-0124 | |
| SCALE NONE | REV LTR A | SHEET 4 OF 6 |

J6 6 POS 0.381" PITCH SCREW TERMINAL

J4 2 X 20 .100 INCH IDC HEADER (optional)

+RAW

J5 8 POS 0.381" PITCH SCREW TERMINAL

J9 6 POS 0.381" PITCH SCREW TERMINAL

J7 2 X 17 .100 INCH IDC HEADER (optional)

J7-2 6 POS 0.381" PITCH SCREW TERMINAL

+K1+K2

J2 8 POS 0.381" PITCH SCREW TERMINAL

J1 2 X 25 .100 INCH IDC HEADER

VCC

J22 2x7 POS 0.1" PITCH .635" HEIGHT HEADER

R62 390 1/16W /RES_OUT PE7

J21 2x13 POS 0.1" PITCH .635" HEIGHT HEADER

J20 2x7 POS 0.1" PITCH .635" HEIGHT HEADER

RP11:3 RP11:4 RP11:2 RP11:1 RP10:4 RP10:3 RP10:2 RP10:1 RP9:4 RP9:3 RP9:2 RP9:1 100

A3B A2B A1B A0B DB0B DB1B DB2B DB3B DB4B DB5B DB6B DB7B

AB0_3 DB0_7B

AO_BUS AI_BUS DIO_BUS COMM_BUS DIO_BUS

J3 8 POS 0.381" PITCH SCREW TERMINAL

J10 8 POS 0.381" PITCH SCREW TERMINAL

J11 2 X 20 .100 INCH IDC HEADER (optional)

J12 6 POS 0.381" PITCH SCREW TERMINAL

J14 6 POS 0.381" PITCH SCREW TERMINAL

J13 2 X 17 .100 INCH IDC HEADER (optional)

J15 6 POS 0.381" PITCH SCREW TERMINAL

| SIZE | DWG NO. | |
|---|---|---|
| C | 090-0124 | |
| SCALE NONE | REV LTR A | SHEET 5 OF 6 |

## STUFFING TABLE

| CIRCUIT | | PART | BL2100/2120 | BL2110/2130 |
|---|---|---|---|---|
| POWER INDICATION | | R55 | NOT INSTALLED | NOT INSTALLED |
| | | Q5 | NOT INSTALLED | NOT INSTALLED |
| CONFIGURATION IND. | | JP3 | 1 – 2 | 2 – 3 |
| OUTPUT PULL-UP/DOWN | OUTPUTS PULLED UP TO +K | R57 | NOT INSTALLED | NOT INSTALLED |
| | | R61 | NOT INSTALLED | NOT INSTALLED |
| | | R70 | NOT INSTALLED | NOT INSTALLED |
| | OUTPUTS PULLED UP TO +5V | R56 | ZERO ohm | ZERO ohm |
| | | R60 | ZERO ohm | ZERO ohm |
| | | R69 | ZERO ohm | ZERO ohm |
| | OUTPUTS PULLED DOWN TO GROUND | R54 | NOT INSTALLED | NOT INSTALLED |
| | | R59 | NOT INSTALLED | NOT INSTALLED |
| | | R71 | NOT INSTALLED | NOT INSTALLED |
| INPUT/OUTPUT DRIVERS 16–23 (CFG. AS INPUT) | | JP2 | 1 – 2 | 1 – 2 |
| | | PR13 | RES PACK 100Kx4 | RES PACK 100Kx4 |
| | | RP14 | RES PACK 100Kx4 | RES PACK 100Kx4 |
| | | Q14 | NOT INSTALLED | NOT INSTALLED |
| | | Q19 | NOT INSTALLED | NOT INSTALLED |
| | | Q28 | NOT INSTALLED | NOT INSTALLED |
| | | Q29 | NOT INSTALLED | NOT INSTALLED |
| | | Q30 | NOT INSTALLED | NOT INSTALLED |
| | | Q35 | NOT INSTALLED | NOT INSTALLED |
| | | Q40 | NOT INSTALLED | NOT INSTALLED |
| | | Q41 | NOT INSTALLED | NOT INSTALLED |
| DIGITAL OUTPUT 00 – 15 SETUP AS SCR & SINK | | R73 | NOT INSTALLED | NOT INSTALLED |
| | | R75 | NOT INSTALLED | NOT INSTALLED |
| | | R76 | NOT INSTALLED | NOT INSTALLED |
| | | R82 | NOT INSTALLED | NOT INSTALLED |
| | | R83 | NOT INSTALLED | NOT INSTALLED |
| | | R89 | NOT INSTALLED | NOT INSTALLED |
| | | R90 | NOT INSTALLED | NOT INSTALLED |
| | | R99 | NOT INSTALLED | NOT INSTALLED |
| | | R100 | NOT INSTALLED | NOT INSTALLED |
| | | R102 | NOT INSTALLED | NOT INSTALLED |
| | | R109 | NOT INSTALLED | NOT INSTALLED |
| | | R110 | NOT INSTALLED | NOT INSTALLED |
| | | R114 | NOT INSTALLED | NOT INSTALLED |
| | | R120 | NOT INSTALLED | NOT INSTALLED |
| | | R122 | NOT INSTALLED | NOT INSTALLED |
| | | R123 | NOT INSTALLED | NOT INSTALLED |
| SCREW TER. CONNECTORS | | J1 | NOT INSTALLED | NOT INSTALLED |
| | | J2 | 8 POS SCREW TER. | 8 POS SCREW TER. |
| | | J3 | 8 POS SCREW TER. | 8 POS SCREW TER. |
| | | J4 | NOT INSTALLED | NOT INSTALLED |
| | | J5 | 6 POS SCREW TER. | 6 POS SCREW TER. |
| | | J6 | 8 POS SCREW TER. | 8 POS SCREW TER. |
| | | J7 | NOT INSTALLED | NOT INSTALLED |
| | | J8 | 6 POS SCREW TER. | 6 POS SCREW TER. |
| | | J9 | 6 POS SCREW TER. | 6 POS SCREW TER. |
| | | J10 | 8 POS SCREW TER. | 8 POS SCREW TER. |
| | | J11 | NOT INSTALLED | NOT INSTALLED |
| | | J12 | 6 POS SCREW TER. | 6 POS SCREW TER. |
| | | J13 | NOT INSTALLED | NOT INSTALLED |
| | | J14 | 6 POS SCREW TER. | 6 POS SCREW TER. |
| | | J15 | 6 POS SCREW TER. | 6 POS SCREW TER. |
| VOLTAGE REFERENCE | +1.707V | R38 | 14.0K 1% RES | NOT INSTALLED |
| | | C54 | 100 nF CAP | NOT INSTALLED |
| | +2.93V | R36 | 25.5K 1% RES | NOT INSTALLED |
| | | R37 | 10.2K 1% RES | NOT INSTALLED |
| | | C41 | 100 nF CAP | NOT INSTALLED |

