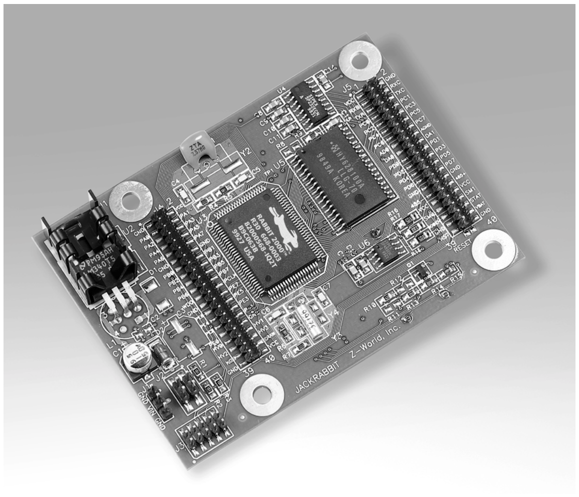
Z·WORLD



Jackrabbit (BL1800)

C-Programmable Controller

User's Manual

010118 - D

Jackrabbit (BL1800) User’s Manual

Part Number 019-0067 • 010118 - D • Printed in U.S.A.

Copyright

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

Trademarks

- Dynamic C® is a registered trademark of Z-World, Inc.
- Windows® is a registered trademark of Microsoft Corporation
- Jackrabbit™ is a trademark of Z-World, Inc.

Notice to Users

When a system failure may cause serious consequences, protecting life and property against such consequences with a backup system or safety device is essential. The buyer agrees that protection against consequences resulting from system failure is the buyer’s responsibility.

This device is not approved for life-support or medical systems.

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

Company Address

Z-World, Inc.
2900 Spafford Street
Davis, California 95616-6800
USA

Telephone: (530) 757-3737
Facsimile: (530) 757-5141
Web site: http://www.zworld.com
E-mail: zworld@zworld.com

TABLE OF CONTENTS

About This Manual

This manual provides instructions for installing, testing, configuring, and interconnecting the Jackrabbit controller and the Jackrabbit Prototyping Board.

Assumptions

Assumptions are made regarding the user's knowledge and experience in the following areas:

- Ability to design and engineer the target system that a Jackrabbit will control.
- Understanding of the basics of operating a software program and editing files under Windows on a PC.
- Knowledge of basic assembly language and architecture for controllers.

  For a full treatment of C, refer to the following texts:

  The C Programming Language by Kernighan and Ritchie (published by Prentice-Hall).

  and/or

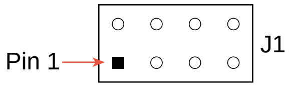
  C: A Reference Manual by Harbison and Steel (published by Prentice-Hall).

- Knowledge of basic assembly language and Rabbit microprocessor architecture.

  For more information on the Rabbit 2000 microprocessor, refer to the ***Rabbit 2000 Microprocessor User's Guide***.

Pin Number 1

A black square indicates pin 1 of all headers.



Measurements

All diagram and graphic measurements are in inches followed by millimeters enclosed in parenthesis.

1. INTRODUCTION

The Jackrabbit is a high-performance, C-programmable controller with a compact form factor. A Rabbit 2000 microprocessor operating at 30 MHz provides fast data processing.

1.1 Features

- 30 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

Appendix A provides detailed specifications for the Jackrabbit.

Three versions of the Jackrabbit are available. Their standard features are summarized in Table 1.

Table 1. Jackrabbit Series Features

Model	Features
BL1800	Full-featured controller with switching voltage regulator.
BL1810	BL1800 with 14.7 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.

