PRODUCT MANUAL

Jackrabbit (BL1800)

C-Programmable Single-Board Computer

User's Manual

019–0067 • 090515–J

Jackrabbit (BL1800) User's Manual

Part Number 019-0067 • 090515–J • Printed in U.S.A.

Trademarks

Rabbit and Dynamic C are registered trademarks of Digi International Inc.

Rabbit 2000 is a trademark of Digi International Inc.

The latest revision of this manual is available on the Rabbit Web site, www.rabbit.com, for free, unregistered download.

Digi International Inc.

www.rabbit.com

TABLE OF CONTENTS

1. INTRODUCTION

The Jackrabbit is a high-performance, C-programmable single-board computer with a compact form factor. A Rabbit® 2000 microprocessor operating at 29.5 MHz provides fast data processing.

1.1 Features

- 29.5 MHz clock
- 24 CMOS-compatible I/O
- 3 analog channels: 1 A/D input, 2 PWM D/A outputs
- 4 high-power outputs (factory-configured as 3 sinking and 1 sourcing)
- 4 serial ports (2 RS-232 or 1 RS-232 with RTS/CTS, 1 RS-485, and 1 CMOS-compatible)
- 6 timers (five 8-bit timers and one 10-bit timer)
- 128K SRAM, 256K flash EPROM
- Real-time clock
- Watchdog supervisor
- Voltage regulator
- Backup battery

Three Jackrabbit models are available. Their standard features are summarized in Table 1.

Table 1. Jackrabbit Features

Model	Features
BL1800	Full-featured controller with switching voltage regulator.
BL1810	BL1800 with 14.74 MHz clock, 128K flash EPROM, linear voltage regulator, sinking outputs sink up to 200 mA, sourcing output sources up to 100 mA, RS-232 serial ports rated for 1 kV ESD
BL1820	BL1810 with 3 additional digital I/O, no RS-485, no backup battery.

Throughout this manual, the term Jackrabbit refers to all three Jackrabbit models in Table 1; individual models are referred to specifically according to the model number in Table 1.

Appendix A provides detailed specifications.

Visit the 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.

1.2 Development and Evaluation Tools

A complete Development Kit, including a Prototyping Board and Dynamic C development software, is available for the Jackrabbit. The Development Kit puts together the essentials you need to design an embedded microprocessor-based system rapidly and efficiently.

1.3 How to Use This Manual

This user's manual is intended to give users detailed information on the Jackrabbit. It does not contain detailed information on the Dynamic C development environment or the Rabbit 2000® microprocessor. Most users will want more detailed information on some or all of these topics in order to put the Jackrabbit to effective use.

1.3.1 Additional Product Information

In addition to the product-specific information contained in the ***Jackrabbit (BL1800) User's Manual*** (this manual), several higher level reference manuals are provided in HTML and PDF form on the accompanying CD-ROM. Advanced users will find these references valuable in developing systems based on the Jackrabbit:

- ***Dynamic C User's Manual***
- ***Dynamic C Function Reference Manual***
- ***Rabbit 2000 Microprocessor User's Manual***

1.3.2 Online Documentation

The online documentation is installed along with Dynamic C, and an icon for the documentation menu is placed on the workstation's desktop. Double-click this icon to reach the menu. If the icon is missing, use your browser to find and load **default.htm** in the **docs** folder, found in the Dynamic C installation folder.

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

1.4 CE Compliance

Equipment is generally divided into two classes.

CLASS A	CLASS B
Digital equipment meant for light industrial use	Digital equipment meant for home use
Less restrictive emissions requirement: less than 40 dB µV/m at 10 m (40 dB relative to 1 µV/m) or 300 µV/m	More restrictive emissions requirement: 30 dB µV/m at 10 m or 100 µV/m

These limits apply over the range of 30–230 MHz. The limits are 7 dB higher for frequencies above 230 MHz. Although the test range goes to 1 GHz, the emissions from Rabbit-based systems at frequencies above 300 MHz are generally well below background noise levels.

The Jackrabbit single-board computer has been tested and was found to be in conformity with the following applicable immunity and emission standards. The BL1810 and BL1820 single-board models are also CE qualified as they are sub-versions of the Jackrabbit. Boards that are CE-compliant have the CE mark.

NOTE: Earlier versions of the Jackrabbit sold before 2002 that do not have the CE mark are ***not*** CE-complaint.

Immunity

The Jackrabbit series of single-board computers meets the following EN55024/1998 immunity standards.

- EN61000-4-3 (Radiated Immunity)
- EN61000-4-4 (EFT)
- EN61000-4-6 (Conducted Immunity)

Additional shielding or filtering may be required for a heavy industrial environment.

Emissions

The Jackrabbit series of single-board computers meets the following emission standards with the Rabbit 2000 spectrum spreader turned on and set to the normal mode. The spectrum spreader is only available with Rev. C or higher of the Rabbit 2000 microprocessor. This microprocessor is used in all Jackrabbit series boards that carry the CE mark.

- EN55022:1998 Class B
- FCC Part 15 Class B

In order for the Jackrabbit boards to meet these EN55022:1998 Class B standards, you must add ferrite absorbers to the serial I/O cables used for RS-232 and RS-485 serial communication. Depending on your application, you may need to add ferrite absorbers to the

digital I/O cables. Your results may vary, depending on your application, so additional shielding or filtering may be needed to maintain the Class B emission qualification.

> **NOTE:** If no ferrite absorbers are fitted, the Jackrabbit boards will still meet EN55022:1998 Class A requirements as long as the spectrum spreader is turned on.

The spectrum spreader is on by default for Jackrabbit models BL1810 and BL1820. The spectrum spreader is off by default for the Jackrabbit model BL1800, and must be turned on with at least one wait state in order for the BL1800 model to be CE-compliant. Section 3.8.3 provides further information about the spectrum spreader and its use, and includes information on how to add a wait state.

1.4.1 Design Guidelines

Note the following requirements for incorporating the Jackrabbit series of single-board computers into your application to comply with CE requirements.

General

- The power supply provided with the Development Kit is for development purposes only. It is the customer's responsibility to provide a CE-compliant power supply for the end-product application.
- When connecting the Jackrabbit single-board computer to outdoor cables, the customer is responsible for providing CE-approved surge/lightning protection.
- Rabbit recommends placing digital I/O or analog cables that are 3 m or longer in a metal conduit to assist in maintaining CE compliance and to conform to good cable design practices. Rabbit also recommends using properly shielded I/O cables in noisy electromagnetic environments.
- When installing or servicing the Jackrabbit, it is the responsibility of the end-user to use proper ESD precautions to prevent ESD damage to the Jackrabbit.

Safety

- For personal safety, all inputs and outputs to and from the Jackrabbit series of single-board computers must not be connected to voltages exceeding SELV levels (42.4 V AC peak, or 60 V DC). Damage to the Rabbit 2000 microprocessor may result if voltages outside the design range of 0 V to 5.5 V DC are applied directly to any of its digital inputs.
- The lithium backup battery circuit on the Jackrabbit single-board computer has been designed to protect the battery from hazardous conditions such as reverse charging and excessive current flows. Do not disable the safety features of the design.

1.4.2 Interfacing the Jackrabbit to Other Devices

Since the Jackrabbit series of single-board computers is designed to be connected to other devices, good EMC practices should be followed to ensure compliance. CE compliance is ultimately the responsibility of the integrator. Additional information, tips, and technical assistance are available from your authorized Rabbit distributor, and are also available on our Web site at www.rabbit.com.

2. GETTING STARTED

This chapter describes the Jackrabbit board in more detail, and explains how to set up and use the accompanying Prototyping Board.

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

2.1 Development Kit Contents

The Jackrabbit Development Kit contains the following items:

- BL1810 single-board computer.
- Prototyping Board.
- Universal AC adapter, 12 V DC, 1 A (includes Canada/Japan/U.S., Australia/N.Z., U.K., and European style plugs). If you are using another power supply, it must provide 7.5–25 V DC at 5 W.

  NOTE: The linear voltage regulator becomes rather hot for voltages above 15 V.
- 10-pin header to DB9 programming cable with integrated level-matching circuitry.
- ***Dynamic C*** CD-ROM, with complete product documentation on disk.
- ***Getting Started*** instructions.
- A bag of accessory parts for use on the Prototyping Board.
- Screwdriver.
- ***Rabbit 2000 Processor Easy Reference*** poster.
- Registration card.

2.2 Development Hardware Connections

There are three steps to connecting the Prototyping Board for use with Dynamic C and the sample programs:

1. Attach the Jackrabbit to the Prototyping Board.
2. Connect the programming cable between the Jackrabbit and the workstation PC.
3. Connect the power supply to the Jackrabbit.

2.2.1 Attach Jackrabbit to Prototyping Board

To attach the Jackrabbit board to the Prototyping Board, turn the Jackrabbit board over so that the battery is facing up. Plug the pins from headers J4 and J5 on the bottom side of the Jackrabbit board into the header sockets at J2 and J6 on the Prototyping Board as indicated in Figure 1.

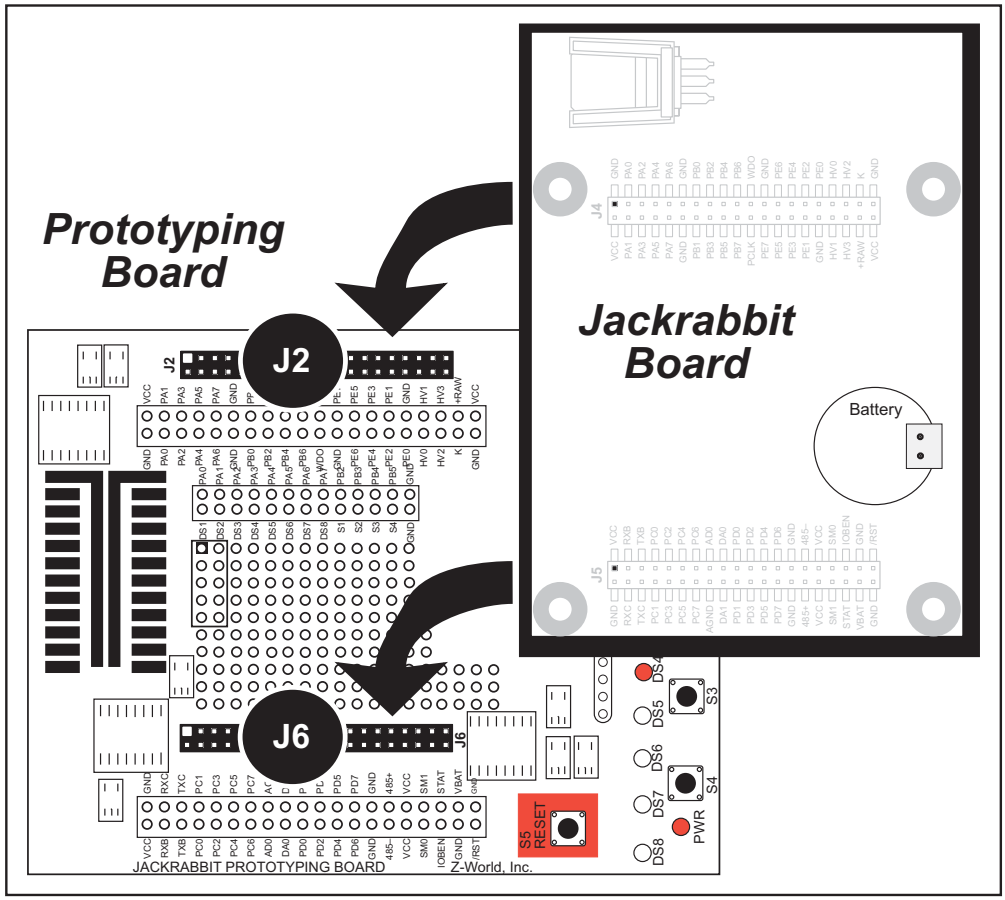


Figure 1. Attach Jackrabbit Board to Prototyping Board

NOTE: It is important that you line up the pins on headers J4 and J5 of the Jackrabbit board exactly with the corresponding pins of header sockets J2 and J6 on the Prototyping Board. The header pins may become bent or damaged if the pin alignment is offset, and the Jackrabbit might not work. Permanent electrical damage to the may also result if a misaligned Jackrabbit is powered up.

Press the Jackrabbit's pins firmly into the Prototyping Board headers.

2.2.2 Connect Programming Cable

The programming cable connects the Jackrabbit to the PC running Dynamic C to download programs and to monitor the Jackrabbit during debugging.

Connect the 10-pin connector of the programming cable labeled **PROG** to header J3 on the Jackrabbit board as shown in Figure 2. Be sure to orient the marked (usually red) edge of the cable towards pin 1 of the connector. (Do not use the **DIAG** connector, which is used for a normal serial connection.)

NOTE: Use only the programming cable that has a red shrink wrap around the RS-232 level converter (Part No. 20-101-0513), which is supplied with the Development Kit. Other Rabbit programming cables are not voltage-compatible or their connector sizes may be different.

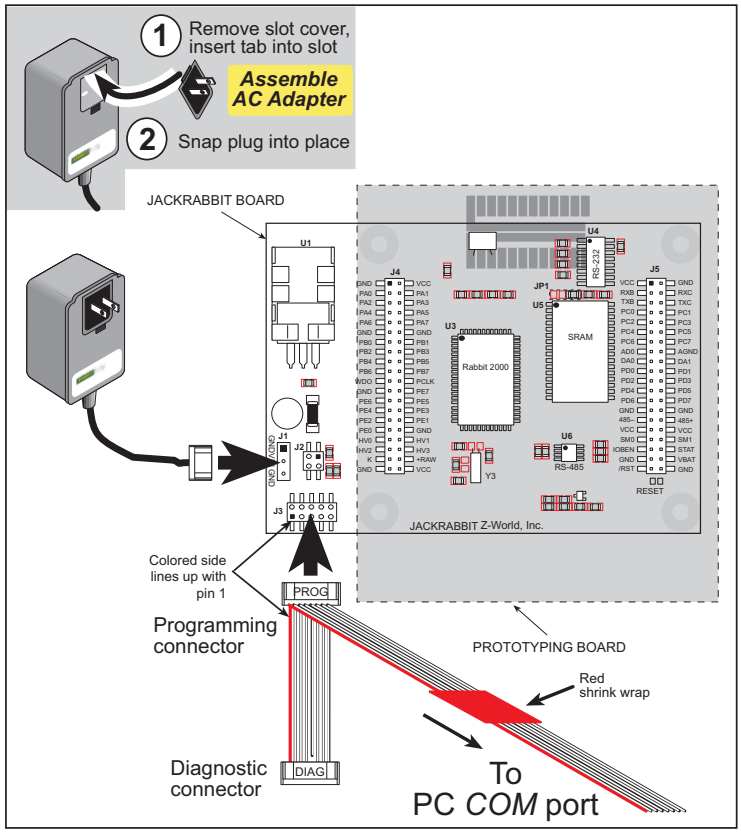


Figure 2. Power and Programming Cable Connections to Jackrabbit Board

NOTE: Never disconnect the programming cable by pulling on the ribbon cable. Carefully pull on the connector to remove it from the header.

Connect the other end of the programming cable to a COM port on your PC.

NOTE: Some PCs now come equipped only with a USB port. It may be possible to use an RS-232/USB converter (Part No. 20-151-0178) with the programming cable supplied with the Jackrabbit Development Kit. Note that not all RS-232/USB converters work with Dynamic C.

2.2.3 Connect Power

When all other connections have been made, you can connect power to the Jackrabbit.

First, prepare the AC adapter for the country where it will be used by selecting the plug. The Jackrabbit Development Kit presently includes Canada/Japan/U.S., Australia/N.Z., U.K., and European style plugs. Snap in the top of the plug assembly into the slot at the top of the AC adapter as shown in Figure 2, then press down on the spring-loaded clip below the plug assembly to allow the plug assembly to click into place.

Hook up the connector from the AC adapter to header J1 on the Jackrabbit board as shown in Figure 2. The orientation of this connector is not important since the VIN (positive) voltage is the middle pin, and GND is available on both ends of the three-pin header J1.

Plug in the AC adapter. The Jackrabbit board and the Prototyping Board are ready to be used.

> **NOTE:** A **RESET** button is provided on the Prototyping Board (see Figure 1) to allow hardware reset without disconnecting power.

To power down the Jackrabbit, unplug the power connector from J1. You should disconnect power before making any circuit adjustments in the prototyping area, changing any connections to the board, or removing the Jackrabbit from the Prototyping Board.

2.3 Installing Dynamic C

If you have not yet installed Dynamic C, do so now by inserting the Dynamic C CD from the Jackrabbit Development Kit 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 **setup.exe** file on your CD drive. Click **OK** to begin the installation once you have selected the **setup.exe** file.

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

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

The ***Dynamic C 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.

Once your installation is complete, you will have up to three icons on your PC desktop. One icon is for Dynamic C, one opens the documentation menu, and the third is for the Rabbit Field Utility, a tool used to download precompiled software to a target system.

If you have purchased any of the optional Dynamic C modules, install them after installing Dynamic C. The modules may be installed in any order. You must install the modules in the same directory where Dynamic C was installed.

2.4 Run a Sample Program

If you already have Dynamic C installed, you are now ready to test your programming connections by running a sample program. Start Dynamic C by double-clicking on the Dynamic C icon on your desktop or in your **Start** menu. Dynamic C uses the serial port specified during installation.

If you are using a USB port to connect your computer to the BL1810, click on the **Communications** tab and verify that **Use USB to Serial Converter** is selected to support the USB programming cable. Click **OK**.You may have to determine which COM port was assigned to the RS-232/USB converter. Open **Control Panel > System > Hardware > Device Manager > Ports** and identify which COM port is used for the USB connection. In Dynamic C, select **Options > Project Options**, then select this COM port on the **Communications** tab, then click **OK**. You may type the COM port number followed by **Enter** on your computer keyboard if the COM port number is outside the range on the dropdown menu.

Find the file **PONG.C**, which is in the Dynamic C **SAMPLES** folder. To run the program, open it with the **File** menu, compile it using the **Compile** menu, and then run it by selecting **Run** in the **Run** menu. The **STDIO** window will open on your PC and will display a small square bouncing around in a box. This program shows that the CPU is working.

2.4.1 Troubleshooting

If Dynamic C cannot find the target system (error message **"No Rabbit Processor Detected."**):

- Check that the BL1810 is powered correctly — the AC adapter should be connected to header J1 on the Jackrabbit board and should be plugged in to a wall outlet.
- Check both ends of the programming cable to ensure that they are firmly plugged into the PC and that the **PROG** connector, not the **DIAG** connector, is plugged in to the programming port on the BL1810 with the colored side lined up with pin 1.
- Ensure that the BL1810 is firmly and correctly installed in its sockets on the Prototyping Board.
- Select a different COM port within Dynamic C. From the **Options** menu, select **Project Options**, then select another COM port from the list on the **Communications** tab, then click **OK**. Press **<Ctrl-Y>** to force Dynamic C to recompile the BIOS.

If a program compiles and loads, but then loses target communication before you can begin debugging, it is possible that your PC cannot handle the default debugging baud rate. Try lowering the debugging baud rate as follows.

- Locate the **Serial Options** dialog in the Dynamic C **Options > Project Options > Communications** menu. Choose a lower debug baud rate, then click **OK**.

2.5 Where Do I Go From Here?

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

1. Run all of the sample programs described in Section 4.2 to get a basic familiarity with Dynamic C and the Jackrabbit's capabilities.
2. For further development, refer to this ***Jackrabbit (BL1800) User's Manual*** for details of the board's hardware components.

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

2.5.1 Real-Time Clock

If you plan to use the real-time clock functionality in your application, you will need to set the real-time clock. You may set the real-time clock using the **SETRTCKB.C** sample program from the Dynamic C **SAMPLES\RTCLOCK** folder. The **RTC_TEST.C** sample program in the Dynamic C **SAMPLES\RTCLOCK** folder provides additional examples of how to read and set the real-time clock

2.5.2 Technical Support

NOTE: If you purchased your Jackrabbit through a distributor or through a Rabbit partner, contact the distributor or partner first for technical support.

If there are any problems at this point:

- Use the Dynamic C **Help** menu to get further assistance with Dynamic C.
- Check the Rabbit Technical Bulletin Board and forums at www.rabbit.com/support/bb/ and at www.rabbit.com/forums/.
- Use the Technical Support e-mail form at www.rabbit.com/support/.

3. SUBSYSTEMS

Chapter 3 describes the principal subsystems and their use for the Jackrabbit.

- Digital Inputs/Outputs
- A/D Converter
- D/A Converters
- Serial Communication
- Memory

Figure 3 shows these Rabbit-based subsystems designed into the Jackrabbit.