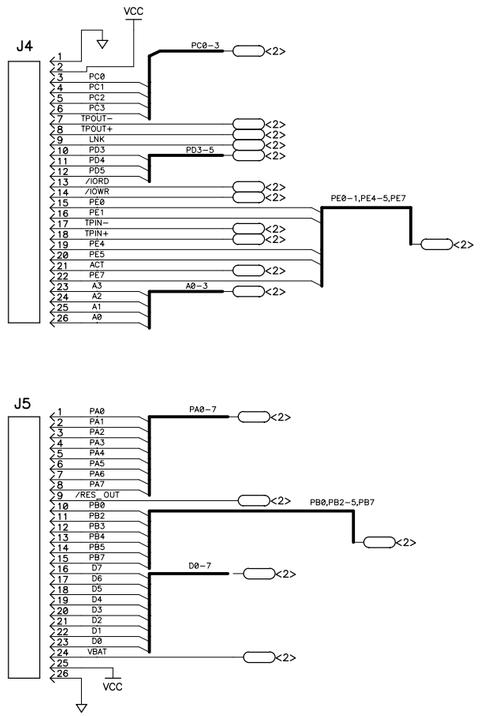
| CIRCUIT | PART | BL2100/2120 | BL2110/2130 |
|---|---|---|---|
| ANALOG POWER (AVCC) | Q3 | 4401 NPN TRAN. | NOT INSTALLED |
| | Q4 | 4401 NPN TRAN. | NOT INSTALLED |
| | CR1 | 27 V ZENER | NOT INSTALLED |
| | R45 | 27K 5% RES | NOT INSTALLED |
| | R48 | 27K 5% RES | NOT INSTALLED |
| | R52 | 27K 5% RES | NOT INSTALLED |
| | C50 | 10 uF CAP | NOT INSTALLED |
| | C65 | 100 nF CAP | NOT INSTALLED |
| | C57 | 10 uF CAP | NOT INSTALLED |
| ANALOG TO DIGITAL CONVERTOR | R2 | 1 MOHM 1% RES | NOT INSTALLED |
| | R3 | 1 MOHM 1% RES | NOT INSTALLED |
| | R4 | 1 MOHM 1% RES | NOT INSTALLED |
| | R5 | 1 MOHM 1% RES | NOT INSTALLED |
| | R6 | 1 MOHM 1% RES | NOT INSTALLED |
| | R8 | 1 MOHM 1% RES | NOT INSTALLED |
| | R9 | 1 MOHM 1% RES | NOT INSTALLED |
| | R10 | 1 MOHM 1% RES | NOT INSTALLED |
| | R11 | 1 MOHM 1% RES | NOT INSTALLED |
| | R12 | 1 MOHM 1% RES | NOT INSTALLED |
| | R13 | 1 MOHM 1% RES | NOT INSTALLED |
| | R16 | 200K 1% RES | NOT INSTALLED |
| | R20 | 200K 1% RES | NOT INSTALLED |
| | R22 | 200K 1% RES | NOT INSTALLED |
| | R23 | 200K 1% RES | NOT INSTALLED |
| | R27 | 200K 1% RES | NOT INSTALLED |
| | R28 | 200K 1% RES | NOT INSTALLED |
| | R41 | 200K 1% RES | NOT INSTALLED |
| | R42 | 200K 1% RES | NOT INSTALLED |
| | R104 | 25K 5% RES | NOT INSTALLED |
| | R112 | 25K 5% RES | NOT INSTALLED |
| | R117 | 25K 5% RES | NOT INSTALLED |
| | C1 | 1000 pF CAP | NOT INSTALLED |
| | C3 | 1000 pF CAP | NOT INSTALLED |
| | C4 | 1000 pF CAP | NOT INSTALLED |
| | C5 | 1000 pF CAP | NOT INSTALLED |
| | C6 | 1000 pF CAP | NOT INSTALLED |
| | C7 | 1000 pF CAP | NOT INSTALLED |
| | C8 | 1000 pF CAP | NOT INSTALLED |
| | C9 | 1000 pF CAP | NOT INSTALLED |
| | C10 | 1000 pF CAP | NOT INSTALLED |
| | C11 | 1000 pF CAP | NOT INSTALLED |
| | C12 | 1000 pF CAP | NOT INSTALLED |
| | C13 | 100 pF CAP | NOT INSTALLED |
| | C14 | 100 pF CAP | NOT INSTALLED |
| | C15 | 100 pF CAP | NOT INSTALLED |
| | C16 | 100 pF CAP | NOT INSTALLED |
| | C18 | 100 pF CAP | NOT INSTALLED |
| | C19 | 100 pF CAP | NOT INSTALLED |
| | C20 | 10 nF CAP | NOT INSTALLED |
| | C23 | 100 pF CAP | NOT INSTALLED |
| | C32 | 100 pF CAP | NOT INSTALLED |
| | C33 | 100 pF CAP | NOT INSTALLED |
| | C46 | 10 nF CAP | NOT INSTALLED |
| | C51 | 100 pF CAP | NOT INSTALLED |
| | C52 | 100 pF CAP | NOT INSTALLED |
| | C53 | 10 nF CAP | NOT INSTALLED |
| | U2 | LMC660 OP AMP | NOT INSTALLED |
| | U5 | TLC2543 A/D | NOT INSTALLED |
| | U7 | LMC660 OP AMP | NOT INSTALLED |