1.2 Development and Evaluation Tools

1.2.1 Development Kit

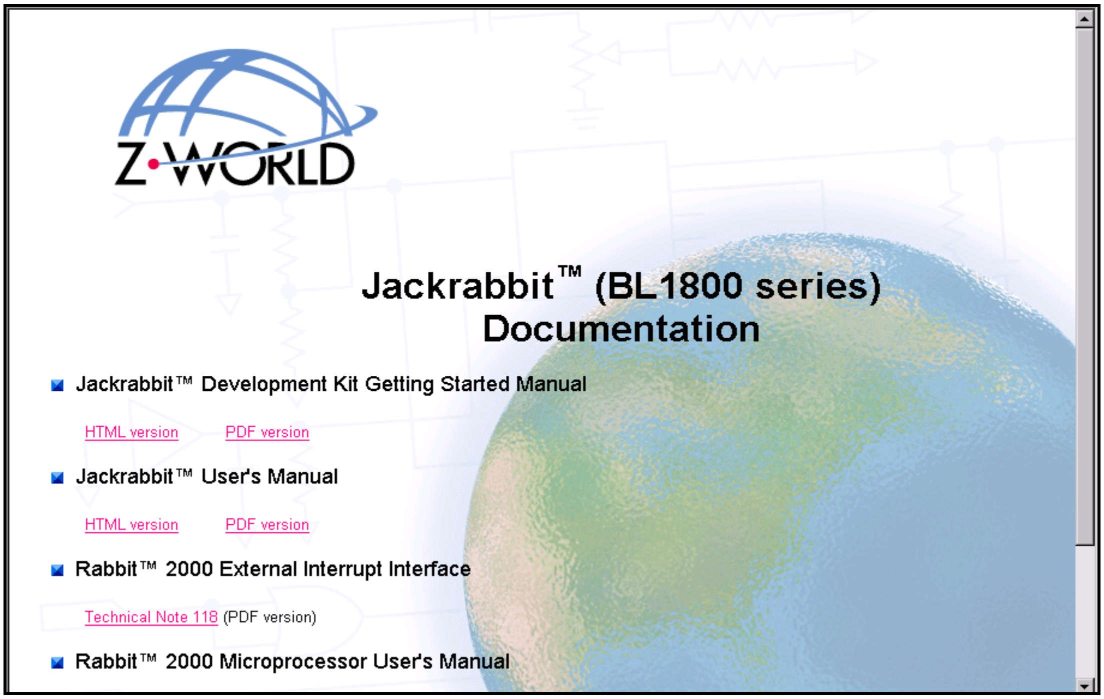
The Development Kit has the essentials that you need to design your own a microprocessor-based system, and includes a complete software development system (Dynamic C).

The items in the Development Kit and their use are as follows:

- CD-ROM with Dynamic C software, Jackrabbit controller and Rabbit™ 2000 microprocessor documentation. You may install this software by inserting the disk into your CD-ROM drive. If it doesn't start automatically, click on "setup.exe." This software runs under Windows '95, Windows '98, and Windows NT. We suggest taking the option to load the documentation to your hard disk. The documentation is in both HTML and Adobe PDF format, and may be viewed with a browser.
- Jackrabbit BL1810 controller board. This is a complete controller board that includes a Rabbit 2000 processor, 128K of flash memory, and 128K of RAM.
- Prototyping Board. The Jackrabbit board can be plugged into this board. The Prototyping Board includes various accessories such as pushbutton switches, LEDs, and a beeper. In addition, you can add your own circuitry in the prototyping space provided.
- Programming cable. The programming cable is used to connect your PC serial port to the Jackrabbit board to write and debug C programs that run on the Jackrabbit board.
- Loose parts kit. This bag of parts contains parts that you can solder to the Prototyping Board for various demonstrations.
- AC adapter. The AC adapter is used to power the Jackrabbit board. The wall transformer is supplied only for Development Kits sold for the North American market. The Jackrabbit can also be powered from any DC voltage source between 7.5 V and 25 V. The regulator becomes rather hot for voltages above 15 V.

1.2.2 Documentation

- Our documentation is provided in paperless form on the CD-ROM included in the Development Kit. (A paper copy of the "Getting Started" instructions and a "Dynamic C Tutorial" *is* included.) Most documents, including this comprehensive ***Jackrabbit User's Manual***, are provided in two formats: HTML and PDF. HTML documents can be viewed with an Internet browser, either *Netscape Navigator* or *Internet Explorer*. HTML documents are very convenient because all the documents are hyperlinked together, and it is easy to navigate from one place to another. PDF documents can be viewed using the Adobe Acrobat reader, which is automatically invoked from the browser. The PDF format is best suited for documents requiring high resolution, such as schematics, or if you want to print the document. Don't print a hard copy from the HTML version because the HTML version has no page numbers and the cross-references and table of contents links only work if viewed on line. The PDF versions contain page number references to allow navigation when reading a paper version of the manual. To view the online documentation with a browser, open the file **`default.htm`** in the **`docs`** folder. When you open the **`default.htm`** file with your browser, you will see a page similar to that shown below.