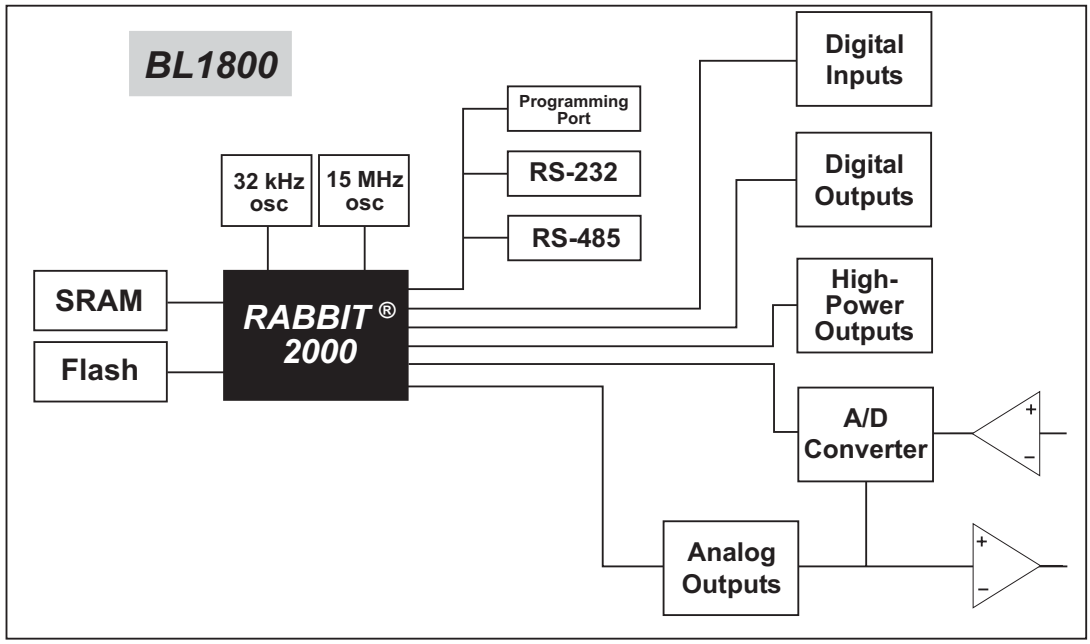


Figure 3. Jackrabbit Subsystems

3.1 Jackrabbit Pinouts

Figure 4 shows the pinout for headers J4 and J5, which carry the signals associated with the Jackrabbit subsystems.

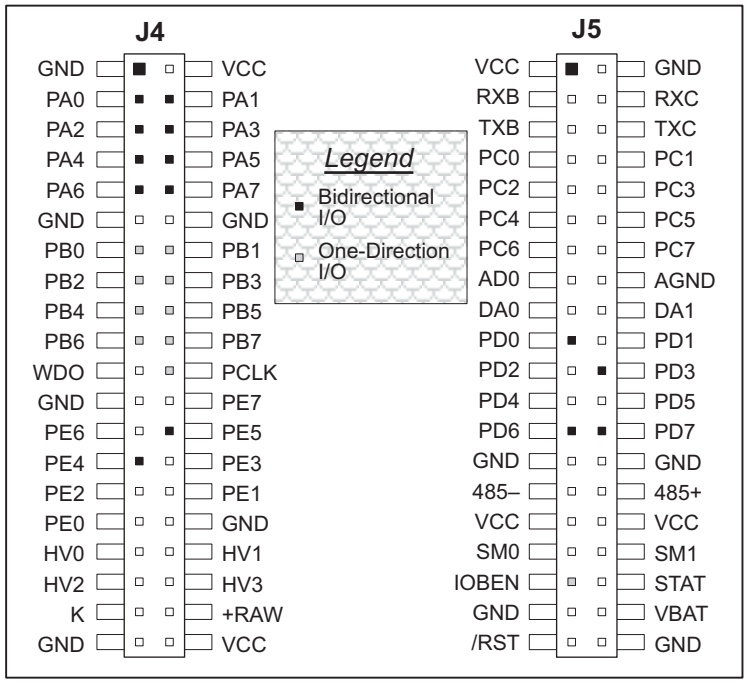


Figure 4. Pinout for Jackrabbit Headers J4 and J5

3.1.1 Headers

Standard Jackrabbit models are equipped with two 2 × 20 IDC headers (J4 and J5) with a 2 mm pitch.

3.2 Digital Inputs/Outputs

3.2.1 Digital Inputs

The Jackrabbit has six CMOS-level digital inputs, PB0–PB5, each of which is pulled up to +5 V as shown in Figure 5. The BL1820, which does not have RS-485, has one additional CMOS-level digital input, PC1.

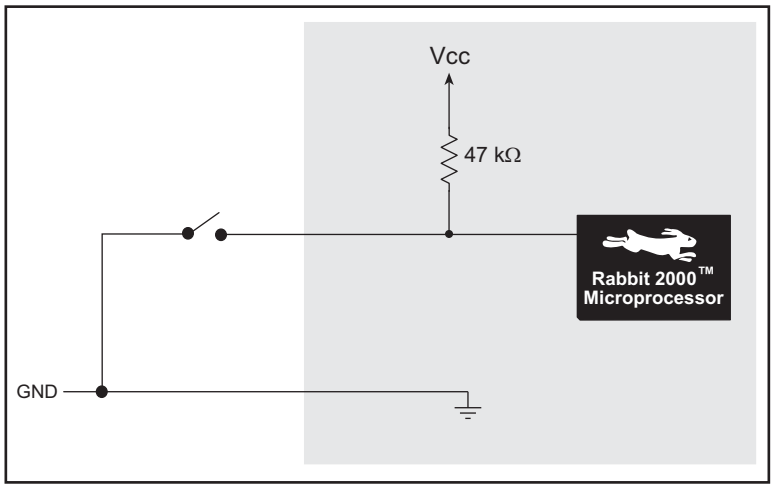


Figure 5. Digital Inputs

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

NOTE: Since the voltage limits on the inputs to the Rabbit 2000 microprocessor are 0 to 5.5 V DC, the end user must ensure that the voltage applied to any I/O pin is within these limits.

3.2.2 Digital Outputs

The Jackrabbit has four CMOS-level digital outputs, PB6–PB7, PCLK, and IOBEN. Four high-power outputs, HV0–HV3, are also available—HV0–HV2 can each sink up to 1 A (200 mA for the BL1810 and BL1820) at 30 V, and HV3 can source up to 500 mA (100 mA for the BL1810 and BL1820) at 30 V. The BL1820, which does not have RS-485, has one additional CMOS-level digital output, PC0.

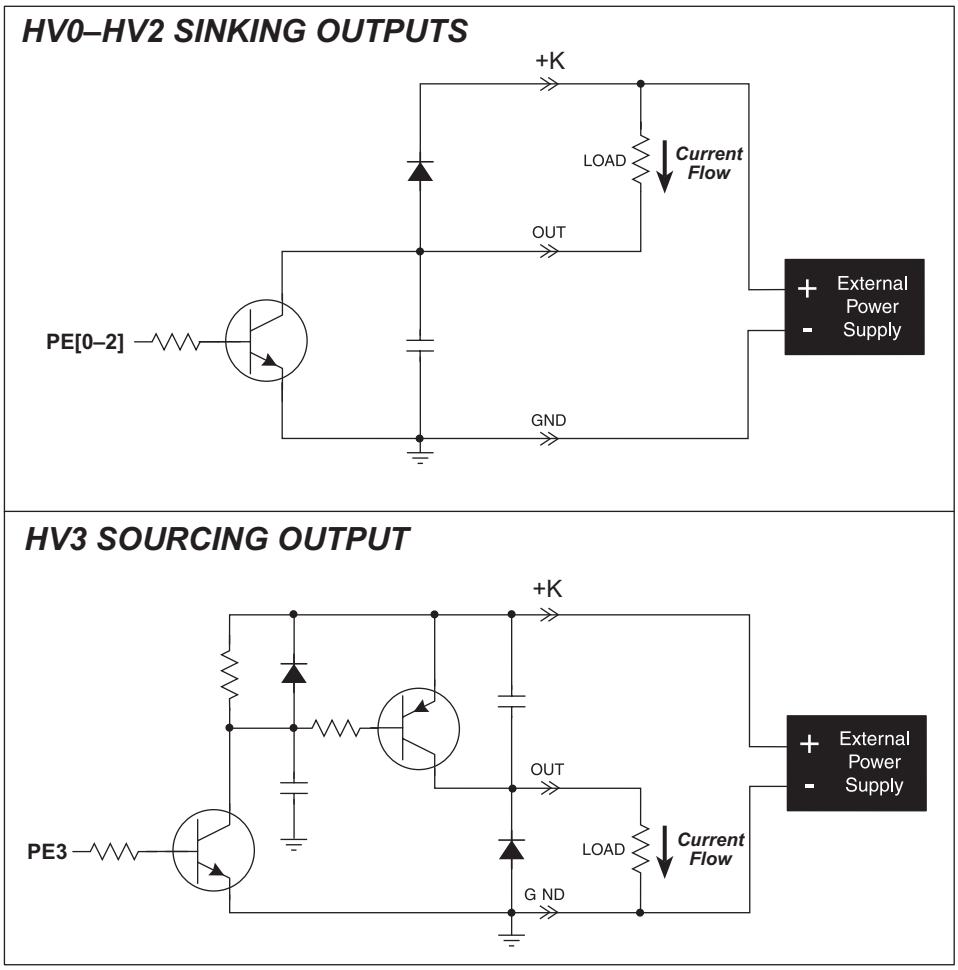


Figure 6. Jackrabbit High-Power Digital Outputs

The common power supply for the four high-power outputs is called K, and is available on header J4. Connect K to the power supply that powers the load, which is usually a separate power supply to that used for the Jackrabbit, and must be no more than 30 V because of the power limitations of the resistors used in the sourcing output circuit.

The K connection performs two functions.

1. K supplies power to the sinking/sourcing transistors used in the high-power circuits.
2. A diode-capacitor combination in the circuit "snubs" voltage transients when inductive loads such as relays and solenoids are driven.

3.2.2.1 Configurable High-Power Output (HV3)

HV3, shown schematically in Figure 7, is factory-configured to be a sourcing output.

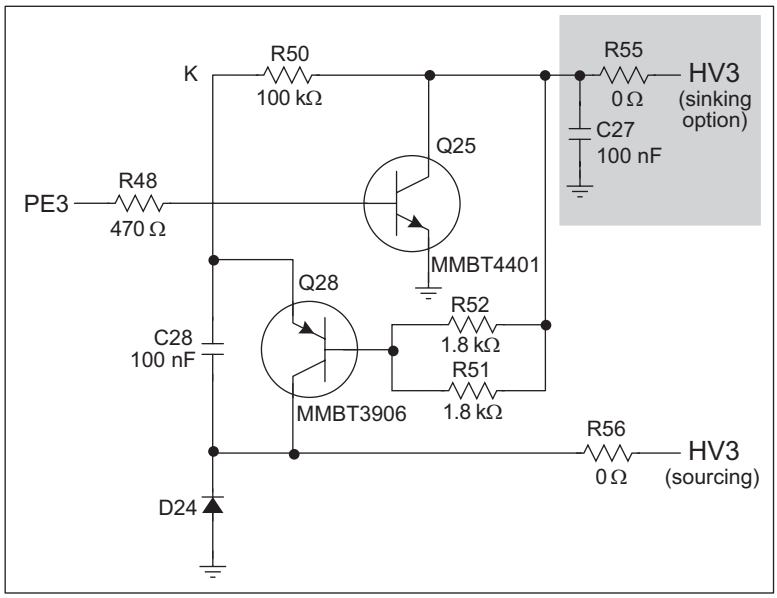


Figure 7. Configurable High-Current Output

When used as a sourcing output, HV3 is switched to K when PE3 on the Rabbit 2000 goes high, and the two transistors shown in Figure 7 are turned on. The maximum sourcing current is 100 mA (BL1810 and BL1820) or 500 mA (BL1800), and the maximum K is 30 V. This voltage limit on K arises because R51 and R52 at the base of Q28 can each dissipate 500 mW for a total of 1 W. The 30 V limit then constrains the sinking outputs as well because K is common to all four high-current outputs.

HV3 can also be reconfigured as a sinking output. To do so, remove the 0 Ω surface-mounted resistor R56, and solder on a 0 Ω surface-mounted resistor or jumper wire at R55. If you plan to drive inductive loads, add a diode at D21. Figure 8 shows the location of these components.

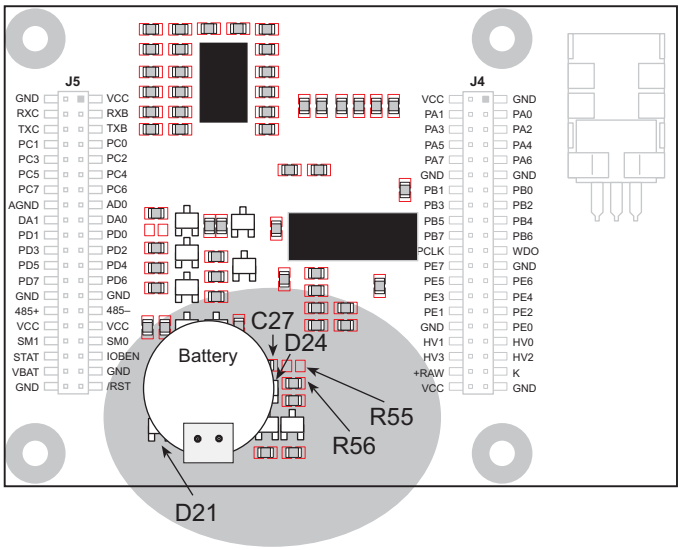


Figure 8. Changing HV3 to a Sinking Output

3.2.3 Bidirectional I/O

The Jackrabbit has 14 CMOS-level bidirectional I/O: PA0–PA7, PD0, PD3, PD6–PD7, and PE4–PE5. The BL1820, which does not have RS-485, has one additional bidirectional I/O, PD5.

3.3 A/D Converter

The analog-to-digital (A/D) converter, shown in Figure 9, compares the DA0 voltage to AD0, the voltage presented to the converter. DA0 therefore cannot be used for the digital-to-analog (D/A) converter when the A/D converter is being used.

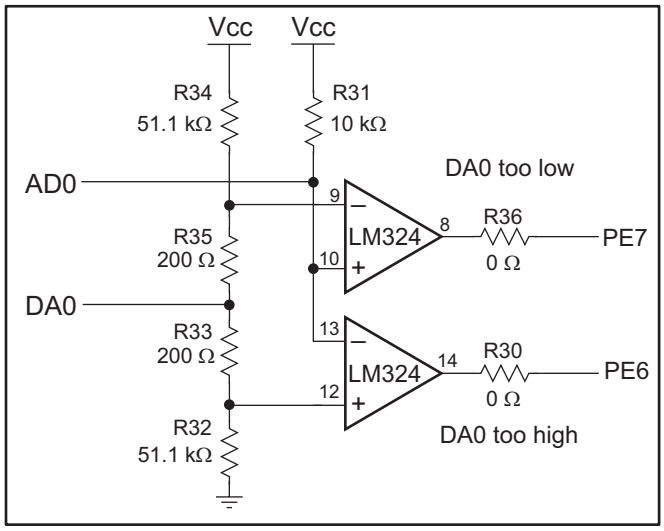


Figure 9. Schematic Diagram of A/D Converter

The A/D converter transforms the voltage at DA0 into a 20 mV window centered around DA0. For example, if DA0 is 2.0 V, the window in the A/D converter would be 1.990 V to 2.010 V. If AD0 > 2.010 V, PE7 would read high and PE6 would read low. If 1.990 V < AD0 < 2.010 V, PE7 would read low and PE6 would read low. This is the case when the A/D input is exactly the same as DA0. If AD0 < 1.990 V, PE7 would read low and PE6 would read high.

PE6 can be imagined to be a “DA0 voltage is too high” indicator. If DA0 is larger than the analog voltage presented at AD0, then PE6 will be true (high). If this happens, the program will need to reduce the DA0 voltage.

PE7 can be imagined to be a “DA0 voltage is too low” indicator. If DA0 is smaller than the analog voltage presented at AD0, then PE7 will be true (high). If this happens, the program will need to raise the DA0 voltage.

The A/D input, AD0, is the same as DA0 only when PE6 and PE7 are low. Because the A/D converter circuit uses a 20 mV window, the accuracy is ±10 mV. DA0 can range from 0.1 V to 2.8 V, which represents 270 steps of ±10 mV. This is better than 8-bit accuracy. Since the D/A converter is able to change the DA0 output in 3.88 mV steps, there are 697 steps over the range from 0.1 V to 2.8 V. This represents a resolution of more than 9 bits.

There is a 10 kΩ resistor, R31, connected between Vcc and AD0. This resistor should provide an appropriate voltage divider bias for a variety of common thermistors so that they can be connected directly between AD0 and ground. The A/D converter load is the 10 kΩ resistor connected to Vcc. Remove R31 if a smaller load is desired—this will lead to a very high input impedance for the A/D converter.

The A/D converter has no reference voltage. There is a relative accuracy between measurements, but no absolute accuracy. This is because Vcc can vary ±5%, the pulse-width modulated outputs might not reach the full 0 V and 5 V rails out of the Rabbit 2000 microprocessor, and the gain resistors used in the circuit have a 1% tolerance. For these reasons, each Jackrabbit needs to be calibrated individually, with the constants held in software, to be able to rely on an absolute accuracy. The Jackrabbit is sold without this calibration support.

The algorithm provided to perform the conversion does a successive approximation search for the analog voltage. This takes an average of 150 ms, and a maximum of 165 ms, with a 14.7 MHz Jackrabbit.

3.4 D/A Converters

Two digital-to-analog (D/A) converter outputs, DA0 and DA1, are supplied on the Jackrabbit. These are shown in Figure 10.

The D/A converters have no reference voltage. Although they may be fairly accurate from one programmed voltage to the next, they do not have absolute accuracy. This is because Vcc can change ±5%, the PWM outputs might not achieve the full 0 V and 5 V rail out of the processor, and the gain resistors in the circuit have a 1% tolerance. The D/A converters therefore need individual calibration, with the calibration constants held in software before absolute accuracy can be relied on. The Jackrabbit is sold without such calibration.

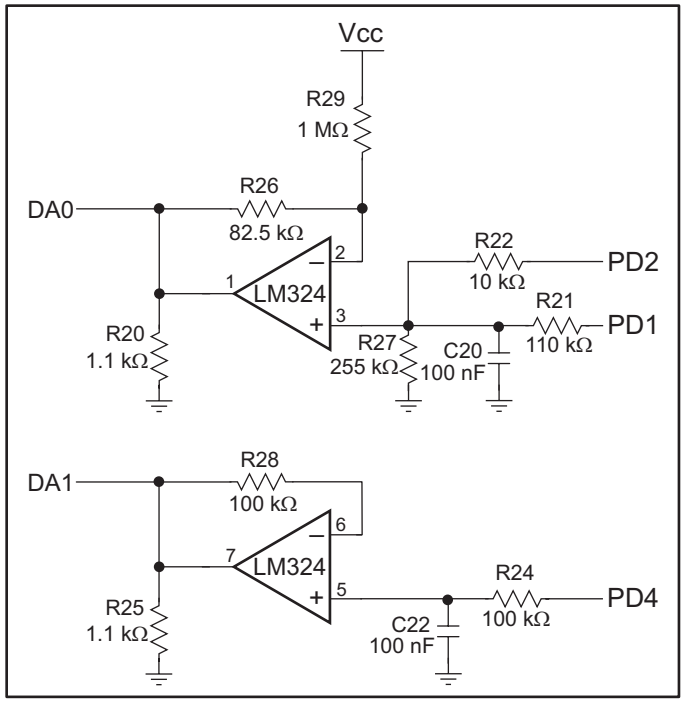


Figure 10. Schematic Diagram of D/A Converters

Note that DA0 is used to provide a reference voltage for the A/D converter and is unavailable for D/A conversion when the A/D converter is being used.

Pulse-width modulation (PWM) is used for the D/A conversion. This means that the digital signal, which is either 0 V or 5 V, is a train of pulses. This means that if the signal is taken to be usually at 0 V (or ground), there will be 5 V pulses. The voltage will be 0 V for a given time, then jump to 5 V for a given time, then back to ground for a given time, then back to 5 V, and so on. A hardware filter in the circuit consisting of a resistor and capacitor averages the 5 V signal and the 0 V signal over time. Therefore, if the time that the signal is at 5 V is equal to the time the signal is 0 V, the duty cycle will be 50%, and the average signal will be 2.5 V. If the time at 5 V is only 25% of the time, then the average voltage will be 1.25 V. Thus, the software needs to only vary the time the signal is at 5 V with respect to the time the signal is at 0 V to achieve any desired voltage between 0 V and 5 V.

It is very easy to do pulse-width modulation with the Rabbit 2000 microprocessor because of the chip's architecture.

3.4.1 DA1

The op amp supporting DA1 converts pulse-width modulated signals to an analog voltage between 0 V and 5 V. A digital signal that varies with time is fed from PD4. The resolution of the DA1 output depends on the smallest increment of time to change the on/off time (the time between 5 V and 0 V). The Jackrabbit uses the Rabbit 2000's Port D control registers to clock out the signal at a timer timeout. The timer used is timer B. Timer B has 10 bits of resolution so that the voltage can be varied in 1/1024 increments. The resolution is thus about 5 mV (5 V/1024).