| CIRCUIT | PART | BL2100/2120 | BL2110/2130 |
|---|---|---|---|
| DIGITAL TO ANALOG CONVERTOR | R24 | 255K 1% RES | NOT INSTALLED |
| | R25 | 102K 1% RES | NOT INSTALLED |
| | R26 | 255K 1% RES | NOT INSTALLED |
| | R31 | 102K 1% RES | NOT INSTALLED |
| | R32 | 86.6K 1% RES | NOT INSTALLED |
| | R33 | 86.6K 1% RES | NOT INSTALLED |
| | R34 | 86.6K 1% RES | NOT INSTALLED |
| | R35 | 86.6K 1% RES | NOT INSTALLED |
| | R39 | 255K 1% RES | NOT INSTALLED |
| | R40 | 255K 1% RES | NOT INSTALLED |
| | R46 | 102K 1% RES | NOT INSTALLED |
| | R47 | 102K 1% RES | NOT INSTALLED |
| | C30 | 100 pF CAP | NOT INSTALLED |
| | C31 | 100 pF CAP | NOT INSTALLED |
| | C48 | 100 pF CAP | NOT INSTALLED |
| | C49 | 100 pF CAP | NOT INSTALLED |
| | C50 | 10 nF CAP | NOT INSTALLED |
| | C62 | 10 nF CAP | NOT INSTALLED |
| | C67 | 10 nF CAP | NOT INSTALLED |
| | U6 | LM324 OP AMP | NOT INSTALLED |
| | U9 | TLV5618 D/A | NOT INSTALLED |
| | U10 | TLV5618 D/A | NOT INSTALLED |
| ANALOG CONNECTORS | J2 | 8 POS SCREW TER. | NOT INSTALLED |
| | J3 | 8 POS SCREW TER. | NOT INSTALLED |

## POWER TABLE

| REF DES | DEVICE | DEVICE VOLTAGE INFORMATION | | | | | FILTER CAP |
|---|---|---|---|---|---|---|---|
| | | VCC | AVCC | +V | GND | AGND | |
| U1 | LMC660 | | | 4 | | 11 | C17 |
| U2 | LMC660 | | | 4 | | 11 | C20 |
| U3 | 232A | 16 | | | 15 | | C22 |
| U5 | TLC2543 | | | 20 | | 10 | C46 |
| U6 | LM324 | | 4 | | | 11 | C50 |
| U7 | LMC660 | | | 4 | | 11 | C53 |
| U8 | SP483EN | 8 | | | 5 | | C61 |
| U9 | TLV5618 | | | 8 | | 5 | C62 |
| U10 | TLV5618 | | | 8 | | 5 | C67 |
| U11 | 74VHC245 | 20 | | | 10 | | C77 |
| U12 | 74VHC245 | 20 | | | 10 | | C68 |
| U13 | 74VHC574 | 20 | | | 10 | | C88 |
| U14 | 74VHC245 | 20 | | | 10 | | C89 |
| U15 | 74VHC139 | 16 | | | 8 | | C90 |
| U16 | 74VHC574 | 20 | | | 10 | | C99 |
| U17 | 74VHC574 | 20 | | | 10 | | C100 |
| U18 | 74VHC245 | 20 | | | 10 | | C111 |
| U19 | 74VHC02 | 14 | | | 7 | | C120 |