Jackrabbit™ (BL1800 series) Documentation

- Jackrabbit™ Development Kit Getting Started Manual

  HTML version PDF version

- Jackrabbit™ User's Manual

  HTML version PDF version

- Rabbit™ 2000 External Interrupt Interface

  Technical Note 118 (PDF version)

- Rabbit™ 2000 Microprocessor User's Manual

1.2.3 Software

The Jackrabbit is programmed using Z-World's Dynamic C, an integrated development environment that includes an editor, a C compiler, and a debugger. Library functions provide an easy-to-use interface for the Jackrabbit board.

The Prototyping Board includes pushbutton switches, LEDs, and a beeper, and is plugged into the Jackrabbit board. By writing programs that run on the Jackrabbit board, you can flash the LEDs, beep the beeper, and otherwise demonstrate the capabilities of the Jackrabbit. Schematics for both boards are included on the CD-ROM in PDF format.

The Jackrabbit board has a standard Rabbit programming connector, which is a 10-pin header with a 2 mm pitch. A programming cable is used to connect a PC serial port (COM port) to the Jackrabbit board. The programming cable has a level converter board in the middle of the cable since the programming connector supports CMOS logic levels, and not the higher voltage RS-232 levels that are used by PC serial ports. When the programming cable is connected, Dynamic C running on the PC can hard-reset the Jackrabbit board and cold-boot it. The cold boot includes compiling and downloading a BIOS program that stays resident while you work. If you crash the target, Dynamic C will automatically reboot and recompile the BIOS if it senses that a target communication error occurred.

You have a choice of doing your software development in the flash memory or in the static RAM included on the Jackrabbit board. There are 128K bytes in each memory. Some versions of the Jackrabbit board have only 32K bytes of static RAM. If you use one of these boards, you must do development in flash memory. The advantage of working in RAM is to save wear on the flash, which is limited to about 100,000 writes.

Note that an application can be developed in RAM, but cannot run standalone from RAM after the programming cable is disconnected. All applications can only run from flash.

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

Dynamic C provides a number of debugging features. You can single-step your program, either in C, statement by statement, or in assembly language, instruction by instruction. You can set break points, where the program will stop, on any statement. You can evaluate watch expressions. A watch expression is any C expression that can be evaluated in the context of the program. If the program is at a break point, a watch expression can view any expression using local or external variables. If the program is running and a call to the debugger is included in the user's code (**`runwatch();`**), it is possible to evaluate watch expressions using global variables only while the target program continues to run, slowed down only by the need to refresh a display in response to a **<Ctrl-U>** command.

2. GETTING STARTED INSTRUCTIONS

Chapter 2 contains detailed instructions for installing the software on your PC and for connecting the Jackrabbit board to your PC in order to run sample programs.

2.1 Software Installation

You will need approximately 20 megabytes of free space on your hard disk to install and run Dynamic C. The software can be installed on your C drive or any other convenient drive.

2.2 Getting Hooked Up

Figure 1 below shows an overview of how the serial and power connections are made to the Jackrabbit board, the Prototyping Board, and to your PC.