R28 is present solely to balance the op amp input current bias. R25 helps to achieve a voltage close to ground for a 0% duty cycle.

A design constraint dictates how fast timer B must run. The hardware filter has a resistor-capacitor filter that averages the 0 V and 5 V values. Its effect is to smooth out the digital pulse train. It cannot be perfect, and so there will be some ripple in the output voltage. The maximum signal decay between pulses will occur when DA1 is set to 2.5 V. This means the pulse train will have a 50% duty cycle. The maximum signal decay will be

$$2.5\ \text{V} \times \left[1 - e^{\left(\frac{-t}{RC}\right)}\right]$$

where RC = 0.01 s for 14.74 MHz Jackrabbits, and t is the pulse on or off time (not the length of the total cycle).

Timer B is driven at the Rabbit 2000 frequency divided by 2. The frequency achievable with a 14.74 MHz clock is (14.74 MHz/2)/1024 = 7.17 kHz. This is a period of 1/f = 139 µs. For a 50% duty cycle, half of the period will be high (70 µs at 5 V), and half will be low (70 µs at 0 V). Thus, a 14.74 MHz Jackrabbit has t = 70 µs. Based on the standard capacitor discharge formula, this means that the maximum voltage change will be

$$2.5\ \text{V} \times \left[1 - e^{\left(\frac{-70\ \mu\text{s}}{0.01\ \text{s}}\right)}\right] = 17.4\ \text{mV}$$

This is less than a 20 mV peak-to-peak ripple.

The DA1 output can be less than 100 mV for a 0% duty cycle and above 3.5 V for a 100% duty cycle. Because of software limitations on the low side and hardware limitations on the high side, the duty cycle can only be programmed from 12% to 72%. The low limitation allows the software to perform other tasks as well as maintain the PWM for the D/A converters. The high limitation is simply the maximum voltage obtainable with the LM324 op amp used in the circuit. Anything outside the 12%–72% range gets output as

either a 0% or a 100% duty cycle. The duty cycle is programmed as the high-time count of 1024 total counts of the Rabbit 2000's timer B. Thus, 256 counts would be 25% of 1024 counts, and corresponds to a 25% duty cycle.

Table 2 lists typical DA1 voltages measured for various duty cycle values with a load larger than 1 MΩ.

Table 2. Typical DA1 Voltages for Various Duty Cycles

Duty Cycle (%)	Voltage (V)	Programmed Count
0	0.002	0–122
12	0.620	123
25	1.242	256
50	2.483	512
72	3.567	742
100	3.567	743–1024

It is important to remember that the DA1 output voltage will not be realized instantaneously after programming in a value. There is a settling time because of the RC time constant (R24 × C22), which is 10 ms. For example, the voltage at any given time is

$$V = V_P - (V_P - V_{DA1})e^{(-t/RC)} \qquad \text{(EQ 1)}$$

where V is the voltage at time t, V_P is the programmed voltage, V_{DA1} is the last DA1 output voltage from the D/A converter, and RC is the time constant (10 ms). The settling will be within 99.326% (or within about 21 mV for a 3 V change in voltage) after five time constants, or 50 ms. Six time constants, 60 ms, will allow settling to within 99.75% (or to within about 8 mV for a 3 V change in voltage). Seven time constants, 70 ms, will allow settling to within 99.91% (or to within about 3 mV for a 3 V change in voltage).

An LM324 op amp, which can comfortably source 10 mA throughout the D/A converter range, drives the D/A converter output. If the output voltage is above 1 V, the D/A converter can comfortably sink 10 mA. Below 1 V, the D/A converter can only sink a maximum of 100 µA.

To summarize, DA1 is provided uncalibrated, can be programmed with a resolution of 5 mV and a peak-to-peak ripple less than 20 mV over the range from 0.7 V to 3.5 V and 0 V. The settling time to within 21 mV is 50 ms.

3.4.2 DA0

The op amp supporting DA0 translates a 12%–88% duty cycle to an analog voltage range of 0 V to 3 V. The software operates only within this duty cycle; a duty cycle less than 12% is rounded down to 0%, and any duty cycle above 88% is rounded up to 100%.

DA0 uses a voltage divider that consists of R21 and R27 and a gain-offset circuit that consists of R26 and R29 to achieve the output range of 0 V to 3 V within the software duty cycle.

The DA0 output can be less than 100 mV for a 0% duty cycle and above 3.0 V for a 100% duty cycle. The duty cycle is programmed as the high-time count of 1024 total counts of the Rabbit 2000's timer B. Thus, 256 counts would be 25% of 1024 counts, and corresponds to a 25% duty cycle.

Table 3 lists typical DA0 voltages measured for various duty cycle values with a load larger than 1 MΩ.

Table 3. Typical DA0 Voltages for Various Duty Cycles

Duty Cycle (%)	Voltage (V)	Programmed Count
0	0.074	0–122
12	0.076	123
25	0.530	256
50	1.467	512
75	2.406	768
88	2.875	901
100	3.345	902–1024

It is important to remember that the DA0 output voltage will not be realized instantaneously after programming in a value. There is a settling time because of the RC time constant (R21 || R27 × C20), which is 7.68 ms. For example, the voltage at any given time is

$$V = V_P - (V_P - V_{DA0})e^{(-t/RC)} \qquad \text{(EQ 2)}$$

where V is the voltage at time t, V_P is the programmed voltage, V_{DA0} is the last DA0 output voltage from the D/A converter, and RC is the time constant (7.68 ms). The settling will be within 99.326% (or within about 21 mV for a 3 V change in voltage) after five time constants, or 38 ms. Six time constants, 46 ms, will allow settling to within 99.75% (or to within about 8 mV for a 3 V change in voltage). Seven time constants, 54 ms, will allow settling to within 99.91% (or to within about 3 mV for a 3 V change in voltage).

The settling time is reduced somewhat by precharging capacitor C20 with pulse-width modulation from PD2.

The resolution of the DA0 output depends on the smallest increment of time to change the on/off time (the time between 5 V and 0 V). The Jackrabbit uses the Rabbit 2000's Port D control registers to clock out the signal at a timer timeout. The timer used is timer B. Timer B has 10 bits of resolution so that the voltage can be varied in 1/1024 increments. The resolution is thus about 3.88 mV for the DA0 output voltage range of 0 V to 3 V in the 12%–88% duty cycle.

An LM324 op amp, which can comfortably source 10 mA throughout the D/A converter range, drives the D/A converter output. If the output voltage is above 1 V, the D/A converter can comfortably sink 10 mA. Below 1 V, the D/A converter can only sink a maximum of 100 µA.

The peak-to-peak ripple on DA0 is less than 3 mV. There is a way to get rid of the ripple for very small periods of time. To do that, simply program the PWM port from a PWM ***output*** to a high-impedance ***input***. This will allow the capacitor to hold the voltage subject only to leakage currents, which add up to about 1 µA. This will cause the capacitor to change voltage at the rate of 10 V per second, or 10 mV per millisecond. Practically, this means that the PWM can stop for about 1 ms (seven 1024-count D/A converter cycles on a 14.74 MHz processor clock) with a voltage movement of less than 10 mV.

To summarize, DA0 is provided uncalibrated, can be programmed with a resolution of 3.88 mV and a peak-to-peak ripple less than 3 mV over the range from 0.1 V to 2.8 V and at 3.35 V. The settling time to within 3 mV is 54 ms.

3.5 Serial Communication

The Jackrabbit has two RS-232 (3-wire) serial channels, one RS-485 serial channel, and one synchronous CMOS serial channel.

3.5.1 RS-232

The Jackrabbit's two RS-232 serial channels are connected to an RS-232 transceiver, U4, an industry-standard MAX232 chip. U4 provides the voltage output, slew rate, and input voltage immunity required to meet the RS-232 serial communication protocol. Basically, the chip translates the Rabbit 2000's 0 V to +Vcc signals to ±10 V. Note that the polarity is reversed in an RS-232 circuit so that +5 V is output as –10 V and 0 V is output as +10 V. U4 also provides the proper line loading for reliable communication.

The Rabbit 2000 serial port B signals are presented as RS-232 compliant signals TXB (serial port B transmit) and RXB (serial port B receive) on header J5.

The Rabbit 2000 serial port C signals are presented as RS-232 compliant signals TXC (serial port C transmit) and RXC (serial port C receive) on header J5.

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

Because two RS-232 transmit and two RS-232 receive lines are available, one serial channel can be used for serial transmit and receive, and the other serial channel can be used as a general digital I/O for RTS/CTS handshaking. Although the present release of Dynamic C does not support RTS/CTS handshaking in its libraries, it is possible to write your own software.

3.5.2 RS-485

The Jackrabbit has one RS-485 serial channel, which is connected to the Rabbit 2000 serial port D through U6, an RS-485 transceiver. U6 supports the RS-485 serial communication protocol. The chip's slew rate limiters provide for a maximum baud rate of 250,000 bps. The half-duplex communication uses the Rabbit 2000's PD5 pin to control the data enable on the communication line.

The Jackrabbit can be used in an RS-485 multidrop network. Connect the RS-485+ to RS-485+ and RS-485– to RS-485– using single twisted-pair wires (nonstranded, tinned) as shown in Figure 11.

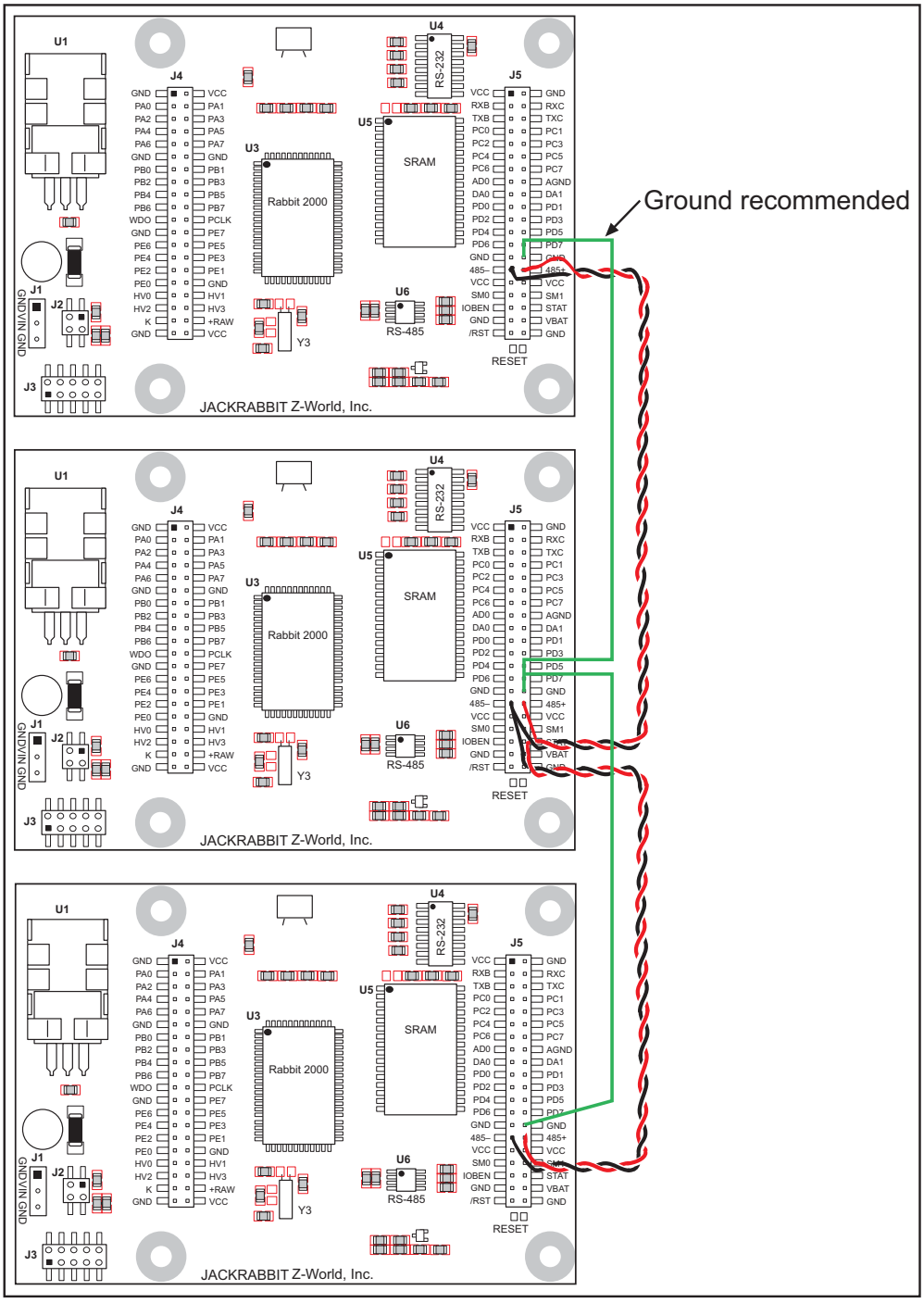


Figure 11. Multidrop Jackrabbit Network

The Jackrabbit comes with a 220 Ω termination resistor and 681 Ω bias resistors already installed, as shown in Figure 12.

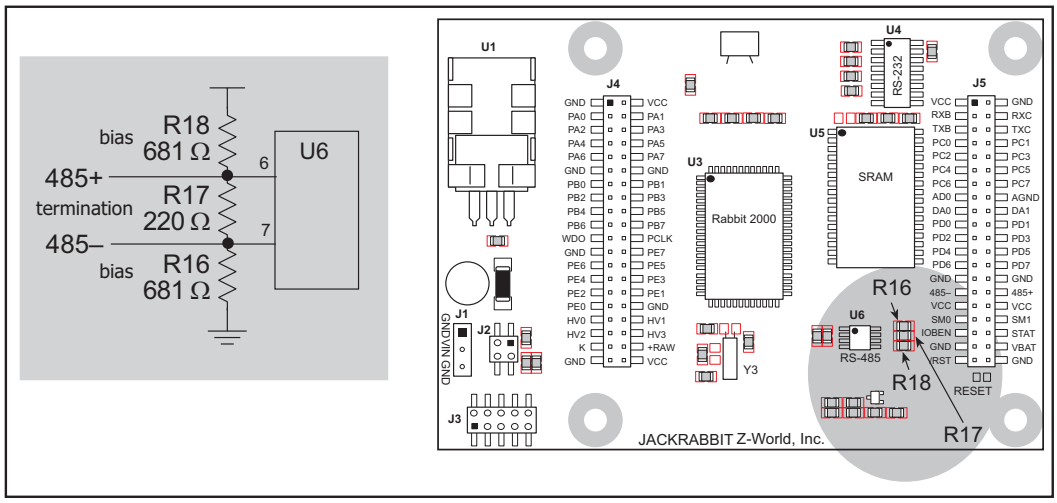


Figure 12. RS-485 Termination and Bias Resistors

The load these bias and termination resistors present to the RS-485 transceiver (U6) limits the number of Jackrabbits in a multidrop network to one master and nine slaves, unless the bias and termination resistors are removed. When using more than 10 Jackrabbits in a multidrop network, leave the 681 Ω bias resistors in place on the master Jackrabbit, and leave the 220 Ω termination resistors in place on the Jackrabbit at each end of the network.

3.5.3 Programming Port

The Jackrabbit has a 10-pin program header labeled J3. The programming port uses the Rabbit 2000's Serial Port A for communication. Dynamic C uses the programming port to download and debug programs.

The programming port is also used for the following operations.

- Cold-boot the Rabbit 2000 after a reset.
- Remotely download and debug a program over an Ethernet connection using the RabbitLink EG2110.
- Fast copy designated portions of flash memory from one Rabbit-based board (the master) to another (the slave) using the Rabbit Cloning Board.

Alternate Uses of the Serial Programming Port

All three clocked Serial Port A signals are available as

- a synchronous serial port
- an asynchronous serial port, with the clock line usable as a general CMOS input

The serial programming port may also be used as a serial port via the **DIAG** connector on the serial programming cable.

In addition to Serial Port A, the Rabbit 2000 startup-mode (SMODE0, SMODE1), status, and reset pins are available on the serial programming port.

The two startup mode pins determine what happens after a reset—the Rabbit 2000 is either cold-booted or the program begins executing at address 0x0000. These two SMODE pins can be used as general inputs once the cold boot is complete.

The status pin is used by Dynamic C to determine whether a Rabbit microprocessor is present. The status output has three different programmable functions:

1. It can be driven low on the first op code fetch cycle.
2. It can be driven low during an interrupt acknowledge cycle.
3. It can also serve as a general-purpose output.

The /RESET_IN pin is an external input that is used to reset the Rabbit 2000 and the onboard peripheral circuits on the RabbitCore module. The serial programming port can be used to force a hard reset on the RabbitCore module by asserting the /RESET_IN signal.

Refer to the ***Rabbit 2000 Microprocessor User's Manual*** for more information.

3.6 Programming Cable

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

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

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

3.6.1 Changing Between Program Mode and Run Mode

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

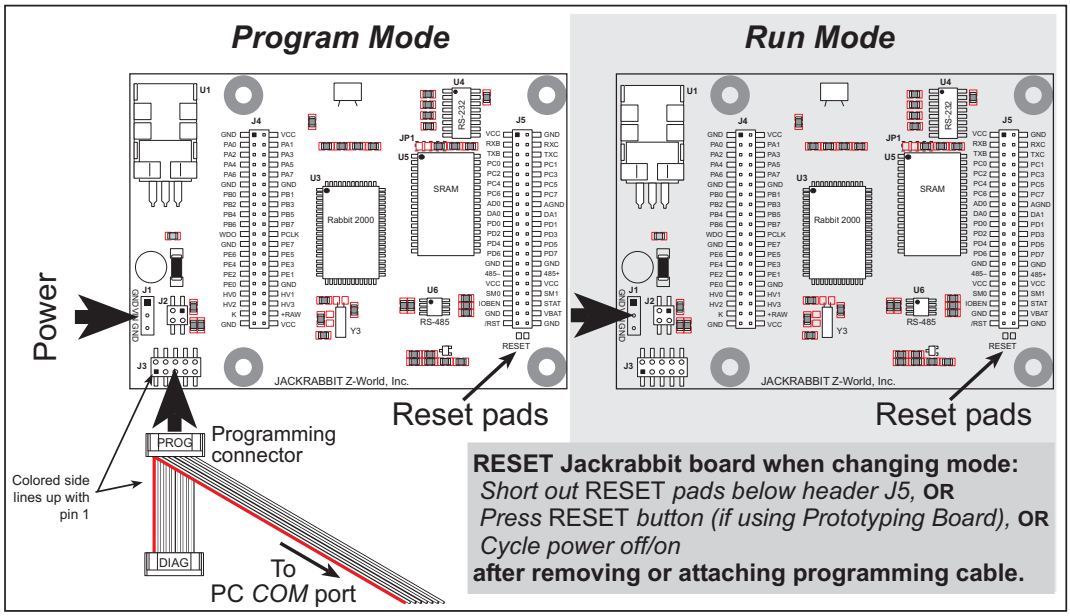


Figure 13. Jackrabbit Program Mode and Run Mode Setup

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

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

3.7 Memory

3.7.1 SRAM

The Jackrabbit is designed to accept 32K to 512K of SRAM packaged in an SOIC case.

Standard Jackrabbit models come with 128K of SRAM. A factory-installed option for 512K of SRAM is available. Figure 14 shows the locations and the jumper settings for the jumpers at JP1 used to set the SRAM size. The "jumpers" are 0 Ω surface-mounted resistors.

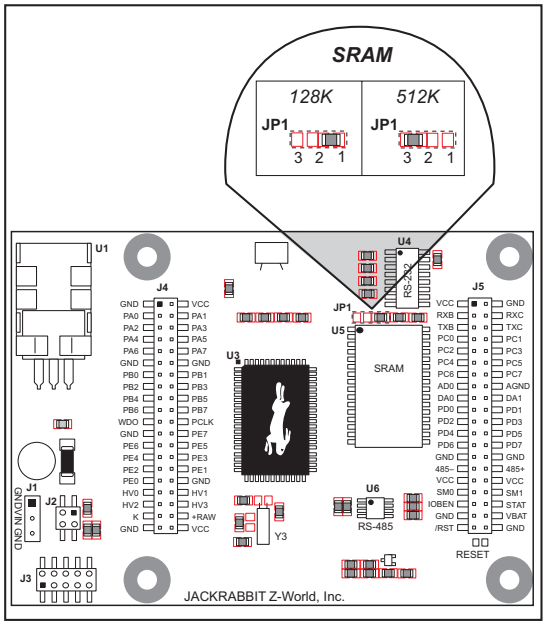


Figure 14. Jackrabbit Jumper Settings for SRAM Size

NOTE: For ordering or other information involving the factory-installed 512K SRAM option, call your Rabbit sales representative or distributor.

No 0 Ω surface-mounted resistors are installed at JP1 for 32K SRAM.

3.7.2 Flash EPROM

The Jackrabbit is also designed to accept 128K to 512K of flash EPROM packaged in a TSOP case.