| SIZE | DWG NO. | |
|---|---|---|
| C | 090–0124 | |
| SCALE NONE | REV LTR A | SHEET 6 OF 6 |




## STUFFING TABLE

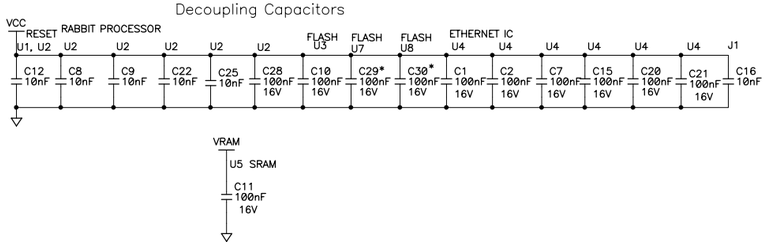
| | CIRCUIT | PART | RCM2200 | RCM2210 | RCM2250 |
|---|---|---|---|---|---|
| POWER TO VRAM SWITCH | WITH BATTERY BACKUP CIRCUITRY | R33 | NOT INSTALLED | NOT INSTALLED | NOT INSTALLED |
| CS CONTROL SWITCH | WITH BATTERY BACKUP CIRCUITRY | R27 | NOT INSTALLED | NOT INSTALLED | NOT INSTALLED |
| SRAM | MAIN | U5 | 128K SRAM | 256K 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 | | | | |

## 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 |

Decoupling Capacitors




NOTES: UNLESS OTHERWISE SPECIFIED;
1. ALL RESISTOR VALUES ARE IN OHMS, 1/16W, 5%
2. ALL CAPACITORS ARE 50VDC OR HIGHER.
3. THE ORIGINATION SOURCE OF A VOLTAGE IS REPRESENTED BY ( VCC ▲ ), AND ALL REFERENCES TO THAT VOLTAGE ARE REPRESENTED BY ( VCC ▼ ).
4. 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.

 SEPTEMBER 11, 2001

| APPEND THE FOLLOWING DOCUMENTS WHEN CHANGING THIS DOCUMENT: | | DRAWING CONTENT: | |
|---|---|---|---|
| | | DRAWN BY: (INITIAL RELEASE) RJH | 3/15/01 |
| | | REVISED BY: RJH | 11SEP01 |
| | | APPROVALS: INITIAL RELEASE | |
| | | PROJECT ENGINEER: RICHARD HESS | 3/22/01 |
| | | ENGINEERING MANAGER: R.MATTHEWS | 3/22/01 |
| | | SIGNATURES | DATE |

TITLE: SCHEMATIC DIAGRAM MICRO ETHERNET RABBITCORES RCM2200 SERIES

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

SIZE: C | DWG NO.: 090-0120

SCALE: NONE | RELEASE DATE | SHEET 1 OF 2

PROCESSOR

SRAM*

FLASH *

FLASH *

FLASH *

U2 RABBIT 2000

3,28,53,78,92=VCC
2,27,39,52,77,89=GND

U5 SRAM 512K X 8

JP7 VRAM
32=VRAM, 16=GND

U3 FLASH* 8=VCC, 24=GND

U8 FLASH* 8=VCC, 24=GND

U7 FLASH* 8=VCC, 24=GND

JP4 FLASH SELECT*
1-2 128K/256K FLASH
2-3 512K FLASH

JP6 FLASH SELECT*
1-2 128K/256K FLASH
2-3 512K FLASH

JP1 FLASH SELECT*
1-2 128K/256K FLASH
2-3 512K FLASH

POWER TO RABBIT RTC AND 32KHZ OSC.

TO RABBIT

TO OSCILLATOR

32KHZ OSC.

POWER TO VRAM

TO SRAM

CS Control Switch

RESET GENERATOR

ETC811L

RESET=LOW

J1 RABBIT PROGRAMMING PORT

RXA
GND
CLKA
VCC
/RESET
TXA
N.C.
STATUS
SMODE0
SMODE1

BATTERY NOT INSTALLED

U4 RTL8019AS

6,17,47,57,70,89=VCC; 14,28,44,52,83,86=GND

J2*

XMIT

RCV

LF1S022

RJ-45 FRONT VIEW

RJ-45 CONNECTOR

TRANSMIT−
TRANSMIT+
RECEIVE−
RECEIVE+

10BASE−T ETHERNET

11 SEP 2001

COPYRIGHT 2001, Z−WORLD, INC.

| SIZE | DWG NO. | |
|---|---|---|
| C | 090−0120 | |
| SCALE NONE | REV LTR C | SHEET 2 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 | | | | |

## 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,C10,C9,22,23 |
| U3 | FLASH | | 24 | 8 | | | C16 |
| U5 | SRAM 128K X 8 | | 24 | | 8 | | C11 |

NOTES: UNLESS OTHERWISE SPECIFIED;
1. ALL RESISTOR VALUES ARE IN OHMS, 1/16W, 5%
2. ALL CAPACITORS ARE 50VDC OR HIGHER.
3. THE ORIGINATION SOURCE OF A VOLTAGE IS REPRESENTED BY ( $\overset{VCC}{\uparrow}$ ), AND ALL REFERENCES TO THAT VOLTAGE ARE REPRESENTED BY ( $\overset{VCC}{\perp}$ ).
4. R27, R33, J2, J3, BT1, DS1, & R29 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.
6. JP1 AND JP2 ARE JUMPERED POSITION 1 TO POSITION 2 BY FACTORY DEFAULT.