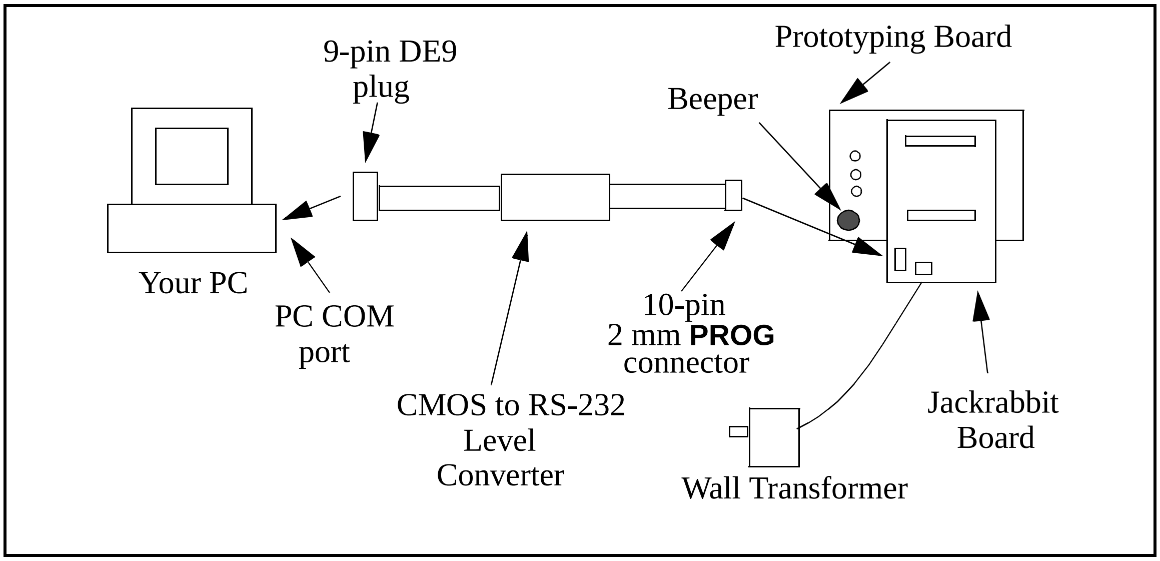
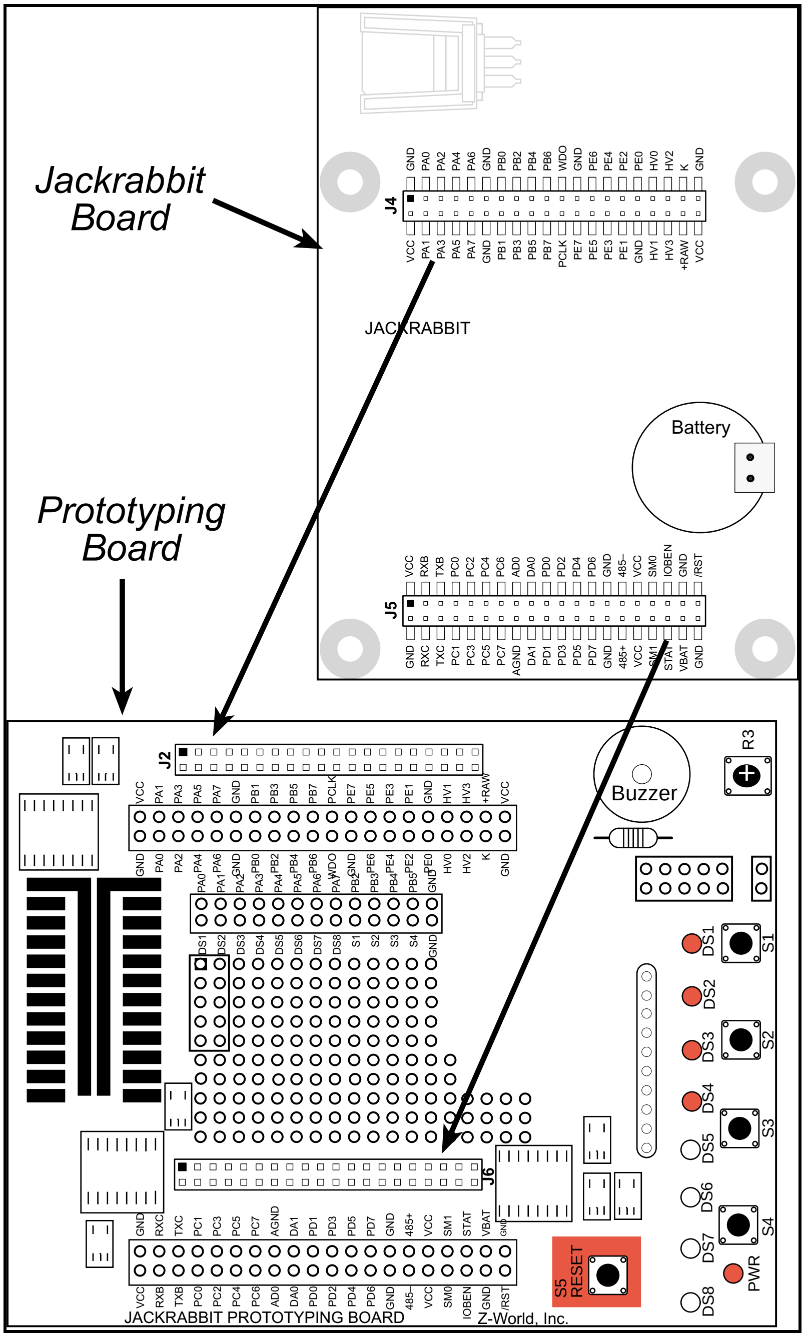