NOTE: Rabbit 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 exists at JP3 with 0 Ω surface-mounted resistors for Jackrabbit boards labeled 175-0255. 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 Technical Note 218, ***Implementing a Serial Download Manager for a 256K Flash***, for details.

3.8 Other Hardware

3.8.1 External Interrupts

Jackrabbit boards that carry the CE mark have external interrupts available on digital inputs PE4 and PE5.

3.8.2 Clock Doubler

Jackrabbit BL1810 and BL1820 models take advantage of the Rabbit 2000 microprocessor's internal clock doubler. A built-in clock doubler allows half-frequency crystals to be used to reduce radiated emissions. The 14.74 MHz frequency is generated using a 7.37 MHz resonator. The clock doubler is disabled automatically in the BIOS for crystals or resonators with a frequency above 12.9 MHz.

The clock doubler may be disabled if 14.74 MHz clock speeds are not required. Disabling the Rabbit 2000 microprocessor's internal clock doubler will reduce power consumption and further reduce radiated emissions. Disable the clock doubler by adding a simple configuration macro as shown below.

1. Select the "Defines" tab from the Dynamic C **Options > Project Options** menu.
2. Add the line **CLOCK_DOUBLED=0** to always disable the clock doubler.

   The clock doubler is enabled by default, and usually no entry is needed. If you need to specify that the clock doubler is always enabled, add the line **CLOCK_DOUBLED=1** to always enable the clock doubler. The clock speed will be doubled as long as the crystal frequency is less than or equal to 26.7264 MHz.
3. Click **OK** to save the macro. The clock doubler will now remain off whenever you are in the project file where you defined the macro.

3.8.3 Spectrum Spreader

Jackrabbit boards that carry the CE mark have a Rabbit 2000 microprocessor that features a spectrum spreader, which helps to mitigate EMI problems. By default, the spectrum spreader is on automatically for Jackrabbit BL1810 and BL1820 boards that carry the CE mark when used with Dynamic C 7.32 or later versions, but the spectrum spreader may also be turned off or set to a stronger setting. The spectrum spreader settings may be changed through a simple configuration macro as shown below.

1. Select the "Defines" tab from the Dynamic C **Options > Project Options** menu.
2. Normal spreading is the default, and usually no entry is needed. If you need to specify normal spreading, add the line

   ```
   ENABLE_SPREADER=1
   ```

   For strong spreading, add the line

   ```
   ENABLE_SPREADER=2
   ```

   To disable the spectrum spreader, add the line

   ```
   ENABLE_SPREADER=0
   ```

   NOTE: The strong spectrum-spreading setting is not recommended since it may limit the maximum clock speed or the maximum baud rate. It is unlikely that the strong setting will be used in a real application.
3. Click **OK** to save the macro. The spectrum spreader will now remain off whenever you are in the project file where you defined the macro.

The spectrum spreader is off by default for Jackrabbit BL1800 models, and needs to be enabled for them to be CE-compliant. To allow the flash memory and RAM chips to accommodate the occasional higher frequencies associated with the spectrum spreader being turned on for the Jackrabbit BL1800 models only, you will need at least one wait state for both the flash memory and the RAM. The strong spectrum-spreading setting is not needed for any Jackrabbit board.

The number of wait states is specified in the following code from the **LIB\BOARDTYPES** library. (There are 0 wait states by default.)

```
#ifndef NUM_RAM_WAITST
#define NUM_RAM_WAITST 0
#endif
#ifndef NUM_FLASH_WAITST
#define NUM_FLASH_WAITST 0
#endif
```

There is no spectrum spreader functionality for Jackrabbit boards that do not carry the CE mark or when using any Jackrabbit with a version of Dynamic C prior to 7.30.

4. SOFTWARE REFERENCE

To develop and debug programs for the Jackrabbit (and for all other Rabbit hardware), you must install and use Dynamic C. It runs on an IBM-compatible PC and is designed for use with Rabbit-based single-board computers and other devices based on the Rabbit microprocessor. This chapter provides a tour of the major features of Dynamic C with respect to the Jackrabbit.

4.1 An Overview of Dynamic C

Dynamic C has been in use worldwide since 1989. It is specially designed for programming embedded systems, and features quick compile and interactive debugging. A complete reference guide to Dynamic C is contained in the ***Dynamic C User's Manual***.

You have a choice of doing your software development in the flash memory or in the data SRAM included on the Jackrabbit. The flash memory and SRAM options are selected with the **Options > Project Options > Compiler** menu.

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

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

NOTE: Do not depend on the flash memory sector size or type. Due to the volatility of the flash memory market, the Jackrabbit and Dynamic C were designed to accommodate flash devices with various sector sizes.

Developing software with Dynamic C is simple. Users can write, compile, and test C and assembly code without leaving the Dynamic C development environment. Debugging occurs while the application runs on the target. Alternatively, users can compile a program to an image file for later loading. Dynamic C runs on PCs under Windows NT and later—note that Dynamic C is still being evaluated for compatibility with Windows Vista at the time of writing, and should not be expected to run correctly under Windows Vista at this time. Programs can be downloaded at baud rates of up to 460,800 bps after the program compiles.

Dynamic C has a number of standard features:

- Full-feature source and/or assembly-level debugger, no in-circuit emulator required.
- Royalty-free TCP/IP stack with source code and most common protocols.
- Hundreds of functions in source-code libraries and sample programs:
  - Exceptionally fast support for floating-point arithmetic and transcendental functions.
  - RS-232 and RS-485 serial communication.
  - Analog and digital I/O drivers.
  - I^2C, SPI, GPS, file system.
  - LCD display and keypad drivers.
- Powerful language extensions for cooperative or preemptive multitasking
- Loader utility program to load binary images into Rabbit-based targets in the absence of Dynamic C.
- Provision for customers to create their own source code libraries and augment on-line help by creating "function description" block comments using a special format for library functions.
- Standard debugging features:
  - Breakpoints—Set breakpoints that can disable interrupts.
  - Single-stepping—Step into or over functions at a source or machine code level, µC/OS-II aware.
  - Code disassembly—The disassembly window displays addresses, opcodes, mnemonics, and machine cycle times. Switch between debugging at machine-code level and source-code level by simply opening or closing the disassembly window.
  - Watch expressions—Watch expressions are compiled when defined, so complex expressions including function calls may be placed into watch expressions. Watch expressions can be updated with or without stopping program execution.
  - Register window—All processor registers and flags are displayed. The contents of general registers may be modified in the window by the user.
  - Stack window—shows the contents of the top of the stack.
  - Hex memory dump—displays the contents of memory at any address.
  - **STDIO** window—`printf` outputs to this window and keyboard input on the host PC can be detected for debugging purposes. `printf` output may also be sent to a serial port or file.

4.2 Sample Programs

Sample programs are provided in the Dynamic C **SAMPLES** folder.

The various folders contain specific sample programs that illustrate the use of the corresponding Dynamic C libraries. For example, the sample program **PONG.C** demonstrates the output to the Dynamic C **STDIO** window.

The **SAMPLES\JACKRAB** folder contains sample programs that illustrate features unique to the Jackrabbit.

Follow the instructions included with the sample program to connect the Jackrabbit and the other hardware identified in the instructions.

To run a sample program, open it with the **File** menu (if it is not still open), then compile and run it by selecting **Run** in the **Run** menu (or press **F9**). The Jackrabbit must be in Program Mode (programming cable is attached to programming port as shown in Figure 2) and must be connected to a PC using the programming cable.

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

The sample programs are listed in Table 4.

Table 4. Jackrabbit Sample Programs

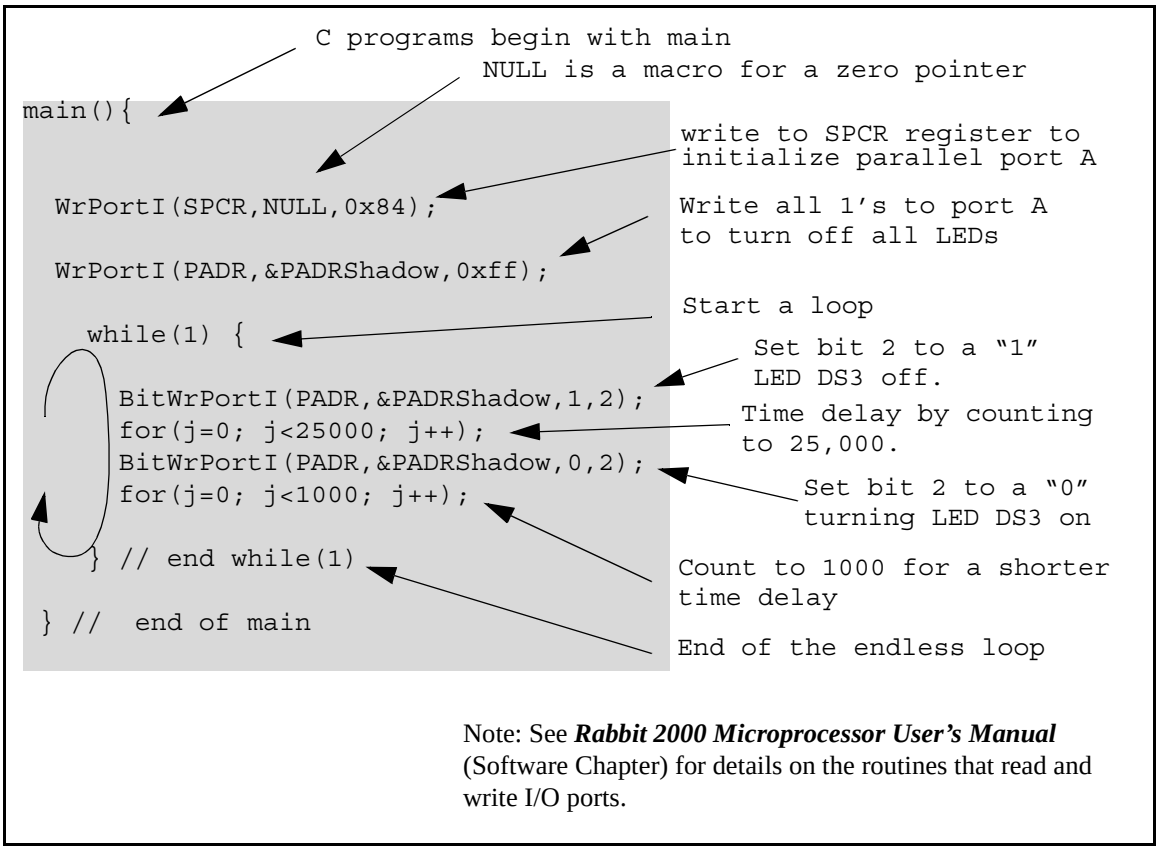
DEMOJR1.C
DEMOJR2.C
DEMOJR3.C
JRIOTEST.C
JR_FLOWCONTROL.C
JR_PARITY.C
JRIO_COF.C
LCD_DEMO.C
RABDB01.C
RABDB02.C

The first five sample programs provide a step-by-step introduction to the Jackrabbit board. Additional sample programs illustrate more advanced topics.

4.2.1 DEMOJR1.C

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

First, open the file **DEMOJR1.C**, which is in the **SAMPLES/JACKRAB** folder. The program will appear in a window, as shown in Figure 15 below (minus some comments). Use the mouse to place the cursor on the function name ***WrPortI*** in the program and type **<Ctrl-H>**. This will bring up a documentation box for the function **WrPortI**. In general, you can do this with all functions in Dynamic C libraries, including libraries that you write yourself—the ***Dynamic C User's Manual*** provides more information. Close the documentation box and continue.



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

Figure 15. Sample Program DEMOJR1.C

To run the program **DEMOJR1.C**, load it with the **File > Open** menu, then compile and run it by pressing **F9** or by selecting **Run** in the **Run** menu. The LED on the Prototyping Board should start flashing if everything went well. If this doesn't work review the following points.

- The target should be ready, which is indicated by the message "BIOS successfully compiled..." If you did not receive this message or you get a communication error, recompile the BIOS by typing **<Ctrl-Y>** or select **Recompile BIOS** from the **Compile** menu.
- A message reports that "No Rabbit processor detected" in cases where the Jackrabbit and Prototyping Board are not connected together, the wall transformer is not connected, or is not plugged in. (The red power LED lights whenever power is connected.)

- The programming cable must be connected to the Jackrabbit board. (The colored wire on the programming cable is closest to pin 1 on header J3 on the Jackrabbit board, as shown in Figure 2.) The other end of the programming cable must be connected to the PC serial port. The COM port specified in the Dynamic C **Options** menu must be the same as the one the programming cable is connected to.
- To check if you have the correct serial port, select **Compile**, then **Compile BIOS**, or type **<Ctrl-Y>**. If the "BIOS successfully compiled …" message does not display, try a different serial port using the Dynamic C **Options** menu until you find the one you are plugged into. Don't change anything in this menu except the COM number. The baud rate should be 115,200 bps and the stop bits should be 1.

Single-Stepping

Compile or re-compile **DEMOJR1.C** by clicking the **Compile** button on the task bar. The program will compile and the screen will come up with a highlighted character (green) at the first executable statement of the program. Use the **F8** key to single-step. Each time the **F8** key is pressed, the cursor will advance one statement. When you get to the **for(j=0, j< ...** statement, it becomes impractical to single-step further because you would have to press **F8** thousands of times. We will use this statement to illustrate watch expressions.

Watch Expression

Type **<Ctrl-W>** or chose **Add/Del Watch Expression** in the **Inspect** menu. A box will come up. Type the lower case letter j and click on *add to top* and *close*. Now continue single-stepping with **F8**. Each time you step, the watch expression (**j**) will be evaluated and printed in the watch window. Note how the value of **j** advances when the statement **j++** is executed.

Break Point

Move the cursor to the start of the statement:

```
for(j=0; j<1000; j++);
```

To set a break point on this statement, type **F2** or select **Breakpoint** from the **Run** menu. A red highlight will appear on the first character of the statement. To get the program running at full speed, type **F9** or select **Run** on the **Run** menu. The program will advance until it hits the break point. Then the break point will start flashing and show both red and green colors. Note that LED DS3 is now solidly turned on. This is because we have passed the statement turning on LED DS3. Note that **j** in the watch window has the value 25000. This is because the loop above terminated when **j** reached 25000.

To remove the break point, type **F2** or select **Toggle Breakpoint** on the **Run** menu. To continue program execution, type **F9** or select **Run** from the **Run** menu. Now the LED should be flashing again since the program is running at full speed.

You can set break points while the program is running by positioning the cursor to a statement and using the **F2** key. If the execution thread hits the break point, a break point will take place. You can toggle the break point off with the **F2** key and continue execution with the **F9** key. Try this a few times to get the feel of things.

Editing the Program

Click on the **Edit** box on the task bar. This will set Dynamic C into the edit mode so that you can change the program. Use the **Save as** choice on the **File** menu to save the file with a new name so as not to change the demo program. Save the file as **MYTEST.C**. Now change the number 25000 in the **for (..** statement to 10000. Then use the **F9** key to recompile and run the program. The LED will start flashing, but it will flash much faster than before because you have changed the loop counter terminal value from 25000 to 10000.

Watching Variables Dynamically

Go back to edit mode (select edit) and load the program **DEMOJR2.C** using the **File** menu **Open** command. This program is the same as the first program, except that a variable **k** has been added along with a statement to increment **k** each time around the endless loop. The statement:

```
runwatch();
```

has been added. This is a debugging statement that makes it possible to view variables while the program is running.

Use the **F9** key to compile and run **DEMOJR2.C**. Now type **<Ctrl-W>** to open the watch window and add the watch expression **k** to the top of the list of watch expressions. Now type **<Ctrl-U>**. Each time you type **<Ctrl-U>**, you will see the current value of **k**, which is incrementing about 5 times a second.

As an experiment add another expression to the watch window:

```
k*5
```

Then type **<Ctrl-U>** several times to observe the watch expressions **k** and **k*5**.

Summary of Features

So far you have practiced using the following features of Dynamic C.

- Loading, compiling and running a program. When you load a program it appears in an edit window. You can compile by selecting **Compile** on the task bar or from the **Compile** menu. When you compile the program, it is compiled into machine language and downloaded to the target over the serial port. The execution proceeds to the first statement of main where it pauses, waiting for you to command the program to run, which you can do with the **F9** key or by selecting **Run** on the **Run** menu. If want to compile and start the program running with one keystroke, use **F9**, the run command. If the program is not already compiled, the run command will compile it first.
- Single-stepping. This is done with the **F8** key. The **F7** key can also be used for single-stepping. If the **F7** key is used, then descent into subroutines will take place. With the **F8** key the subroutine is executed at full speed when the statement that calls it is stepped over.

- Setting break points. The **F2** key is used to turn on or turn off (toggle) a break point at the cursor position if the program has already been compiled. You can set a break point if the program is paused at a break point. You can also set a break point in a program that is running at full speed. This will cause the program to break if the execution thread hits your break point.
- Watch expressions. A watch expression is a C expression that is evaluated on command in the watch window. An expression is basically any type of C formula that can include operators, variables and function calls, but not statements that require multiple lines such as *for* or *switch*. You can have a list of watch expressions in the watch window. If you are single-stepping, then they are all evaluated on each step. You can also command the watch expression to be evaluated by using the **<Ctrl-U>** command. When a watch expression is evaluated at a break point, it is evaluated as if the statement was at the beginning of the function where you are single-stepping. If your program is running you can also evaluate watch expressions with a **<Ctrl-U>** if your program has a **`runwatch()`** command that is frequently executed. In this case, only expressions involving global variables can be evaluated, and the expression is evaluated as if it were in a separate function with no local variables.

4.2.2 Other Sample Programs Illustrating Digital I/O

- **DEMOJR2.C**—repeatedly flashes LED DS3 (which is controlled by PA2) on the Prototyping Board.

  This sample program also illustrates the use of the **runwatch()** function to allow Dynamic C to update watch expressions while running. To test this:

  1. Add a watch expression for "k" under "Inspect:Add/Del Watch Expression."
  2. Click "Add to top" so that it will be permanently in the watch list.
  3. While the program is running, type **<Ctrl+U>** to update the watch window.

- **DEMOJR3.C**—demonstrates the use of costatements to LED DS4 (which is controlled by PA3) on the Prototyping Board. This sample program will also watch button S1 (PB2) and toggle LED DS1 (which is controlled by PA0) on/off when pressed. Note that S1 presses are debounced by the software.

  The pins on Parallel Port A can all be set as either outputs or as inputs via the slave port control register (SPCTR). Do not use Parallel Port A if the slave port is being used.

  Bits 0–5 on Parallel Port B are always inputs, and bits 6–7 are always outputs. Do not use Parallel Port B if the slave port is being used.

- **JRIOTEST.C**—exercises the JackRabbit's four digital output channels, the one analog input channel, and the two analog output channels.

- **JRIO_COF.C**—demonstrates the use of cofunctions with the analog input driver. Before you run this sample program, connect DA1 to AD0 on header J7 of the Prototyping Board to provide an input voltage. Once the sample program is running, it will read the input voltage ten times while another costatement is executed concurrently. The values will be printed out in the Dynamic C **STDIO** window at the end of the program.

Before running the **RABDB01.C** and the **RABDB02.C** sample programs, you will need to install 3 mm LEDs such as the Vishay Telefunken TLUR4400 at DS5–DS8 on the Jackrabbit Prototyping Board. These LEDs are included with the Jackrabbit Development Kit.

- **RABDB01.C**—flashes LEDs DS5–DS8 on the Jackrabbit Prototyping Board (which are connected to PA4–PA7) when corresponding switches S1–S4 (which are connected to PB2–PB5) are pressed. The buzzer, which is driven by HV0 from PE0, will also sound whenever switch S1 switch is pressed.

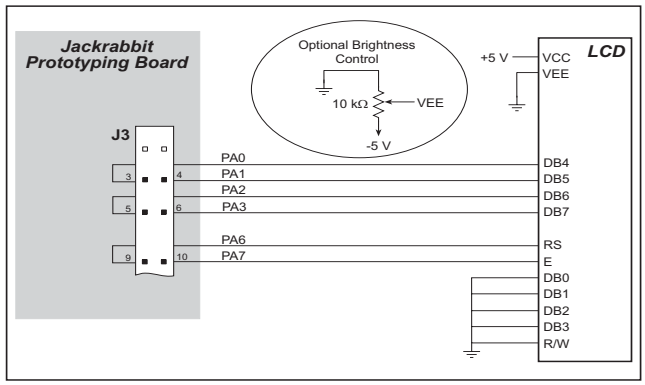
- **RABDB02.C**—flashes LEDs DS5–DS8 on the Jackrabbit Prototyping Board (which are connected to PA4–PA7) when corresponding switches S1–S4 (which are connected to PB2–PB5) are pressed. The buzzer, which is driven by HV0 from PE0, will also sound whenever switch S1 switch is pressed.

Before running the **LCD_DEMO.C** sample program, you will need an LCD based on the HD44780 (or an equivalent) controller.

- **LCD_DEMO.C**—demonstrates a 4-bit interface to an LCD based on the HD44780 (or an equivalent) controller.