 OCT 08, 2001

| APPEND THE FOLLOWING DOCUMENTS WHEN CHANGING THIS DOCUMENT: | |
|---|---|
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |

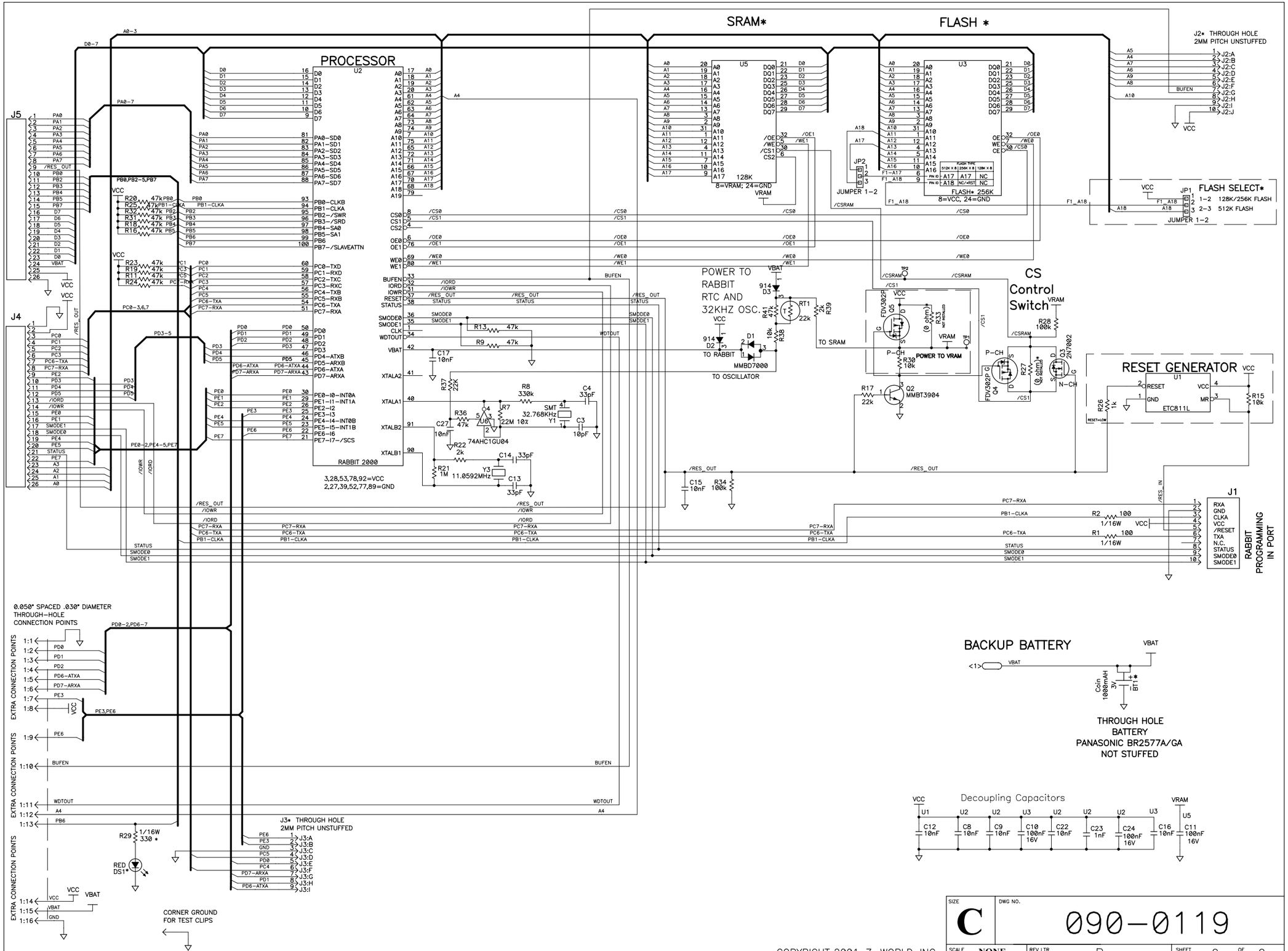
| DRAWING CONTENT: | |
|---|---|
| DRAWN BY: (INITIAL RELEASE) RJH | 3/22/01 |
| REVISED BY: RJH | 08OCT01 |
| APPROVALS: INITIAL RELEASE | |
| PROJECT ENGINEER: | |
| ENGINEERING MANAGER: | |
| SIGNATURES | DATE |