Figure 1. Jackrabbit Hookup Connections

2.2.1 Prototyping Board

To attach the Jackrabbit board to the Prototyping Board, turn the Jackrabbit board over so that the battery is facing up. Plug headers J4 and J5 into the sockets on the Prototyping Board as indicated in Figure 2.



Jackrabbit Board
Prototyping Board
J4
GND PA0 PA2 PA4 PA6 GND PB0 PB2 PB4 PB6 WDO GND PE6 PE4 PE2 PE0 HV0 HV2 K GND
VCC PA1 PA3 PA5 PA7 GND PB1 PB3 PB5 PB7 PCLK PE7 PE5 PE3 PE1 GND HV1 HV3 +RAW VCC
JACKRABBIT
Battery
J5
VCC RXB TXB PC0 PC2 PC4 PC6 AD0 DA0 PD0 PD2 PD4 PD6 GND 485– VCC SM0 IOBEN GND /RST
GND RXC TXC PC1 PC3 PC5 PC7 AGND DA1 PD1 PD3 PD5 PD7 GND 485+ VCC SM1 STAT VBAT GND
J2
J6
Buzzer
R3
DS1 DS2 DS3 DS4 DS5 DS6 DS7 DS8
S1 S2 S3 S4
PWR
S5 RESET
JACKRABBIT PROTOTYPING BOARD
Z-World, Inc.

Figure 2. Attaching Jackrabbit Board to Prototyping Board

2.2.2 Jackrabbit Board

1. Connect the 10-pin **PROG** connector of the programming cable to header J3 on the Jackrabbit board as shown in Figure 3. (If your programming cable has only one unlabeled 10-pin connector, attach that connector to header J3 on the Jackrabbit board.) Connect the other end of the programming cable to a COM port on your PC. Note that COM1 is the default COM port used by Dynamic C.