  Connect the LCD to Parallel Port A.

  PA0—LCD DB4
  PA1—LCD DB5
  PA2—LCD DB6
  PA3—LCD DB7
  PA6—LCD RS Register Select (0 = command, 1 = data)
  PA7—LCD E:normally low, latches on high to low transition

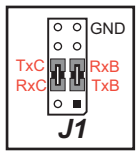


  The R/W pin and DB0–DB3 on the LCD are grounded. DB0–DB3 are not used with a 4-bit interface. This sample program only involves writing, and that is why we ground the R/W pin. VEE is used to control brightness—the simplest thing to do is ground it. If you need to control the brightness, you can connect a potentiometer between ground and -5 V, with the arm going to the VEE pin. Check the specs on the LCD before doing this as some LCDs may require a different connection. VCC is +5 V and VSS is ground.

4.2.3 RS-232 Serial Communication Sample Programs

- **JR_FLOWCONTROL.C**—This program demonstrates hardware flow control by configuring Serial Port C (PC3/PC2) for CTS/RTS with serial data coming from TxB at 115,200 bps. One character at a time is received and is displayed in the **STDIO** window.

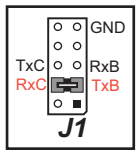
  To set up the Prototyping Board, you will need to tie PC4 and PC5 (TxB and RxB) together at header J1, and you will also tie PC2 and PC3 (TxC and RxC) together as shown in the diagram.

  A repeating triangular pattern should print out in the **STDIO** window. The program will periodically switch flow control on or off to demonstrate the effect of no flow control.

  Refer to the **serBflowcontrolOn()** function call in the ***Dynamic C Function Reference Manual*** for a general description on how to set up flow control lines.

- **JR_PARITY.C**—This program demonstrates the use of parity modes by repeatedly sending byte values 0–127 from Serial Port B to Serial Port C. The program will switch between generating parity or not on Serial Port B. Serial Port C will always be checking parity, so parity errors should occur during every other sequence.

  To set up the Prototyping Board, you will need to tie PC4 and PC3 (TxB and RxC) together at header J1 as shown in the diagram.

  The Dynamic C **STDIO** window will display the error sequence.

4.2.4 RS-485 Serial Communication Sample Program

The following sample program illustrates the use of the RS-485 serial drivers. The sample program shows a byte being transmitted, and then the RS-485 transceiver waits for a reply.

```
#define DINBUFSIZE  15
#define DOUTBUFSIZE 15

void main( void ){
   int nEcho,nReply;
   char cChar;
   Jr485Init ();// Init RS485 Control (PD5)
   serDopen ( 9600 );// Open Serial Port D
   for (;;) {// Forever
      for (cChar='a';cChar<='z';++cChar){
                              // Send Alphabet
         Jr485Tx ();// Enable RS485 Transmitter
         serDputc ( cChar );// Send Byte
         while ((nEcho = serDgetc ()) == -1);
                     // Wait for Echo
         Jr485Rx ();// Disable RS485 Transmitter
         while ((nReply = serDgetc ()) == -1);
                     // Wait for Reply
         printf ( "%02x -> %02x\n",nEcho,nReply );
      }
   }
}
```

NOTE: If your version of Dynamic C is earlier than 6.55, see Technical Note 117, ***Jackrabbit (BL1800 Series) RS-485 Bulletin***, for information on restrictions on using both DA1 and RS-485 at the same time.

4.3 Cooperative Multitasking

Cooperative multitasking is a convenient way to perform several different tasks at the same time. An example would be to step a machine through a sequence of steps and at the same time independently carry on a dialog with the operator via a human interface. Cooperative multitasking differs from a different approach called preemptive multitasking. Dynamic C supports both types of multitasking. In cooperative multitasking each separate task voluntarily surrenders its compute time when it does not need to perform any more activity immediately. In preemptive multitasking control is forcibly removed from the task via an interrupt.

Dynamic C has language extensions to support multitasking. The major C constructs are called *costatements, cofunctions,* and *slicing*. These are described more completely in the ***Dynamic C User's Manual***. The example below, sample program **`DEMOJR3.C`**, uses costatements. A costatement is a way to perform a sequence of operations that involve pauses or waits for some external event to take place. A complete description of costatements is in the ***Dynamic C User's Manual***. The **`DEMOJR3.C`** sample program has two independent tasks. The first task flashes LED DS4 once a second. The second task uses button S1 on the Prototyping Board to toggle the logical value of a virtual switch, **`vswitch`**, and flash DS1 each time the button is pressed. This task also debounces button S1.

```
int vswitch;             // state of virtual switch controlled by button
S1
main(){                  // begin main program
                         // set up parallel port A as output
    WrPortI(SPCR,NULL,0x84);
    WrPortI(PADR,&PADRShadow,0xff);       // turn off all LEDs
    vswitch=0;                            // initialize virtual switch off
(1) while (1) {                           // Endless loop
      BigLoopTop();                     // Begin a big endless loop

    // first task flash LED DS4 every second for 200 milliseconds

(2)   costate {                           // begin a costatement
        BitWrPortI(PADR,&PADRShadow,0,3); // LED DS4 on
(3)     waitfor(DelayMs(200));            // light on for 200 ms
        BitWrPortI(PADR,&PADRShadow,1,3); // LED DS4 off
        waitfor(DelayMs(800));            // light off for 800 ms
(4)   }                                   // end of costatement

    // second task - debounce switch #1 and toggle virtual switch vswitch

    // check button 1 and toggle vswitch on or off

      costate {
(5)     if(BitRdPortI(PBDR,2)) abort;  // if button not down skip out
        waitfor(DelayMs(50));          // wait 50 ms
        if(BitRdPortI(PBDR,2)) abort;  // if button not still down skip
out
        vswitch=!vswitch;   // toggle virtual switch- button was down 50
ms
        while (1) {                    // wait for button to be off 200 ms
          waitfor(BitRdPortI(PBDR,2)); // wait for button to go up
          waitfor(DelayMs(200));       // wait for 200 milliseconds
          if(BitRdPortI(PBDR,2)) break;// if button up break
        }                              // end of while(1)
      }                                // end of costatement

    // make LED agree with vswitch if vswitch has changed

(6)   if( (PADRShadow & 1) == vswitch) {
        BitWrPortI(PADR,&PADRShadow,!vswitch,0);
      )
(7) }                              // end of while loop, go back to start
}                                   // end of main, never come here
```

The numbers in the left margin are reference indicators and are not a part of the code. Load and run the program. Note that LED DS4 flashes once per second. Push button S1 several times and note how LED DS1 is toggled.

The flashing of LED DS4 is performed by the costatement starting at the line marked (2). Costatements need to be executed regularly, often at least every 25 ms. To accomplish this, the costatements are enclosed in a *while* loop. The term while loop is used as a handy way to describe a style of real-time programming in which most operations are done in one loop. The while loop starts at (1) and ends at (7). The function **`BigLoopTop()`** is

used to collect some operations that are helpful to do once on every pass through the loop. Place the cursor on this function name **BigLoopTop()** and hit **<Ctrl-H>** to learn more.

The statement at (3) waits for a time delay, in this case 200 ms. The costatement is being executed on each pass through the big loop. When a **waitfor** condition is encountered the first time, the current value of **MS_TIMER** is saved and then on each subsequent pass the saved value is compared to the current value. If a **waitfor** condition is not encountered, then a jump is made to the end of the costatement (4), and on the next pass of the loop, when the execution thread reaches the beginning of the costatement, execution passes directly to the **waitfor** statement. Once 200 ms has passed, the statement after the waitfor is executed. The costatement has the property that it can wait for long periods of time, but not use a lot of execution time. Each costatement is a little program with its own statement pointer that advances in response to conditions. On each pass through the big loop, as little as one statement in the costatement is executed, starting at the current position of the costatement's statement pointer. Consult the ***Dynamic C User's Manual*** for more details.

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

At (6) a use for a shadow register is illustrated. A shadow register is used to keep track of the contents of an I/O port that is write only - it can't be read back. If every time a write is made to the port the same bits are set in the shadow register, then the shadow register has the same data as the port register. In this case a test is made to see the state of the LED and make it agree with the state of vswitch. This test is not strictly necessary, the output register could be set every time to agree with **vswitch**, but it is placed here to illustrate the concept of a shadow register.

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

4.3.1 Advantages of Cooperative Multitasking

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

4.4 Jackrabbit Function Calls

4.4.1 I/O Drivers

The Jackrabbit contains four high-power digital output channels, two D/A converter output channels, and one A/D converter input channel. These I/O channels can be accessed using the functions found in the **JRIO.LIB** library.

4.4.1.1 Initialization

The function **jrioInit()** must be called before any other function from the **JRIO.LIB** library. This function initializes the digital outputs and sets up the driver for the analog input/outputs. The digital outputs correspond to the Rabbit processor's port E bits 0–3, and the analog I/O uses timer B; bits 1, 2, and 4 of port D; and bits 6 and 7 of port E.

The function void **jrioInit()** initializes the I/O drivers for Jackrabbit. In particular, it sets up parallel port D bits 1, 2, and 4 for analog output, port E bits 0–3 for digital output, and starts up the pulse-width modulation routines for the A/D and D/A channels. Note that these routines can consume up to 20% of the CPU's processing power; the routines use timer B and the B1 and B2 match registers.

4.4.1.2 Digital Outputs

The Jackrabbit contains four high-power digital output drivers, HV0–HV3, on header J4. These can be turned on and off with the following functions from the library **JRIO.LIB**.

HV0, HV1, and HV2 are open-collector sinking outputs, and are able to sink up to 1 A (200 mA for the Jackrabbit and BL1820) from a 30 V source connected to the K line on header J4. HV3 is a sourcing output that is able to source up to 500 mA (100 mA for the BL1810 and BL1820) from a 30 V source connected to the K line.

CAUTION: Remember to cut the trace between K and Vcc inside the outline for header JP2 on the top side of the Prototyping Board if you are supplying K from a separate power supply. An exacto knife, a precision grinder tool, or a screwdriver may be used to cut through the traces as shown in Figure 16.

NOTE: Failure to do this could lead to the destruction of the Rabbit 2000 microprocessor and other components once the Jackrabbit is connected to the Prototyping Board.

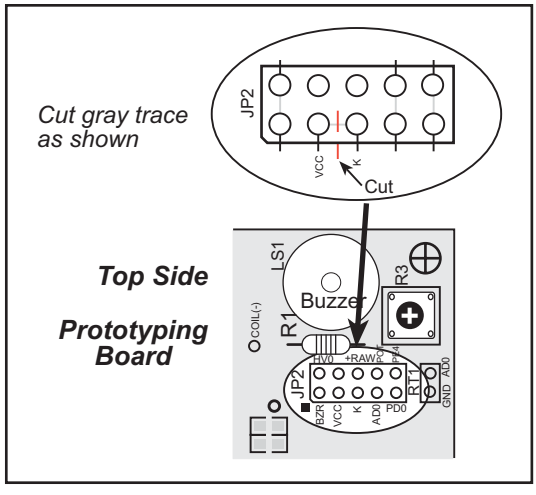


Figure 16. Cut Trace on Prototyping Board When Vcc and K Are Different

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

sets the state of a digital output bit.

jrioInit must be called first.

channel is the output channel number (0-3 on the Jackrabbit).

value is the output value (0 or 1).

```
void digOn(int channel);
```

sets the state of a digital output bit to on (1).

jrioInit must be called first.

channel is the output channel number (0–3 on the Jackrabbit).

```
void digOff(int channel);
```

sets the state of a digital output bit to off (0).

jrioInit must be called first.

channel is the output channel number (0–3 on the Jackrabbit).

NOTE: See the sample program **JRIOTEST.C** for an example of using the digital output functions.

4.4.1.3 Analog Output

The two analog output channels on the Jackrabbit (DA0 and DA1 on header J5) are controlled by a pulse-width modulation (PWM) driver. This requires the use of some fraction of the CPU cycles when the driver is running (up to 20% when both D/A channels are used). A voltage is selected by giving a value from 0 to 1024 to the driver, corresponding roughly to 0.1 V to 3.5 V on DA0. Because of the PWM interrupt frequency, the PWM driver can provide a continuous range of voltage output in the range from 0.1 V to 3.0 V for DA0, and 0.6 V to 3.6 V for DA1. These ranges can be specified with the constants **PWM_MIN**, **PWM_MAX0**, and **PWM_MAX1**. In other words, setting channel DA0 to the value **PWM_MIN** will output 0.1 V, and setting it to **PWM_MAX0** will output 3.0 V. Similarly, setting DA1 to **PWM_MIN** will output 0.6 V, and setting it to **PWM_MAX1** will output 3.6 V. Values below **PWM_MIN** will be rounded down to 0, and values above **PWM_MAX0** (**PWM_MAX1** for DA1) will be rounded up to 1024.

The output channels can also be set in an “always on” or “always off” mode, which does not require CPU cycles. The “always on” mode is set by requesting an output value of 1024, and will provide about 3.4 V on channel DA0, and 3.6 V on DA1. The “always off” mode is selected by asking for a value of 0, and provides an output of around 0.1 V on DA0 and 0.0 V on DA1.

See Table 5 for a summary of the possible analog output voltages corresponding to values given in the **anaOut** function.

Table 5. Typical Analog Output Voltages Corresponding to Values in **anaOut** *Function*

Channel	0	PWM_MIN	PWM_MAX	1024
DA0	0.08 V	0.08 V	2.875 V	3.4 V
DA1	0.004 V	0.63 V	3.6 V	3.6 V

The output value is set using the following function.

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

sets the state of an analog output channel.

jrioInit must be called first.

channel is the output channel number (0 or 1 on the Jackrabbit).

value is an integer from 0–1024 that corresponds to an output voltage as shown in Table 5.

NOTE: See the sample program **JRIOTEST.C** for examples of using the **anaOut** function.

Effect of Interrupts on Analog I/O

The stability of the voltage output (and hence the voltage input determination as well) depends on the ability of the driver to respond quickly to interrupt requests. Dynamic C debugging, use of the **printf** function, or any serial communications can disrupt the pulse-width modulation utilized by the driver and cause fluctuations in the voltage outputs. Avoid using serial communications or **printf** statements during portions of your program where the voltage must remain steady. Also be aware that debugging and running Dynamic C in polling mode will cause fluctuations. Finally, be certain to disable the PWM drivers by setting the output values to 0 or 1024 when you are done using them to free up the CPU.

Calibration of Values to Voltages

The analog output channels on the Jackrabbit can be more accurately calibrated for each individual Jackrabbit in the following manner (calibration of DA0 is assumed in this example, calibration of DA1 would proceed similarly):

- Set desired channel output to **PWM_MIN**.
- Measure voltage V_{min} on DA0.
- Set desired channel output to **PWM_MAX0**.
- Measure voltage V_{max} on DA0.

A linear relation between input value and voltage can now be calculated:

$$b = V_{max} - m \times \text{PWM\_MAX0}$$

$$m = \frac{V_{max} - V_{min}}{\text{PWM\_MAX0} - \text{PWM\_MIN}}$$

$$\text{voltage} = m \times \text{value} + b$$

4.4.1.4 Analog Input

The analog input channel on the Jackrabbit (AD0 on header J5) works by varying analog output channel DA0 until its voltage matches the input voltage on AD0. DA0 obviously cannot be used while an input voltage is being measured, although channel DA0 is still available. The value returned corresponds to the value that DA0 required to match the input voltage (you would call **anaOut(0,value)** for DA0 to provide that same voltage). If the value returned is negative, then the function considers the value suspect for some reason (most likely a failure of the DA0 voltage to settle quickly). The value can be taken as is, or another measurement can be done.

```
void anaIn(int channel, int *value);
```

Analog input for the Jackrabbit analog input channel (AD0).

jrioInit must be called first.

channel is the input channel number (0 only on the Jackrabbit).

An integer between 0 and 1024 will be returned in **value**, corresponding to a voltage obtained if output channel DA0 was set to that value. If a value is found, but the voltage has not appeared to fully settle, the value will be negative (but equal in magnitude to the found voltage) to allow remeasurement if desired.

NOTE: See sample program **JRIOTEST.C** for an example of the use of **anaIn**.

Two versions of the analog input function are available: the standard function, listed above, that does not return until the measurement has been made, and a cofunction version that can be called from within a costatement. This cofunction version allows other tasks to be performed while the voltage match is being made. The voltage measurement will take ten calls of the cofunction version to make a measurement.

```
void cof_anaIn(int channel, int *value);
```

The parameters are identical to those described above for **anaIn**.

NOTE: See sample program **JRIO_COF.C** for an example of the use of **cof_anaIn**.

4.4.2 Serial Communication Drivers

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 Function Reference Manual*** and Technical Note 213, ***Rabbit Serial Port Software***.

4.4.2.1 RS-485 Serial Communication Drivers

The **JR485.LIB** library in the Dynamic C **LIB/JRABLIB** directory contains three RS-485 drivers for use with the Jackrabbit. These drivers are used with the drivers for Serial Port D in the **RS232.LIB** library because **serDopen** uses PC0 (TXD) and PC1 (RXD), which are connected to pin 4 and pin 1 of the SP483EN RS-485 chip at U6. This chip is half duplex, requiring pin 3 (Data Enable) to be high for pins 6 and 7 to act as outputs, and low for those pins to act as inputs.

Parallel Ports D and E on the Rabbit 2000 are double-buffered to provide precisely timed updating of the output pins. Each port is divided into an upper and a lower nibble. All bits of each nibble must be updated simultaneously. Each nibble may be updated constantly at a rate of **perclk**/2 or on a match of a selected timer (Timer A1, B1, or B2).

The bits used to select the update rate for each nibble are left random at power-up. If a mode other than **perclk**/2 is selected, the bits of a particular port will not update on a simple writing to the port's data register. In particular, PD5, the RS-485 transmitter control, will not set the RS-485 transmitter enable unless the upper nibble of Port D is configured properly.

The **JR485Init** function in Dynamic C release 6.16 has provision to disable the special clocking features associated with the high nibble of Port D. This effectively disables digital-to-analog (D/A) converter output channel DA1, the low-resolution D/A converter channel, which also uses PD4. Channel DA0 has its PWM output clocked separately with the low nibble, and so is not affected. Because the analog-to-digital converter uses D/A channel DA0, analog-to-digital conversion is not affected.

There are three RS-485 serial drivers.

```
void Jr485Init();
```

Sets up parallel port D pins for RS-485 use.

```
void Jr485Tx();
```

Sets pin 3 (DE) of the SP483EN chip high to disable Rx and enable Tx.

```
void Jr485Rx();
```

Resets pin 3 (DE) of the SP483EN chip low to disable Tx and enable Rx.

4.5 Upgrading Dynamic C

4.5.1 Patches and Bug Fixes

Dynamic C patches that focus on bug fixes are available from time to time. Check the Web site www.rabbit.com/support/ for the latest patches, workarounds, and bug fixes.

The default installation of a patch or bug fix is to install the file in a directory (folder) different from that of the original Dynamic C installation. Rabbit recommends using a different directory so that you can verify the operation of the patch without overwriting the existing Dynamic C installation. If you have made any changes to the BIOS or to libraries, or if you have programs in the old directory (folder), make these same changes to the BIOS or libraries in the new directory containing the patch. Do ***not*** simply copy over an entire file since you may overwrite a bug fix; of course, you may copy over any programs you have written. Once you are sure the new patch works entirely to your satisfaction, you may retire the existing installation, but keep it available to handle legacy applications.

4.5.2 Add-On Modules

Dynamic C installations are designed for use with the board they are included with, and are included at no charge as part of our low-cost kits. Rabbit offers add-on Dynamic C modules for purchase, including the popular µC/OS-II real-time operating system, as well as PPP, Advanced Encryption Standard (AES), and other select libraries.

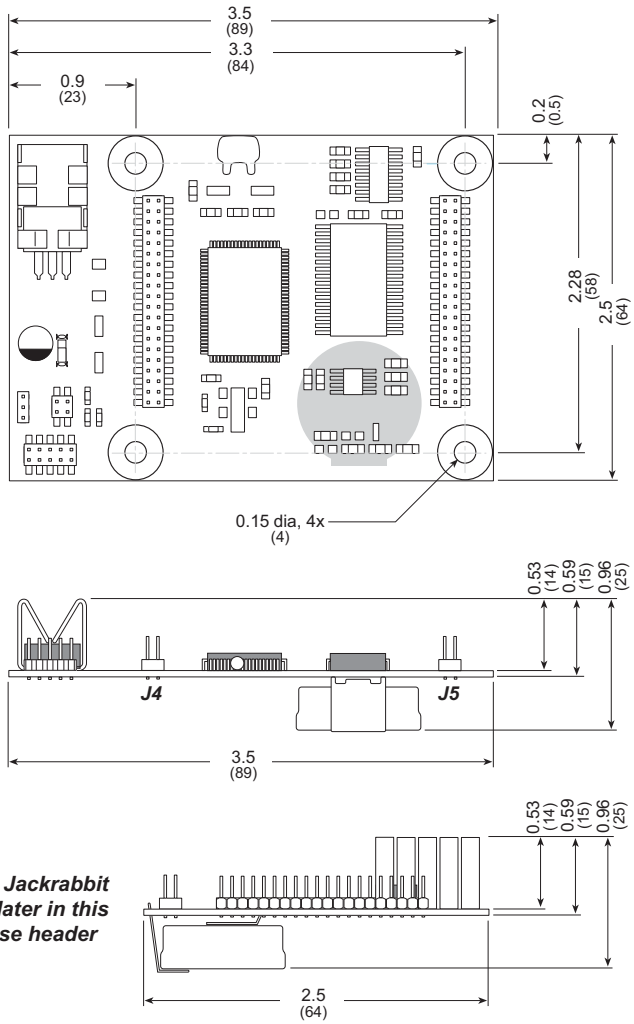
In addition to the Web-based technical support included at no extra charge, a one-year telephone-based technical support module is also available for purchase.