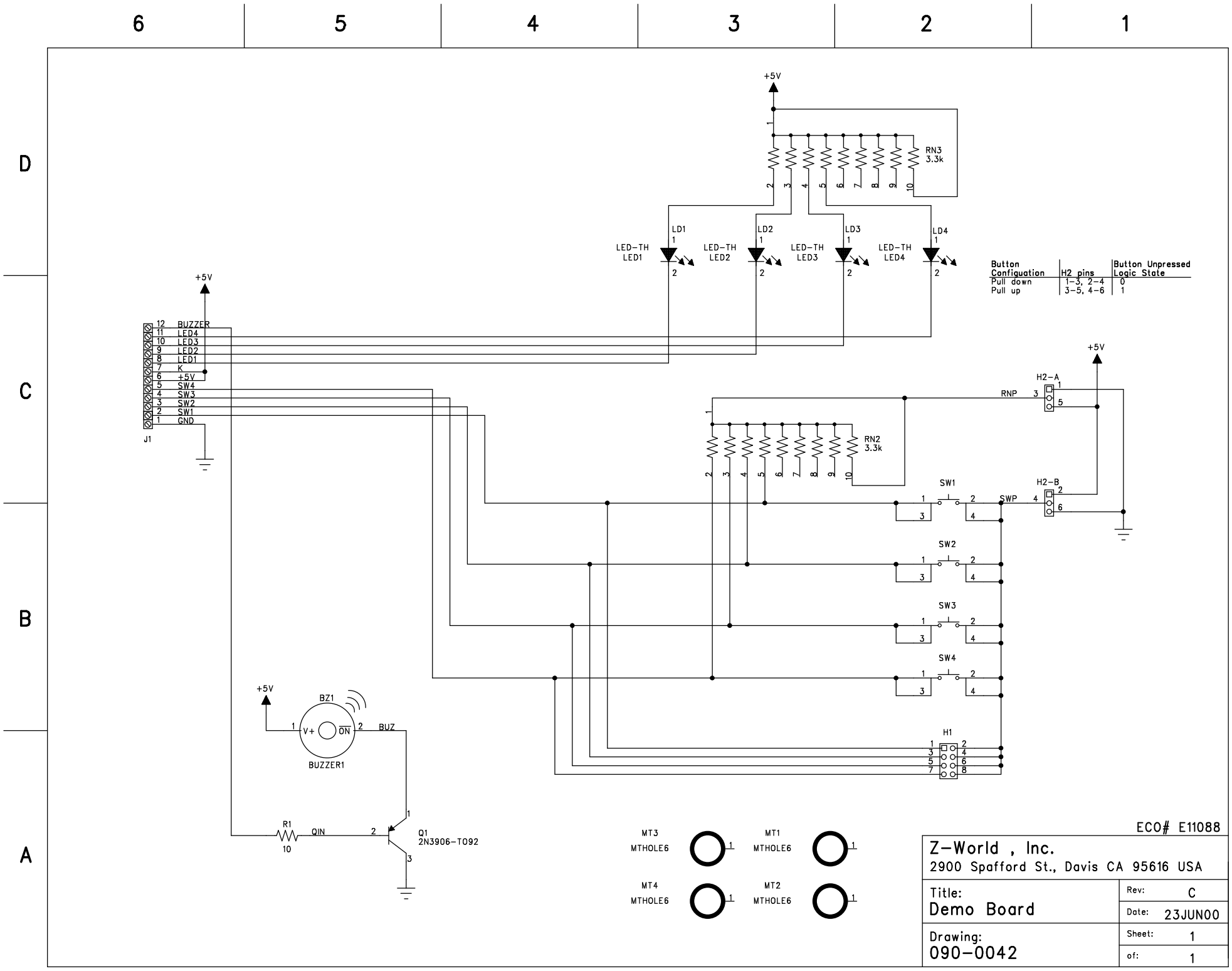
TITLE: SCHEMATIC DIAGRAM RCM2300

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

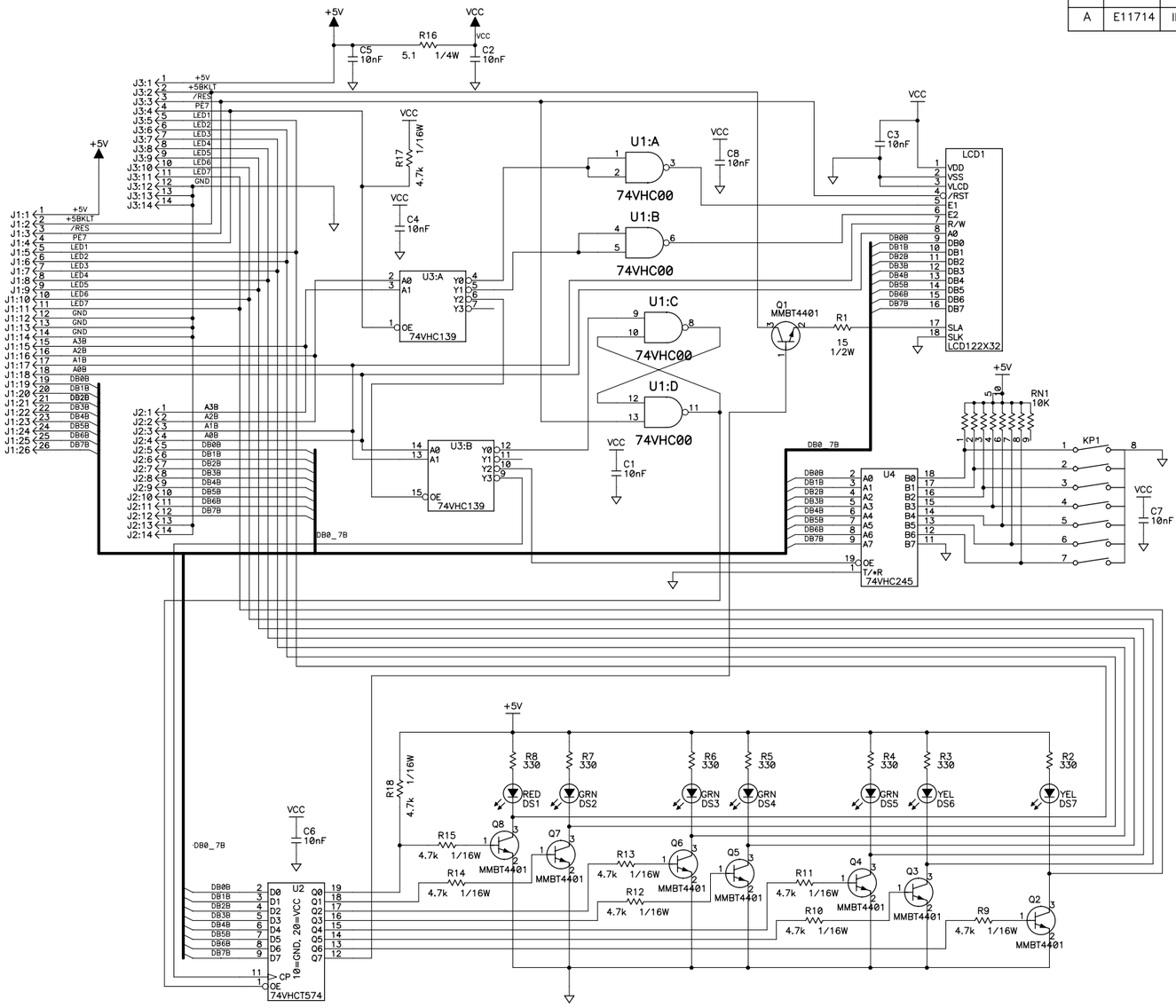
SIZE: C | DWG NO. 090–0119

SCALE: NONE | RELEASE DATE | SHEET 1 OF 2


PROCESSOR
U2
RABBIT 2000
SRAM*
FLASH *
U5
128K
8=VRAM; 24=GND
VRAM
U3
FLASH* 256K
8=VCC, 24=GND
JUMPER 1-2
J2* THROUGH HOLE
2MM PITCH UNSTUFFED
FLASH SELECT*
1-2 128K/256K FLASH
2-3 512K FLASH
JUMPER 1-2
J5
J4
POWER TO
RABBIT
RTC AND
32KHZ OSC.
TO RABBIT
TO OSCILLATOR
MMBD7000
TO SRAM
CS
Control
Switch
POWER TO VRAM
MMBT3904
RESET GENERATOR
ETC811L
3,28,53,78,92=VCC
2,27,39,52,77,89=GND
74AHC1GU04
11.0592MHz
32.768KHz
J1
RXA
GND
CLKA
VCC
/RESET
TXA
N.C.
STATUS
SMODE0
SMODE1
RABBIT
PROGRAMMING
IN PORT
0.050" SPACED .030" DIAMETER
THROUGH-HOLE
CONNECTION POINTS
EXTRA CONNECTION POINTS
J3* THROUGH HOLE
2MM PITCH UNSTUFFED
CORNER GROUND
FOR TEST CLIPS
BACKUP BATTERY
THROUGH HOLE