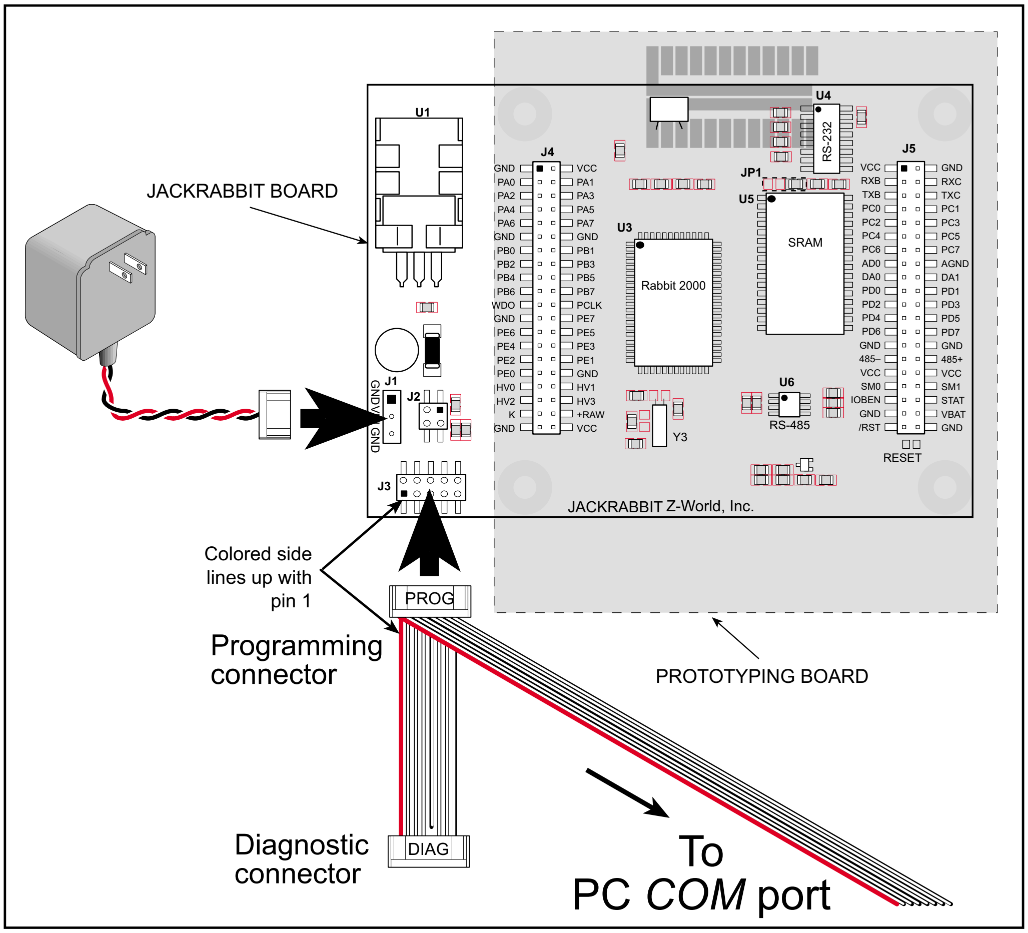


Figure 3. Power and Programming Cable Connections to Jackrabbit Board

2. Hook up the connector from the wall transformer to header J1 on the Jackrabbit board as shown in Figure 3. 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.

3. Plug in the wall transformer. The Jackrabbit board and the Protyping Board are ready to be used.

A RESET button is provided on the Protyping Board (see Figure 2) to allow a harware reset.

2.3 Starting Dynamic C

Once the Jackrabbit board is connected as described in the preceding section, start Dynamic C by double-clicking on the Dynamic C icon or by double-clicking on **`dwc.exe`** in the Dynamic C directory.

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

If you receive the "BIOS successfully compiled …" message after pressing **<Ctrl-Y>** or starting Dynamic C, and this message is followed by "Target not responding," it is possible that your PC cannot handle the 115,200 bps baud rate. Try changing the baud rate to 57,600 bps as follows.

1. Open the BIOS source code file. If you are debugging in flash, this file is named **`JRABBIOS.C`**, and can be found in the **`BIOS`** directory. The **`JRAMBIOS.C`** file in the **`BIOS`** directory can be used for debugging in RAM.
2. Change the line

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

   to read as follows.

   ```
   #define USE115KBAUD 0   // set to 0 to use 57600 baud
   ```
3. Locate the **Serial options** dialog in the Dynamic C **Options** menu. Change the baud rate to 57,600 bps, then press **<Ctrl-Y>**.

If you receive the "BIOS successfully compiled …" message and do not receive a "Target not responding" message, the target is now ready to compile a user program.