APPENDIX A. SPECIFICATIONS

Appendix A provides the specifications for the Jackrabbit.

A.1 Electrical and Mechanical Specifications

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



Please refer to the Jackrabbit footprint diagram later in this appendix for precise header locations.

Figure A-1. Jackrabbit Dimensions

NOTE: All measurements are in inches followed by millimeters enclosed in parentheses. All dimensions have a manufacturing tolerance of ±0.01" (0.25 mm).

Table A-1 lists the electrical, mechanical, and environmental specifications for the Jackrabbit boards.

Table A-1. Jackrabbit Board Specifications

Parameter	BL1800	BL1810	BL1820
Microprocessor	Rabbit 2000 @ 29.5 MHz	Rabbit 2000 @ 14.74 MHz	
Flash EPROM	256K (supports 128K–512K)	128K (supports 128K–512K)	
SRAM	128K (supports 32K–512K)		
Backup Battery	3 V lithium coin type, 950 mA·h, supports real-time clock and SRAM		None
Digital Inputs	6, CMOS-level		7, CMOS-level
Digital Outputs	4 CMOS-level plus 4 high-power outputs—3 sink up to 1 A and 30 V each, 1 sources up to 500 mA	4 CMOS-level plus 4 high-power outputs—3 sink up to 200 mA and 30 V each, 1 sources up to 100 mA	5 CMOS-level plus 4 high-power outputs—3 sink up to 200 mA and 30 V each, 1 sources up to 100 mA
Configurable I/O	14 CMOS-level: 8 are bytewide, 6 are by bit		15 CMOS-level: 8 are bytewide, 7 are by bit
Analog Inputs	One low-grade A/D input—input range 0.1 V to 2.8 V, 9-bit resolution, 8-bit accuracy, 10 samples/s		
Analog Outputs	Two 9-bit filtered and buffered PWM outputs, one 0.1–2.8 V DC, one 0.7–3.5 V DC, update rate 50 Hz		
Serial Ports	Up to four serial ports: • two RS-232 or one RS-232 (with CTS/RTS) rated at 15 kV ESD • one RS-485 rated at 15 kV ESD (RS-485 driver not installed on BL1820) • one 5 V CMOS-compatible programming port Two serial ports (A and B) can be clocked.		
Serial Rate	Max. burst rate = CLK/32 (async) Max. sustained rate = CLK/64		
Connectors	Two 2 × 20, 2 mm IDC headers		
Real-Time Clock	Yes		
Timers	Five 8-bit timers (four cascadable from the first) and one 10-bit timer with two match registers		
Watchdog/Supervisor	Yes		
Power	8–40 V DC, 1.2 W max., switching regulator	7.5–25 V DC, 100 mA, linear regulator	
Operating Temperature	–40°C to +70°C		
Humidity	5% to 95%, noncondensing		
Board Size	2.50" × 3.50" × 0.76" (64 mm × 89 mm × 19 mm)	2.50" × 3.50" × 0.94" (64 mm × 89 mm × 24 mm)	2.50" × 3.50" × 0.63" (64 mm × 89 mm × 16 mm)

A.1.1 Exclusion Zone

It is recommended that you allow for "exclusion zones" around the Jackrabbit when the Jackrabbit is incorporated into an assembly that includes other components. These "exclusion zones" that you keep free of other components and boards will allow for sufficient air flow, and will help to minimize any electrical or EMI interference between adjacent boards. Figure A-2 shows the "exclusion zones."

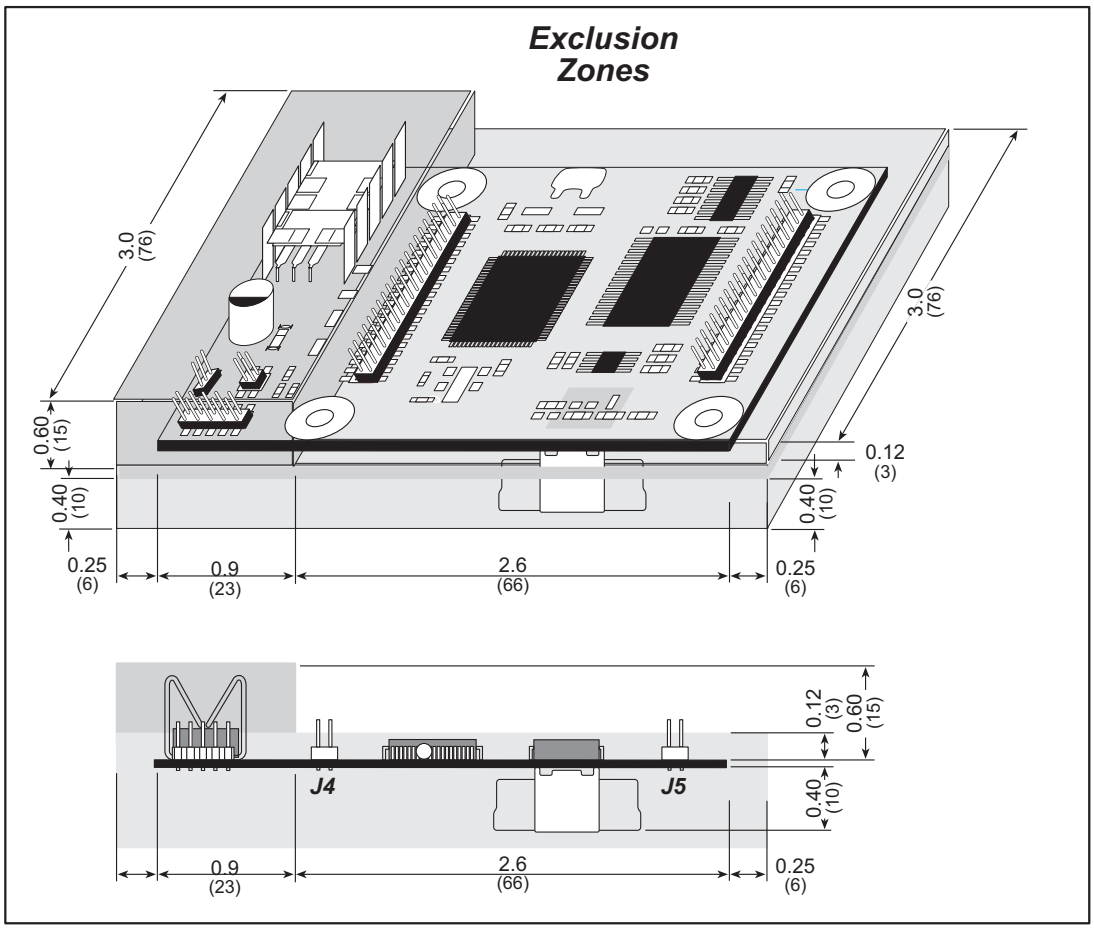


Figure A-2. Jackrabbit "Exclusion Zones"

A.1.2 Headers

The Jackrabbit has 0.1" IDC headers at J1 (1 × 3) and J2 (2 × 2) for the power supply and an external battery connection. There are 2 mm IDC headers at J3 (2 × 5 programming port) and at J4 and J5 (2 × 20 Rabbit subsystems) for physical connection to other boards or ribbon cables.

Figure A-3 shows the Jackrabbit footprint to help you lay out the "motherboard" into which the Jackrabbit is plugged in. These values are relative to the mounting hole closest to header J3.

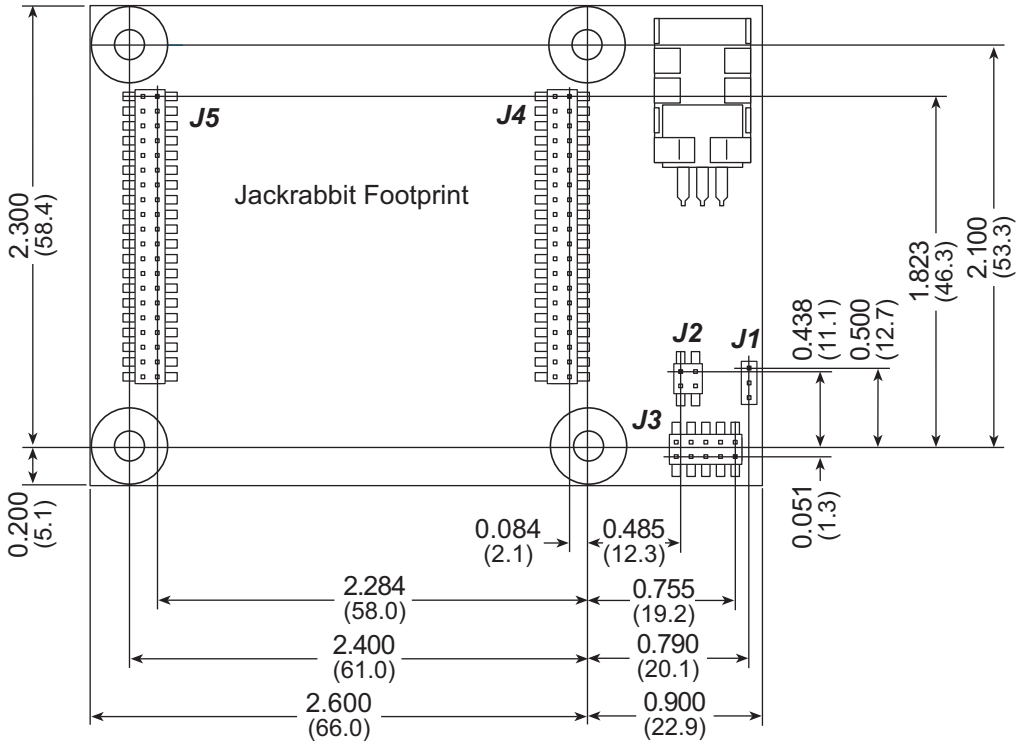


Figure A-3. User Board Footprint for Jackrabbit

A.2 Jumper Configurations

Figure A-4 shows the header and jumper locations used to configure the various Jackrabbit options.

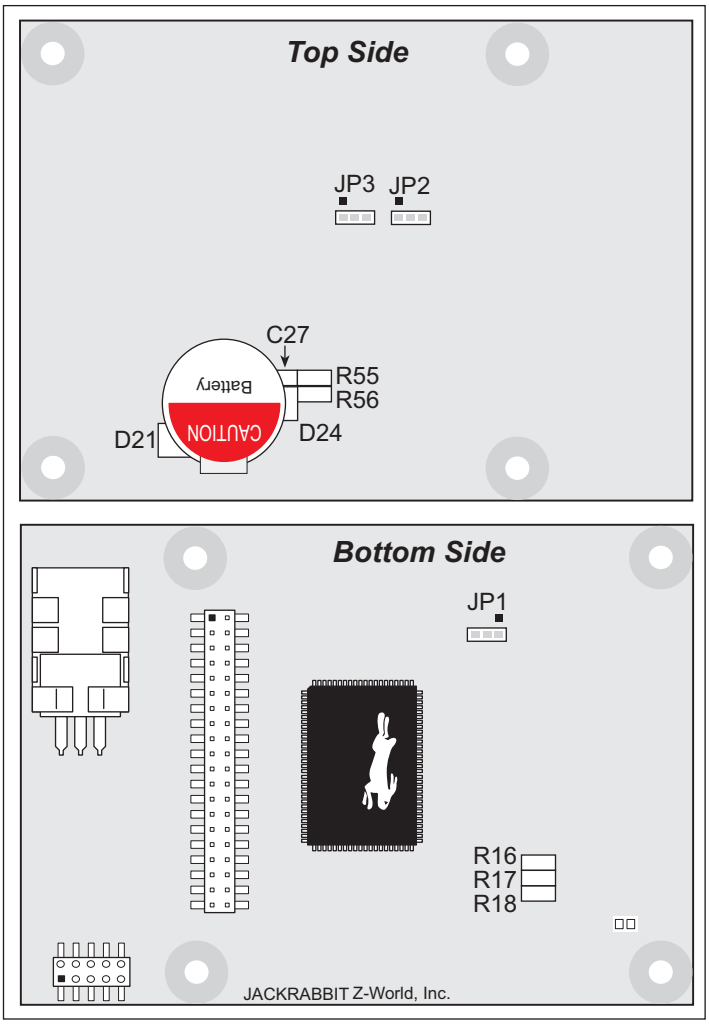


Figure A-4. Location of Jackrabbit Configurable Positions

Table A-2 lists the configuration options. 0 Ω surface mount resistors are used for all the header positions.

Table A-2. Jackrabbit Jumper Configurations

Header	Description	Pins Connected		Factory Default
JP1	SRAM Size	n.c.	32K	
		1–2	128K	✕
		2–3	512K	
JP2	Flash Memory Size	1–2	128K/256K	✕
		2–3	512K	
JP3	Flash Memory Bank Select	1–2	Normal Mode	✕
		2–3	Bank Mode	
—	HV3 Sinking/Sourcing	D21 R55	Sinking	
		R56	Sourcing	✕
—	RS-485 Bias and Termination Resistors (not installed on BL1820)	R17	Termination resistor	✕
		R16 R18	Bias resistors	✕

NOTE: Header JP3 is available only on Jackrabbit boards labeled 175-0255. These boards were introduced in 2003.

A.3 Conformal Coating

The areas around the crystal oscillator and the battery backup circuit on the Jackrabbit have had the Dow Corning silicone-based 1-2620 conformal coating applied. The conformally coated areas are shown in Figure A-5. The conformal coating protects these high-impedance circuits from the effects of moisture and contaminants over time, and helps to maintain the accuracy of the real-time clock.

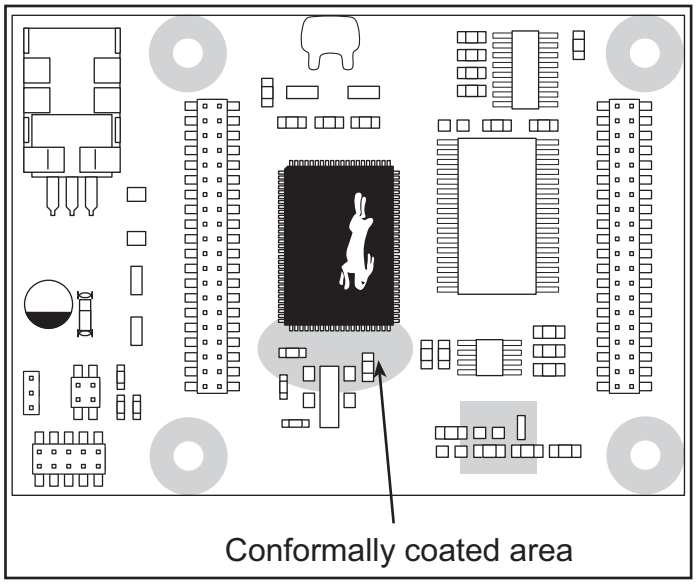


Figure A-5. Jackrabbit Areas Receiving Conformal Coating

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

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

A.4 Use of Rabbit 2000 Parallel Ports

Figure A-6 shows the use of the Rabbit 2000 parallel ports. The Jackrabbit has 27 general-purpose digital inputs/outputs available on headers J4 and J5—15 are bidirectional (one of which is used by the RS-485 chip on BL1800 and BL1810 models), seven are inputs only (one of which is used by the RS-485 chip on BL1800 and BL1810 models), and five are outputs only (one of which is used by the RS-485 chip on BL1800 and BL1810 models), as shown in Figure A-6.

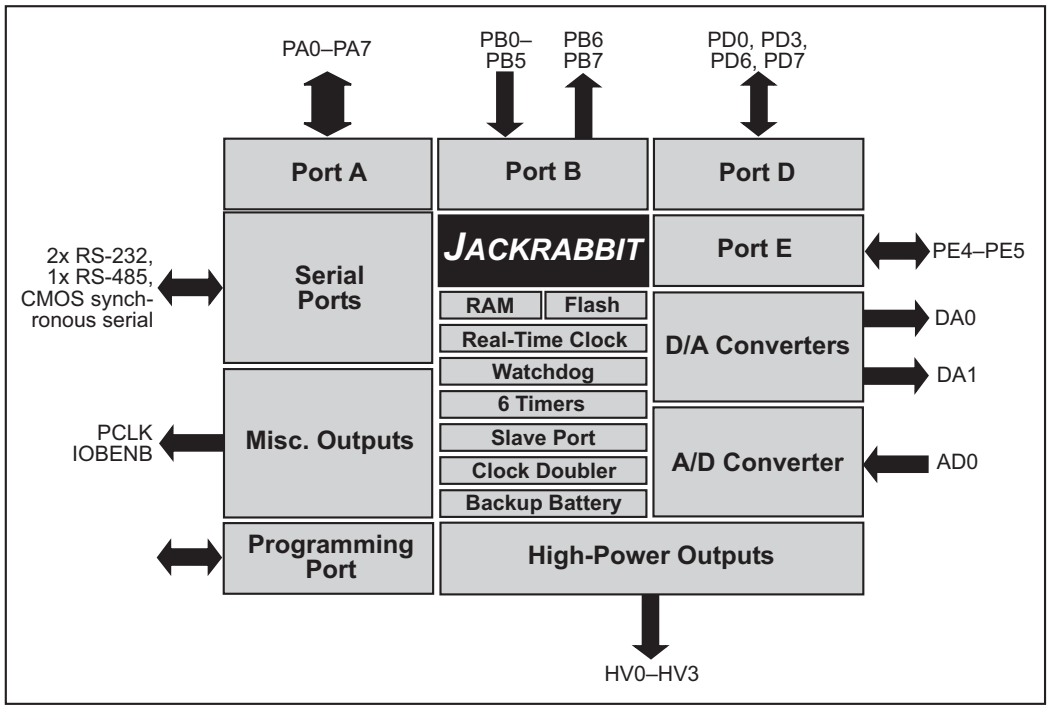


Figure A-6. Jackrabbit Subsystems

The ports on the Rabbit 2000 microprocessor used in the Jackrabbit are configurable, and so the factory defaults can be reconfigured. Table A-3 lists the Rabbit 2000 factory defaults and the alternate configurations.

Table A-3. Jackrabbit Pinout Configurations

Pin	Rabbit 2000 Factory Default	Alternate Use	Jackrabbit Use
PA0–PA7	Parallel I/O	Slave port data bus SD0–SD7	
PB0	Input	Serial port clock CLKB	PB1 (CLKA) is connected to J3 (programming port)
PB1	Input	Serial port clock CLKA	
PB2	Input	Slave port write /SWR	
PB3	Input	Slave port read /SRD	
PB4	Input	Slave port address lines SA1–SA0	
PB5	Input		
PB6	Output		
PB7	Output	Slave port attention line /SLAVEATTN	
PC0	Output	TXD	Connected to RS-485 IC Tx input
PC1	Input	RXD	Connected to RS-485 IC Rx output
PC2	Output	TXC	Connected to RS-232 IC Tx input
PC3	Input	RXC	Connected to RS-232 IC Rx output
PC4	Output	TXB	Connected to RS-232 IC Tx input
PC5	Input	RXB	Connected to RS-232 IC Rx output
PC6	Output	TXA	Connected to programming port
PC7	Input	RXA	

Table A-3. Jackrabbit Pinout Configurations (continued)

Pin	Rabbit 2000 Factory Default	Alternate Use	Jackrabbit Use
PD0	Bitwise or parallel programmable I/O, can be driven or open-drain output		
PD1			Connected to control DA0
PD2			Connected to control DA0
PD3			
PD4		ATXB output	Connected to control DA1
PD5		ARXB input	Connected to RS-485 IC data enable input
PD6		ATXA output	
PD7		ARXA input	
PE0	Bitwise or parallel programmable I/O	I0 output or INT0A input	HV0 output control
PE1		I1 output or INT1A input	HV1 output control
PE2		I2 output	HV2 output control
PE3		I3 output	HV3 output control
PE4		I4 output or external INT0B input	
PE5		I5 output or external INT1B input	
PE6		I6 output	Connected to A/D comparator output
PE7		I7 output or slave port chip select /SCS	Connected to A/D comparator output
PCLK	Peripheral clock	Output	
IOBEN	I/O buffer enable	Output	
WDO	Watchdog output	Single low pulse output	
STAT	Status	Output	Connected to programming port

As shown in Table A-3, pins PE4 and PE5 can instead be used as external INT0B and INT1B interrupts. Pins PD6 and PD7 can instead be used to access Serial Port A on the Rabbit 2000 microprocessor. Pins PB0 and PB1 can instead be used to access the clock on Serial Port B and Serial Port A of the Rabbit 2000 microprocessor.