BATTERY
PANASONIC BR2577A/GA
NOT STUFFED
Decoupling Capacitors
COPYRIGHT 2001, Z-WORLD, INC.
090-0119
SIZE
C
DWG NO.
SCALE
NONE
REV LTR
B
SHEET
2
OF
2


6
5
4
3
2
1
D
C
B
A
+5V
RN3
3.3k
LD1
LD2
LD3
LD4
LED-TH
LED1
LED-TH
LED2
LED-TH
LED3
LED-TH
LED4
Button Configuation
H2 pins
Button Unpressed Logic State
Pull down
1-3, 2-4
0
Pull up
3-5, 4-6
1
BUZZER
LED4
LED3
LED2
LED1
K
+5V
SW4
SW3
SW2
SW1
GND
J1
RN2
3.3k
RNP
H2-A
H2-B
SWP
SW1
SW2
SW3
SW4
H1
BZ1
V+
ON
BUZ
BUZZER1
R1
10
QIN
Q1
2N3906-TO92
MT3
MTHOLE6
MT1
MTHOLE6
MT4
MTHOLE6
MT2
MTHOLE6
ECO# E11088
Z-World , Inc.
2900 Spafford St., Davis CA 95616 USA
Title:
Demo Board
Rev: C
Date: 23JUN00
Drawing:
090-0042
Sheet: 1
of: 1

| REVISION HISTORY | | | REVISION APPROVAL | | | |
|---|---|---|---|---|---|---|
| REV | ECO | DESCRIPTION OF CHANGE | PROJECT ENGINEER | APPROVAL DATE | DOCUMENT CONTROL | APPROVAL DATE |
| A | E11714 | INITIAL RELEASE | | | | |