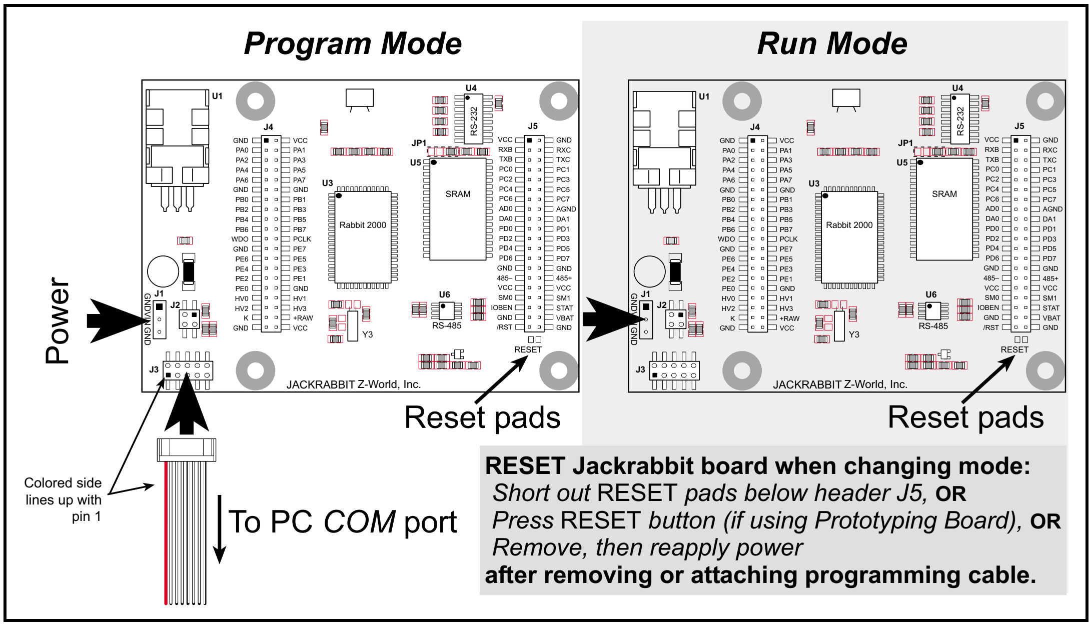
3. SUBSYSTEMS

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

- Switching Between Program Mode and Run Mode
- Digital Inputs/Outputs
- A/D Converter
- D/A Converters
- High-Power Outputs
- Serial Communication
- Memory

3.1 Switching Between Program Mode and Run Mode

The Jackrabbit is automatically in Program Mode when the programming cable is attached, and is automatically in Run Mode when no programming cable is attached. See Figure 4.



RESET Jackrabbit board when changing mode:
Short out RESET *pads below header J5,* **OR**
Press RESET *button (if using Prototyping Board),* **OR**
Remove, then reapply power
after removing or attaching programming cable.

Figure 4. Jackrabbit Program Mode and Run Mode Setup

3.1.1 Detailed Instructions: Changing from Program Mode to Run Mode

1. Disconnect the programming cable from header J3 of the Jackrabbit board.
2. Reset the Jackrabbit board. You may do this as explained in Figure 4. Figure 5 shows the location of the RESET button on the Prototyping Board.

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

3.1.2 Detailed Instructions: Changing from Run Mode to Program Mode

1. Attach the programming cable to header J3 on the Jackrabbit board.
2. Reset the Jackrabbit board. You may do this as explained in Figure 4. Figure 5 shows the location of the RESET button on the Prototyping Board.

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

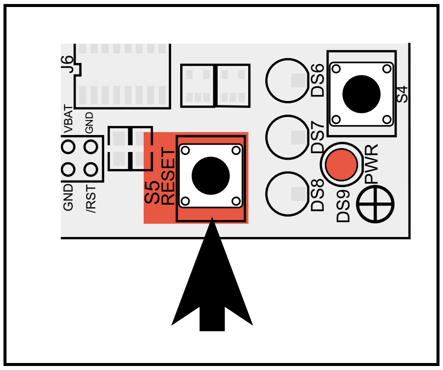


Figure 5. Location of Prototyping Board Reset Button

3.2 Jackrabbit Inputs and Outputs

Figure 6 shows the Jackrabbit digital inputs/outputs, high-power outputs, serial ports, analog-to-digital converter, and digital-to-analog converters.