The four output-only pins are located on PB6–PB7, PCLK, and IOBEN. PB7 can also be used with the slave port of the Rabbit 2000 microprocessor. The primary function of PCLK is as a peripheral clock or a peripheral clock ÷ 2, but PCLK can instead be used as a digital output. Similarly, IOBENB is an I/O buffer enable, but can instead be used as a digital output. STAT and WDO also have limited uses as digital outputs.

APPENDIX B. PROTOTYPING BOARD

Appendix B describes the features and accessories of the Prototyping Board, and explains the use of the Prototyping Board to demonstrate the Jackrabbit and to build prototypes of your own circuits.

B.1 Prototyping Board Overview

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

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

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

The Prototyping Board is shown below in Figure B-1, with its main features identified.

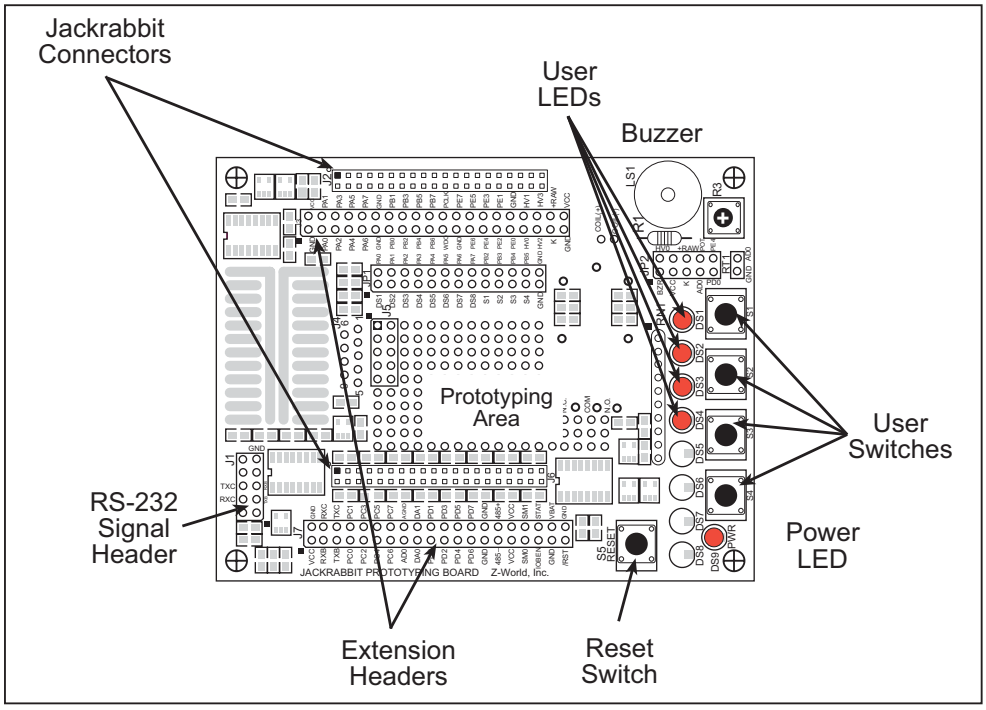


Figure B-1. Prototyping Board

B.1.1 Prototyping Board Features

- **Power LED**—The power LED lights whenever power is connected to the Prototyping Board.
- **Reset Switch**—A momentary-contact, normally open switch is connected directly to the Jackrabbit's **/RESET_IN** pin. Pressing the switch forces a hardware reset of the system.
- **I/O Switches and LEDs**—Four momentary-contact, normally open switches are connected to the PB2–PB5 pins of the Rabbit 2000 microprocessor on the Jackrabbit, and may be read as inputs by sample applications.

  Four LEDs are connected to the PA1–PA4 pins of the of the Rabbit 2000 microprocessor on the Jackrabbit, and may be driven as output indicators by sample applications.
- **Prototyping Area**—A generous prototyping area has been provided for the installation of through-hole components. Several areas for surface-mount devices are also available. (Note that there are SMT device pads on both top and bottom of the Prototyping Board.)
- **Extension Headers**—The complete pin sets of the Jackrabbit are duplicated at these two sets of headers. Developers can solder wires directly into the appropriate holes, or, for more flexible development, 40-pin header strips can be soldered into place. See Figure B-7 for the header pinouts.
- **RS-232**—Two 3-wire or one 5-wire RS-232 serial port are available on the Prototyping Board. Refer to the Prototyping Board schematic (090-0088) for additional details.

  A 10-pin 0.1-inch spacing header strip is installed at J1 to permit connection of a ribbon cable leading to a standard DE-9 serial connector.

B.2 Mechanical Dimensions and Layout

Figure B-2 shows the mechanical dimensions and layout for the Jackrabbit Prototyping Board.

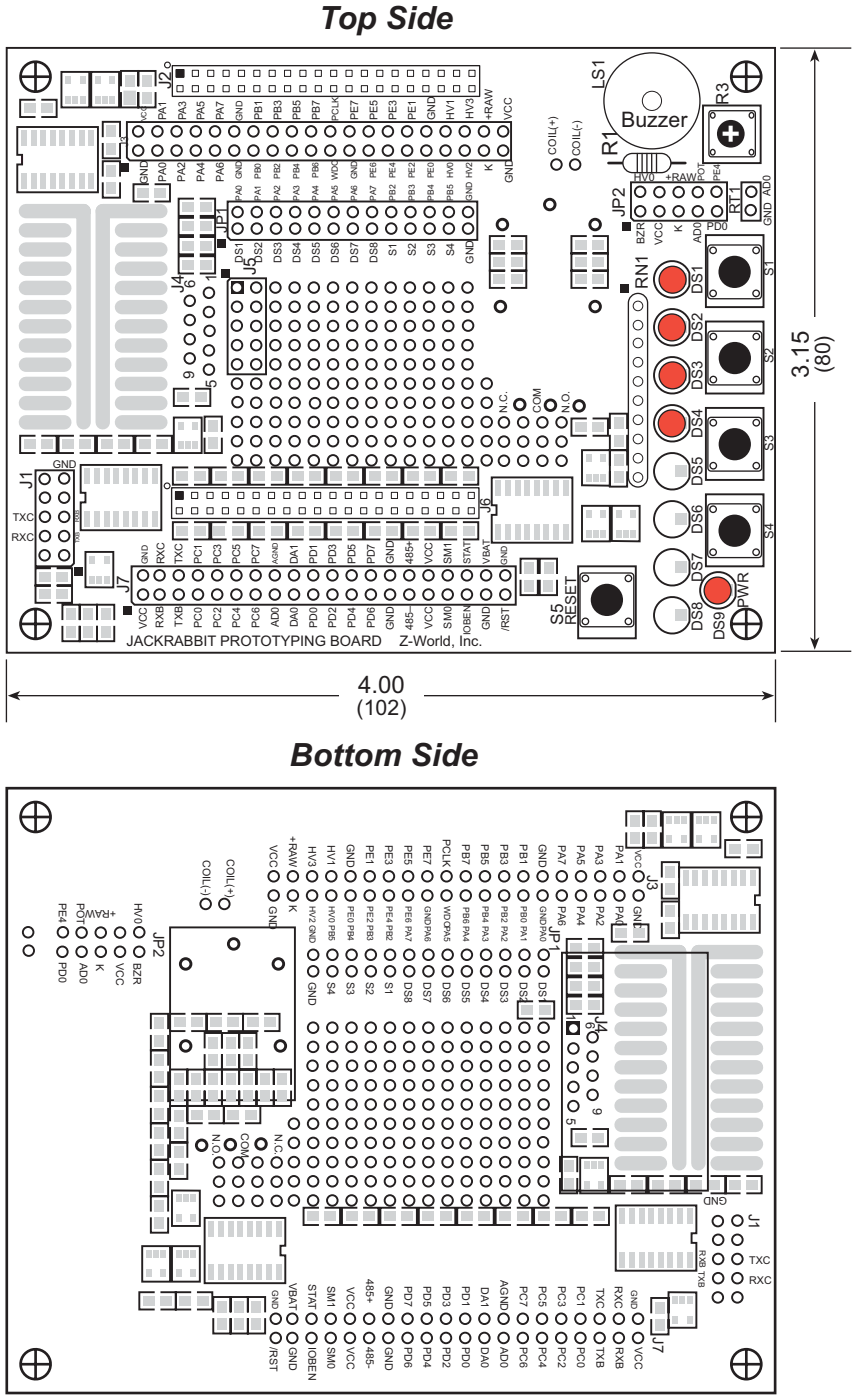


Figure B-2. Jackrabbit Prototyping Board

B.3 Using the Prototyping Board

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

Once you have looked at the basic sample programs described in the ***Jackrabbit (BL1800) Getting Started Manual***, solder the headers included in the bag of spare parts onto the Prototyping Board. Solder a 10-pin header to the top side at location J1. Solder the additional headers shown in Figure B-3 onto the bottom side.

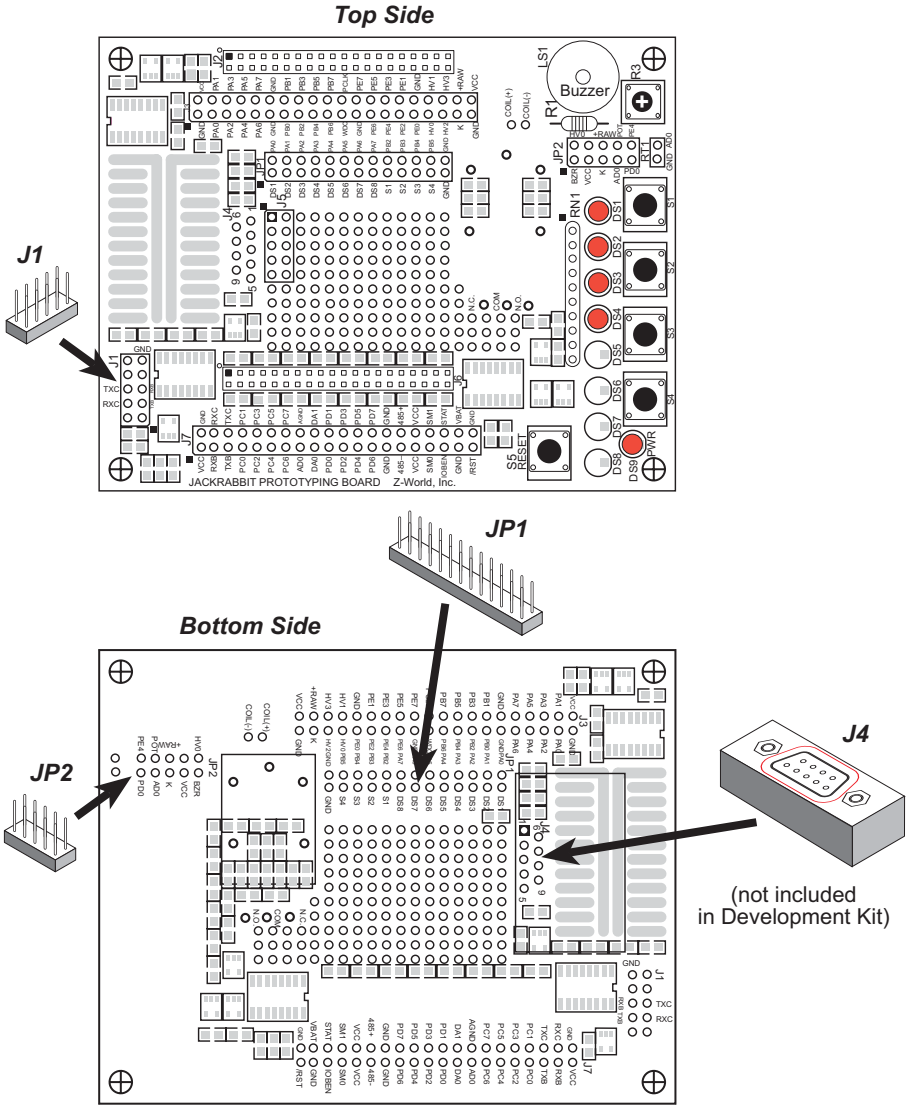


Figure B-3. Where to Solder Additional Headers

B.3.1 Demonstration Board

A relay, a thermistor, four additional LEDs, and a serial cable are included in a bag of parts to further allow exploration of the Jackrabbit‘s operation.

The SPDT relay handles 120 V at 5 A with a 12 V activating coil. The layout to accept this relay is included on the top side of the Prototyping Board. Note that the relay coil connections need to be wired to one of the sinking high-power outputs (HV0–HV2) of the Jackrabbit. Install the relay on the bottom side of the Prototyping Board, as shown in Figure B-4.

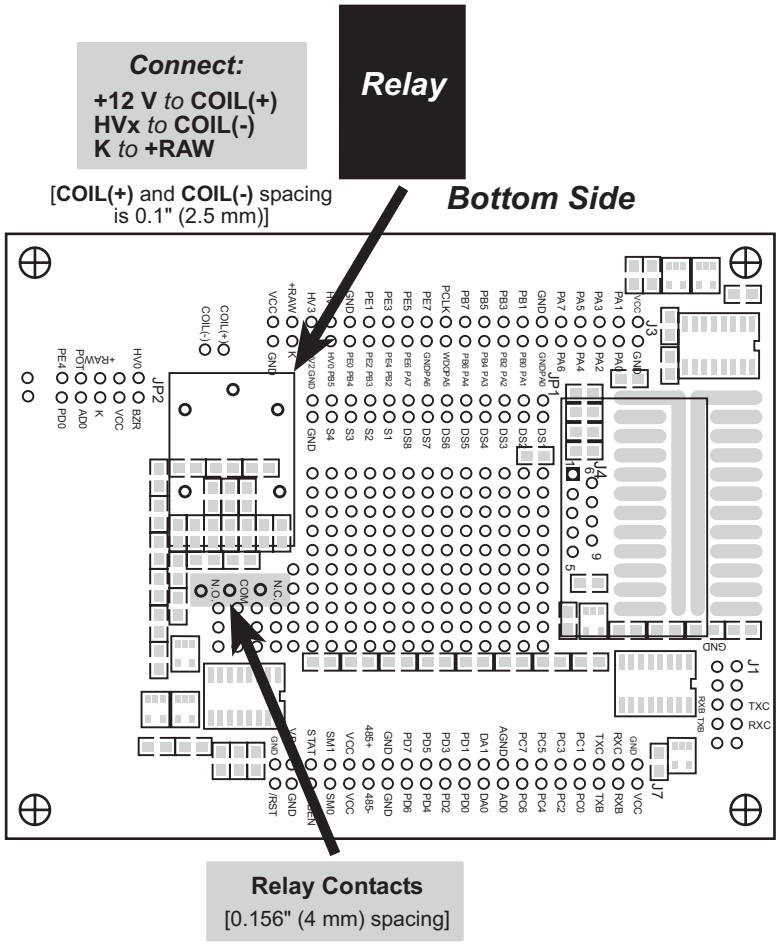


Figure B-4. Installation of Relay on Prototyping Board

NOTE: COIL(+) must be connected to a 12 V power supply when using the relay. This is the nominal voltage supplied as +RAW when the transformer supplied with the Development Kit is used, and in this case you may connect **COIL(+)** to +RAW. If you use another power supply, connect **COIL(+)** to +12 V if +RAW is different.

NOTE: If you do use the transformer supplied with the Development Kit for **COIL(+)**, be aware that its voltage may be as high as 16 V at low current draws. This needs to be taken into consideration if you plan to use a 12 V relay in critical applications.

The thermistor has a nominal room-temperature resistance of about 10 kΩ, which drops to about 6 kΩ at 40°C. Once you solder the thermistor onto the RT1 pads (see Figure B-5) on the Prototyping Board, the A/D converter readings on AD0 will change with temperature.

If the 10 kΩ potentiometer is removed, the change in A/D converter readings with temperature will be larger.

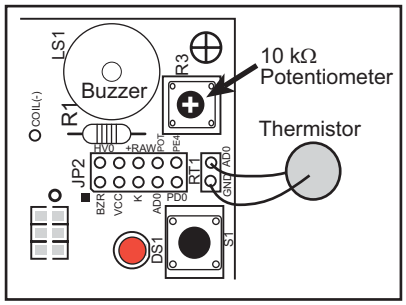


Figure B-5. Thermistor and Potentiometer Locations

The LEDs can be mounted in positions DS5–DS8, shown in Figure B-6, to display the complete status for Parallel Port A.

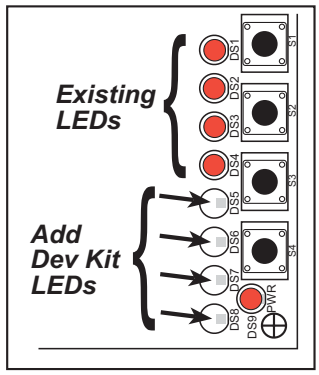


Figure B-6. LED Location

The serial cable included in the parts bag can be used to connect the Jackrabbit's RS-232 outputs from header J1 on the Prototyping Board to an available PC serial port.

Unlike the CMOS-level signals on header J3, the programming port on the Jackrabbit board, the signals on header J1 on the Prototyping Board are full RS-232 level signals without needing the CMOS to RS-232 converter that is present in the programming cable. The RS-232 level signals are processed via the MAX232 transceiver chip, U4, on the Jackrabbit board to Serial Ports B and C of the Rabbit 2000. The CMOS-level signals on the programming port are connected to Serial Port A.

B.3.2 Prototyping Board

To maximize the availability of Jackrabbit resources, the demonstration hardware (LEDs, switches, potentiometer, buzzer) on the Prototyping Board may be disconnected. This is done by cutting the traces seen between and within the silk-screen outline of headers JP1 and JP2 on the Prototyping Board. Figure B-6 shows the 16 places where cuts should be made. An exacto knife or high-speed precision grinder tool like a Dremel® tool would work nicely to cut the traces. Alternatively, if safety is a major concern, a small standard screwdriver may be carefully and forcefully used to wipe through the PCB traces.

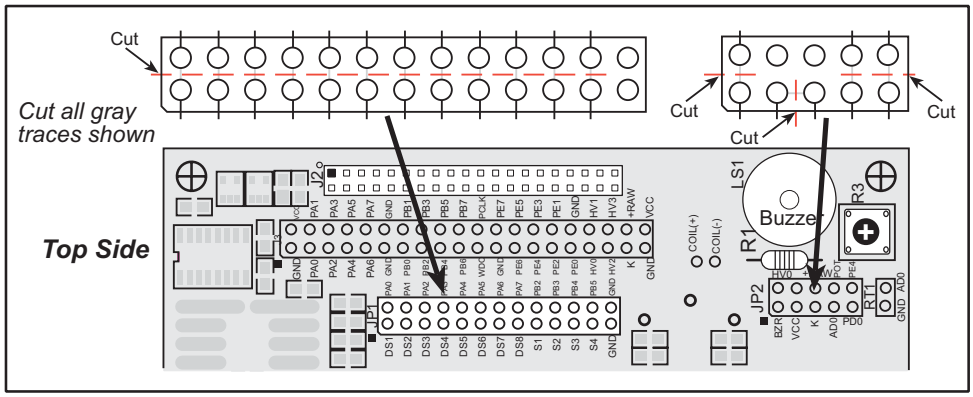


Figure B-6. Where to Cut Traces to Permanently Disable Demonstration Hardware on Prototyping Board

Jumpers across the appropriate pins on headers JP1 and JP2 can be used to reconnect specific demonstration hardware later if needed. Each pin is labeled on the PCB to facilitate placing the jumpers. The jumper positions are summarized in Table B-1.

Table B-1. Prototyping Board Jumper Settings

Header JP1		Header JP2 (continued)	
Pins	**Description**	**Pins**	**Description**
1–2	Buzzer	5–6	PA2 to LED DS3
3–5	K to +5 V	7–8	PA3 to LED DS4
5–6	K to +RAW	9–10	PA4 to LED DS5
7–8	Potentiometer or Thermistor	11–12	PA5 to LED DS6
9–10	Interrupt Enable	13–14	PA6 to LED DS7
Header JP2		15–16	PA7 to LED DS8
Pins	**Description**	17–18	PB2 to switch S1
1–2	PA0 to LED DS1	19–20	PB3 to switch S2
3–4	PA1 to LED DS2	21–22	PB4 to switch S3

Once the LEDs, resistors, and switches are disconnected as described above, the user has a Jackrabbit board with connection points conveniently brought out to labeled points at headers J3 and J7 on the Prototyping Board. Small to medium circuits can be prototyped using point-to-point wiring with 20 to 30 AWG wire between the prototyping area and the holes at locations J3 and J7. Note that the pinouts at locations J3 and J7 on the top side of the Prototyping Board (shown in Figure B-7) are a mirror image of the Jackrabbit board pinouts.