## POWER TABLE

| REF DES | DEVICE | DEVICE VOLTAGE INFORMATION | | | DEVICE: FILTER CAP |
|---|---|---|---|---|---|
| | | GND | VCC | NO CONNECTS | REF DES(s) |
| U1 | 74VHC00 | 7 | 14 | | C8 |
| U2 | 74VHCT574 | 1, 8 | 16 | 12 | C6 |
| U3 | 74VHC139 | 8 | 16 | 10 | C4 |
| U4 | 74VHC245 | 1, 8, 11 | 16 | | C7 |

NOTES: UNLESS OTHERWISE SPECIFIED;

1. ALL RESISTOR VALUES ARE IN OHMS, 1/16W, 5%
2. ALL CAPACITORS ARE 50VDC OR HIGHER.
3. THE ORIGINATION SOURCE OF A VOLTAGE IS REPRESENTED BY ( VCC ), AND ALL REFERENCES TO THAT VOLTAGE ARE REPRESENTED BY ( VCC ).
4. ▭ OUTLINED CIRCUIT MAY NOT BE STUFFED DEPENDING ON MODEL, SEE STUFFING CHART FOR CLARIFICATION.
5. COMPONENT VALUES SHOWN WITH AN ASTERISK (*) FOLLOWING THE VALUE, MAY HAVE DIFFERENT VALUES, OR MAY NOT BE STUFFED DEPENDING ON MODEL. SEE STUFFING CHART FOR CLARIFICATION.

| APPEND THE FOLLOWING DOCUMENTS WHEN CHANGING THIS DOCUMENT: | |
|---|---|
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |

| DRAWING CONTENT: | |
|---|---|
| DRAWN BY: (INITIAL RELEASE) J. FENG | 23JUL01 |
| REVISED BY: | |
| **APPROVALS: INITIAL RELEASE** | |
| PROJECT ENGINEER: J. FENG | 10OCT01 |
| ENGINEERING MANAGER: | |
| **SIGNATURES** | **DATE** |

TITLE

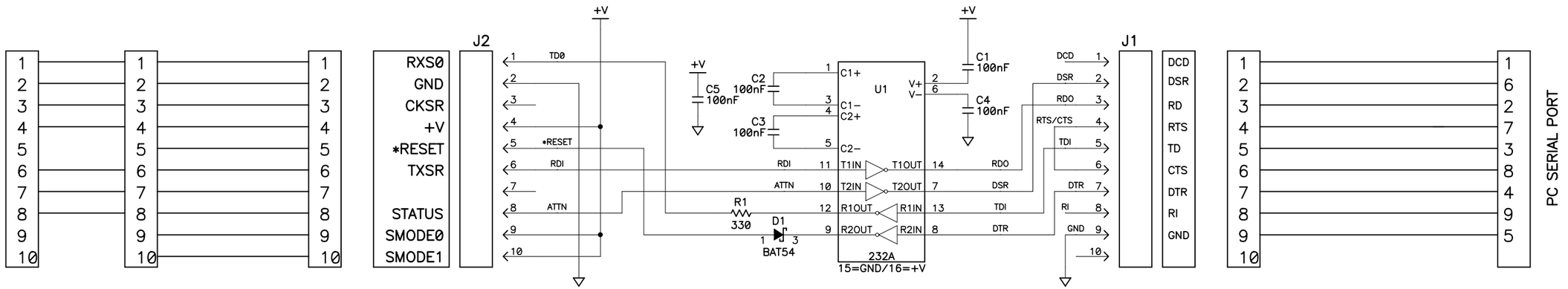
# SCHEMATIC DIAGRAM BL2100 SERIES / DISPLAY BOARD


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


SIZE **C** | DWG NO. 090-0125

SCALE **NONE** | RELEASE 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 | | | | |




Prog. / Diagnostic Cable

Serial Interface Board

Cable 10–pin to D9 female

NOTES: UNLESS OTHERWISE SPECIFIED;
1. ALL RESISTOR VALUES ARE IN OHMS, 1/10W, 5%
2. ALL CAPACITORS ARE 50VDC OR HIGHER.
3. THE ORIGINATION SOURCE OF A VOLTAGE IS REPRESENTED BY ( $\overset{VCC}{\blacktriangle}$ ), AND ALL REFERENCES TO THAT VOLTAGE ARE REPRESENTED BY ( $\overset{VCC}{\top}$ ).

| APPEND THE FOLLOWING DOCUMENTS WHEN CHANGING THIS DOCUMENT: | | DRAWING CONTENT: | | TITLE | |
|---|---|---|---|---|---|
| | | DRAWN BY: (INITIAL RELEASE) E. PEAK | 14MAY01 | SCHEMATIC DIAGRAM RABBIT PROG. CABLE | 2900 SPAFFORD ST. DAVIS, CA 95616 530 - 757 - 4616 |
| | | REVISED BY: K.SCHALLER | 10/04/01 | | |
| | | APPROVALS: INITIAL RELEASE | | | |
| | | PROJECT ENGINEER: | | SIZE B | DWG NO. 090–0128 |
| | | ENGINEERING MANAGER: | | | |
| | | SIGNATURES | DATE | SCALE NONE | RELEASE DATE / SHEET 1 OF 1 |