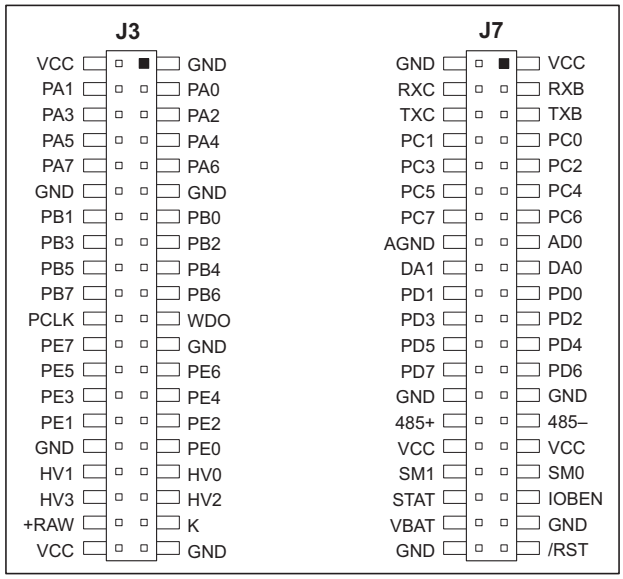


Figure B-7. Jackrabbit I/O Pinout on Prototyping Board (top side)

A user-supplied DE9 connector can be added as shown in Figure B-3. The signals are brought out to location J5 on the top side of the Prototyping Board.

There are six independent surface-mount 14- to 16-pin SOIC pads and fourteen 3- to 5-pin SOT23 pads. Each component has every one of its pin pads connected to a hole in the prototyping area. The layout is such that there is another SOIC or SOT23 pad directly on the other side of the PCB from the SOIC or SOT23 pads. However, each layout location is routed to its unique set of connection holes. Because the traces are very thin, carefully determine which set of holes is connected to which surface-mount pad. There are several standard 0805 passive-component surface-mount pads. These pads are not routed to wiring holes so wire must be soldered directly to the component. In addition, there is a large generic array of wide traces connected to large holes. This is provided as an additional area for surface-mount passive components. There is a moderate amount of 0.1” arrayed through-hole prototyping area (about 137 holes) for mounting through-hole components.

Thus, many circuits requiring special circuitry external to the Jackrabbit can be prototyped and tested with the Prototyping Board. If additional prototyping space is needed, install 40-pin headers at locations J3 and J7 on the top side of the Prototyping Board to connect to sockets that you would install at J3 and J7 on the top side of a second Prototyping Board.

APPENDIX C. POWER MANAGEMENT

C.1 Power Supplies

Power is supplied to the Jackrabbit board from an external source through either header J1 or header J4. J1 is a 3-pin straight header with a pitch of 0.1". V_{in} is on pin 2 between ground on pins 1 and 3. The symmetry allows for a 3-pin cable to be connected either way.

The Jackrabbit board itself is protected against reverse polarity by a Shottky diode at D2 as shown in Figure C-1. The Shottky diode has a low forward voltage drop, 0.3 V, which keeps the minimum DCIN required to power the Jackrabbit lower than a normal silicon diode would allow.

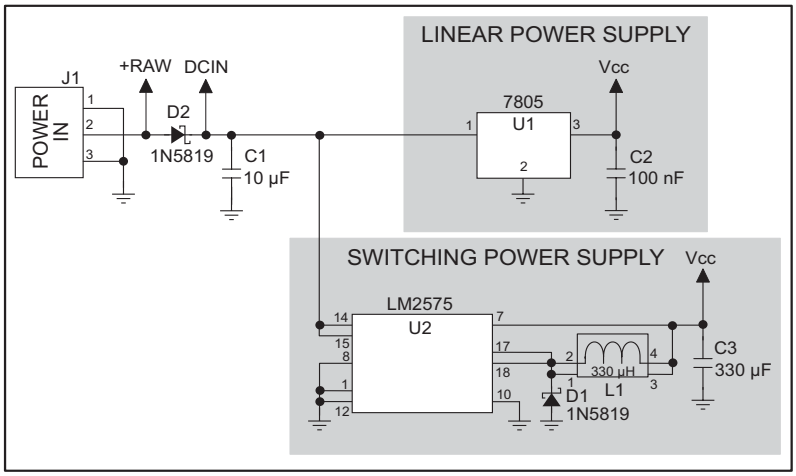


Figure C-1. Jackrabbit Power Supply Schematic

The external power, +RAW, is provided to any daughterboard connected to the Jackrabbit via pin 38 of header J4. +RAW is ***not*** protected against reversed polarity, such as could happen if the cable was connected to header J1 offset by one pin. This absence of protection is intentional so as to provide the maximum possible voltage to any daughterboard connected to the Jackrabbit.

Capacitor C1 provides surge current protection for the voltage regulator, and allows the external power supply to be located some distance away from the Jackrabbit board. A switching power supply or a linear power supply option is provided, depending on the Jackrabbit model.

The linear voltage regulator is simply a fixed-voltage regulator with a ±5% voltage output tolerance as the temperature changes. The regulator has a small heat sink, which increases the maximum external input voltage. Higher external input voltages increase the voltage dropped by the regulator. The Vcc coming out of the regulator is always 5 V.

The power necessarily dissipated by the regulator can be calculated if both the external input voltage and the current drawn by the Jackrabbit board and daughterboards connected to the Jackrabbit board are known. The current provided by the high-power output drivers does not have to be included if a separate power supply is connected to K so that power does not come from Vcc.

The linear regulator maintains its output voltage to within ±5% as long as the heat sink is dissipating less than 2 W. The regulator will operate outside its specifications when the heat sink is dissipating 2 W to 3.3 W. Thermal shutdown turns the regulator off above 3.3 W. Figure C-2 shows the power operating curves.

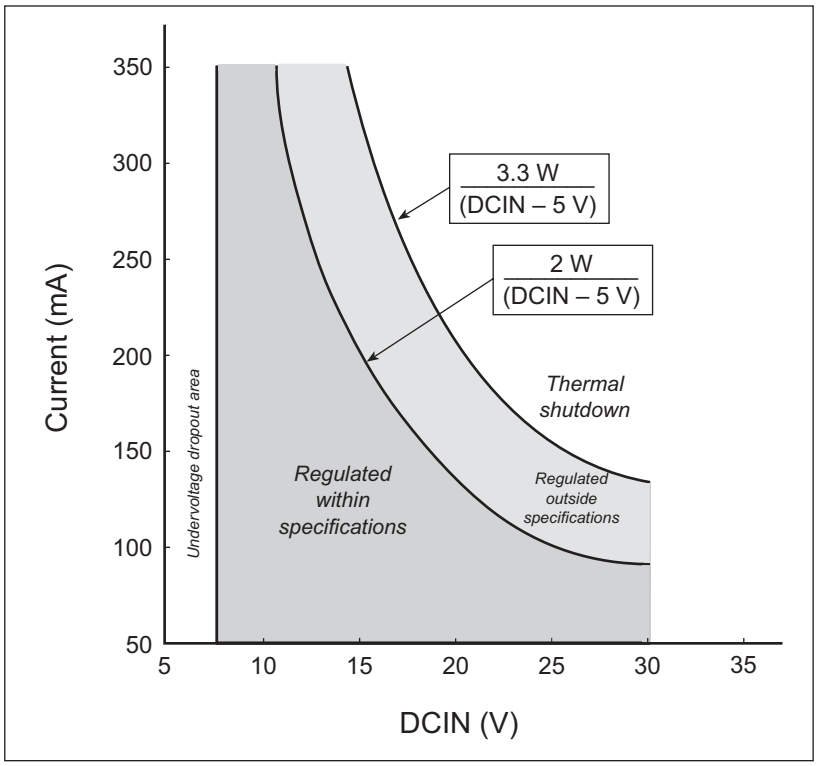


Figure C-2. 7805 Linear Regulator Power Operating Curve

The Jackrabbit operating at 14.74 MHz with no loading at the outputs typically consumes 105 mA when the programming cable is connected, and 95 mA when the programming cable is not connected. This means that DCIN can safely be from 7.5 V to 25 V. An additional 50 mA is available for a daughterboard, but the voltage regulation would suffer slightly.

The switching voltage regulator is used when there is a need for an additional range in the external input voltage or when lower power consumption is desired. The input voltage range is from 8 V to 40 V.

Figure C-3 shows typical power operating curves for both the linear regulator (BL1810 and BL1820) and the switching regulator (BL1800) for a nonloaded Jackrabbit operating at 14.74 MHz with the programming cable connected.

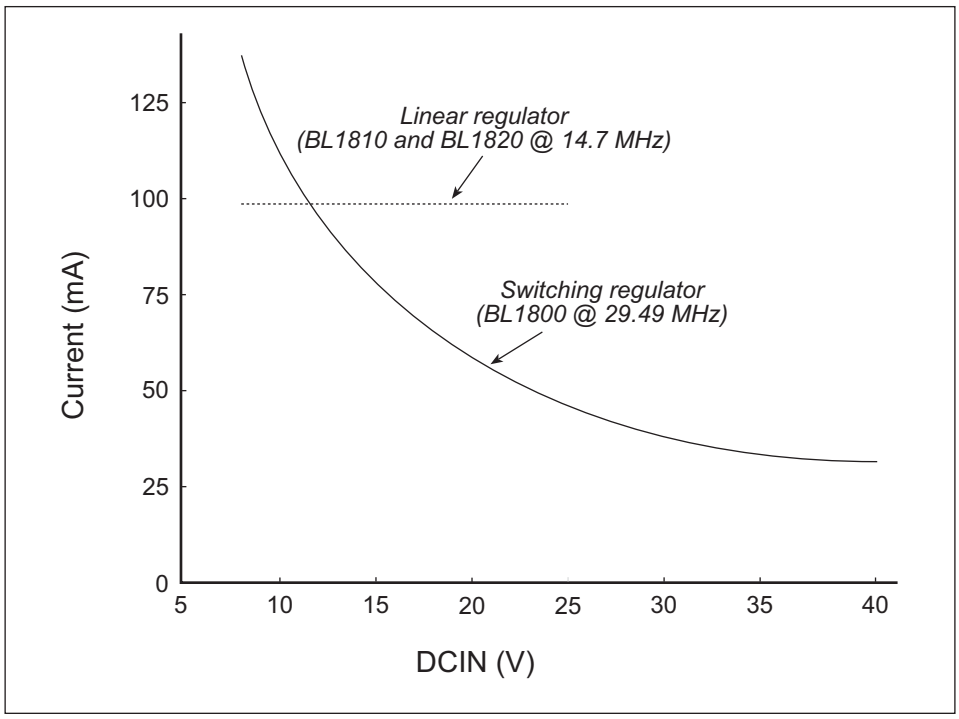


Figure C-3. Typical Jackrabbit Current Consumption

C.2 Batteries and External Battery Connections

The soldered-in 950 mA·h lithium coin cell provides power to the real-time clock and SRAM when external power is removed from the circuit. This allows the Jackrabbit to continue to keep track of time and preserves the SRAM memory contents.

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

$$\frac{950\ \text{mA}\cdot\text{h}}{20\ \mu\text{A}} = 5.4\ \text{years}.$$

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

$$\frac{950\ \text{mA}\cdot\text{h}}{4\ \mu\text{A}} = 27\ \text{years (shelf life = 10 years)}.$$

Since the shelf life of the battery is 10 years, the battery can last for its full shelf life when external power is applied to the Jackrabbit.

Header J2, shown in Figure C-4, allows external access to the battery. This header makes it possible to connect an external 3 V power supply while replacing the soldered-in 3 V lithium coin-type battery. This allows the Jackrabbit SRAM and real-time clock to retain data while the battery is being replaced.

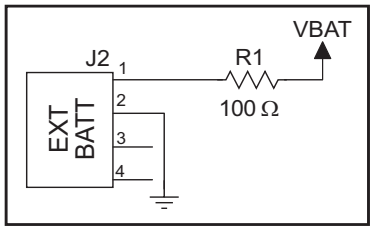


Figure C-4. External Battery Connections at Header J2

Alternatively, header J2 can be used to accommodate an external battery. In this case, be sure to cut out the soldered-in battery on the Jackrabbit to prevent discharging the external battery into a dead battery.

C.2.1 Battery Backup Circuit

Figure C-5 shows the Jackrabbit battery backup circuitry.

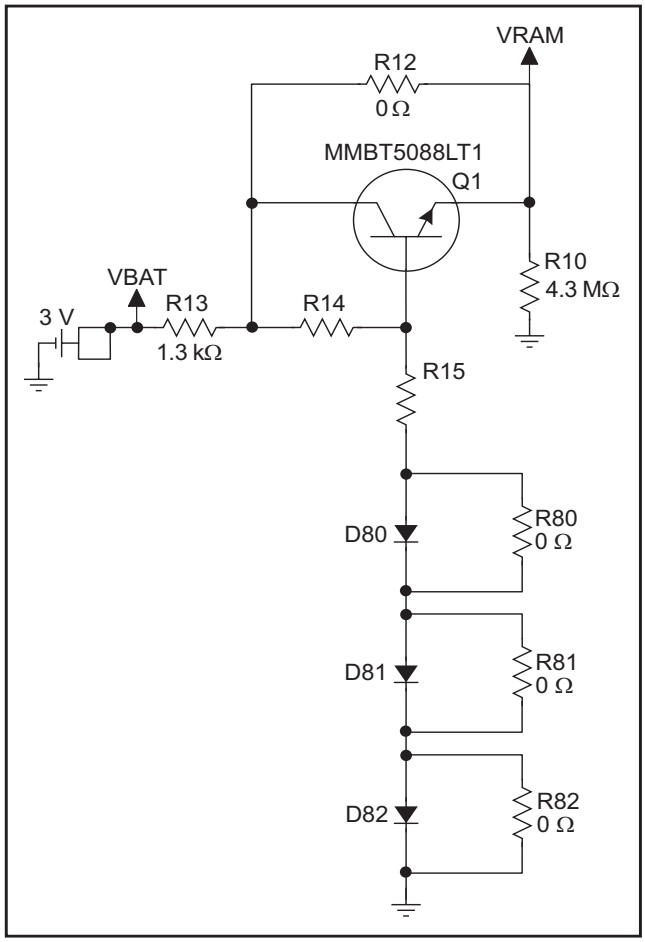


Figure C-5. Jackrabbit Battery Backup Circuit

Resistor R12, shown in Figure C-5, is typically not stuffed on the Jackrabbit board. VRAM and Vcc are equal when power is supplied to the Jackrabbit. R13 prevents any catastrophic failure of Q1 from allowing unlimited current to enter the soldered-in battery.

Resistors R14 and R15 make up a voltage divider between the battery voltage and the temperature-compensation voltage at the anode of diode D80. This voltage divider biases the base of Q1 to about 2.6 V. V_{BE} on Q1 is about 0.55 V. Therefore, VRAM is about 2.05 V.

These voltages vary with temperature. VRAM varies the least because temperature-compensation diodes D80–D82 will offset the variation with temperature of Q1 V_{BE}. R80–R82 may be stuffed instead of the corresponding D80–D82 to provide the optimum temperature compensation.

Resistor R10 provides a minimum load to the regulator circuit.

The battery-backup circuit serves two purposes:

- It reduces the battery voltage to the real-time clock, thereby reducing the current consumed by the real-time clock and lengthening the battery life.
- It ensures that current can flow only ***out*** of the battery to prevent charging the battery.

C.2.2 Power to VRAM Switch

The VRAM switch, shown in Figure C-6, allows the soldered-in battery 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.

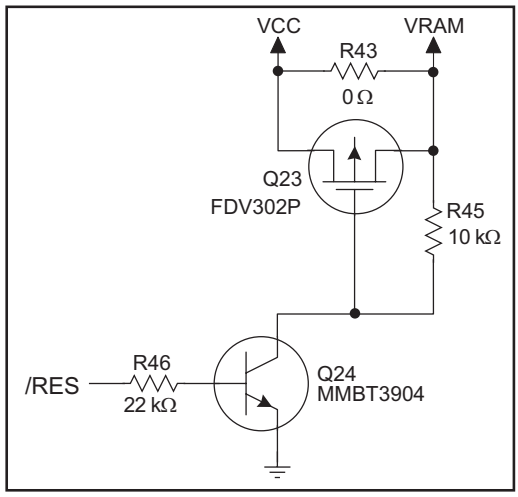


Figure C-6. VRAM Switch

Transistor Q23 is needed to provide a very small voltage drop between Vcc and VRAM (<100 mV, typically 10 mV) so that the processor lines powered by Vcc will not have a significantly different voltage than VRAM.

When the Jackrabbit is not resetting (pin 2 on U21 is high), the /RES line will be high. This turns on Q24, causing its collector (pin 3) to go low. This turns on Q23, allowing VRAM to nearly equal Vcc.

When the Jackrabbit is resetting, the /RES line will go low. This turns off Q23 and Q24, providing an isolation between Vcc and VRAM.

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

C.2.3 Reset Generator

The Jackrabbit uses a reset generator, U21, to reset the Rabbit 2000 microprocessor when the voltage drops below the voltage necessary for reliable operation. The Jackrabbit does not have a reset output presented to the headers. The reset generator has a reset input that can be used to force the Jackrabbit to reset. This input is available on headers J3 and J5, and also on pads directly below header J5. The two pads allow a screwdriver to be used to short the pads, forcing a reset.

C.3 Chip Select Circuit

Figure C-7 shows a schematic of the chip select circuit.

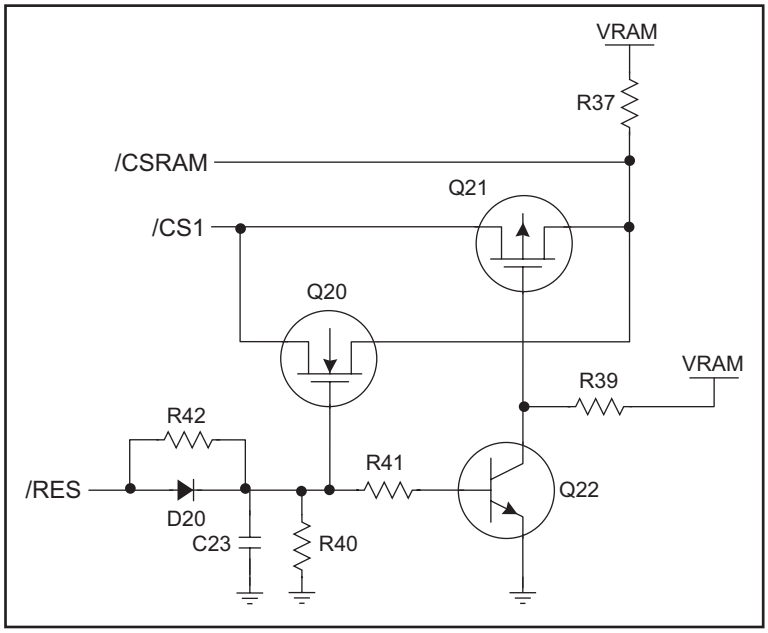


Figure C-7. Chip Select Circuit

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

In a powered-up condition, the CS control circuit must allow the processor's chip select signal /CS1 to control the SRAM's CS signal /CSRAM. So, with power applied, /CSRAM must be the same signal as /CS1, and with power removed, /CSRAM must be held high (but only needs to be battery voltage high). Q20 and Q21 are MOSFET transistors with opposing polarity. They are both turned on when power is applied to the circuit. They allow the CS signal to pass from the processor to the SRAM so that the processor can periodically access the SRAM. When power is removed from the circuit, the transistors will turn off and isolate /CSRAM from the processor. The isolated /CSRAM line has a 100 kΩ pullup resistor to VRAM (R37). 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 Q20 and Q21 are of opposite polarity so that a rail-to-rail voltages can be passed. When the /CS1 voltage is low, Q20 will conduct. When the /CS1 voltage is high, Q21 will conduct. It takes time for the transistors to turn on, creating a propagation delay. This delay is typically very small, about 10 ns to 15ns.

The signal that turns the transistors on is a high on the processor's reset line, /RES. When the Jackrabbit is not in reset, the reset line will be high, turning on N-channel Q20 and Q22. Q22 is a simple inverter needed to turn on Q21, an P-channel MOSFET. When a reset occurs, the /RES line will go low. This will cause C23 to discharge through R42 and R40. This small delay (about 160 µs) ensures that there is adequate time for the processor to write any last byte pending to the SRAM before the processor puts itself into a reset state. When coming out of reset, CS will be enabled very quickly because D20 conducts to charge capacitor C23.

INDEX

R

S

T

U

SCHEMATICS

090-0092 Jackrabbit Schematic

www.rabbit.com/documentation/schemat/090-0092.pdf

090-0088 Jackrabbit Prototyping Board Schematic

www.rabbit.com/documentation/schemat/090-0088.pdf

090-0128 Programming Cable Schematic

www.rabbit.com/documentation/schemat/090-0128.pdf

You may use the URL information provided above to access the latest schematics directly.

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

DIGI:

101-0558 101-0814 20-101-0356 20-101-0357 20-101-0358