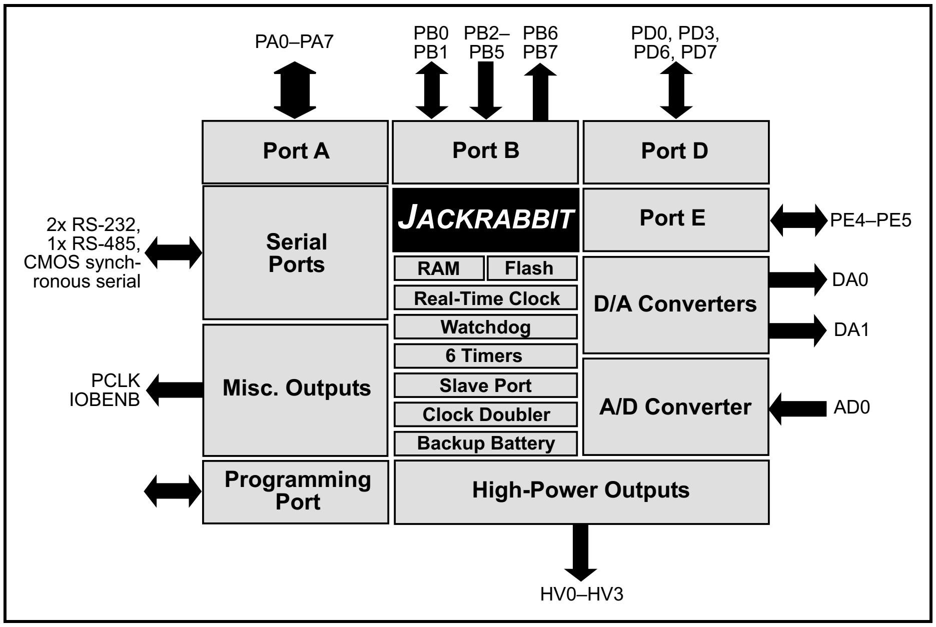
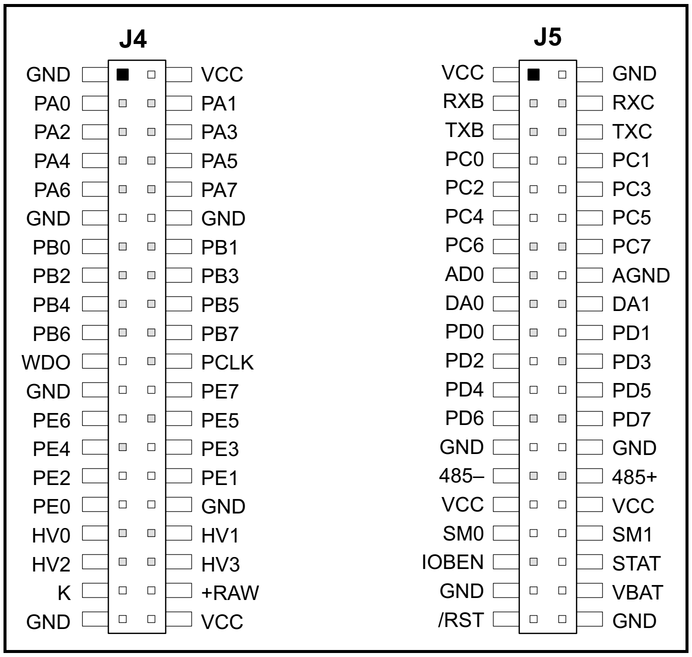


Figure 6. Jackrabbit Subsystems

Figure 7 shows the pinout for headers J4 and J5, which carry the signals associated with the Jackrabbit subsystems.



J4		J5	
GND	VCC	VCC	GND
PA0	PA1	RXB	RXC
PA2	PA3	TXB	TXC
PA4	PA5	PC0	PC1
PA6	PA7	PC2	PC3
GND	GND	PC4	PC5
PB0	PB1	PC6	PC7
PB2	PB3	AD0	AGND
PB4	PB5	DA0	DA1
PB6	PB7	PD0	PD1
WDO	PCLK	PD2	PD3
GND	PE7	PD4	PD5
PE6	PE5	PD6	PD7
PE4	PE3	GND	GND
PE2	PE1	485–	485+
PE0	GND	VCC	VCC
HV0	HV1	SM0	SM1
HV2	HV3	IOBEN	STAT
K	+RAW	GND	VBAT
GND	VCC	/RST	GND

Figure 7. Pinout for Jackrabbit Headers J4 and J5

The ports on the Rabbit 2000 microprocessor used in the Jackrabbit are configurable, and so the factory defaults can be reconfigured. Table 2 lists the Rabbit 2000 factory defaults and the alternate configurations.

Table 2. Jackrabbit Pinout Configrations

Pin	Rabbit 2000 Factory Default	Alternate Use	Jackrabbit Use
PA0–PA7	Parallel I/O	Slave port data bus SD0–SD7	
PB0	Parallel I/O	Serial port clock CLKB	PB1 (CLKA) is connected to J3 (programming port)
PB1		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 2. Jackrabbit Pinout Configrations (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

3.3 Digital Inputs/Outputs

The Jackrabbit has 24 general-purpose digital inputs/outputs available on headers J4 and J5—16 are bidirectional, four are inputs only, and 4 are outputs only, as shown in Figure 6.

The 16 bidirectional inputs are located on pins PA0–PA7, PB0–PB1, PD0, PD3, PD6, PD7, PE4, and PE5. The locations of these pins are shown in Figure 8.

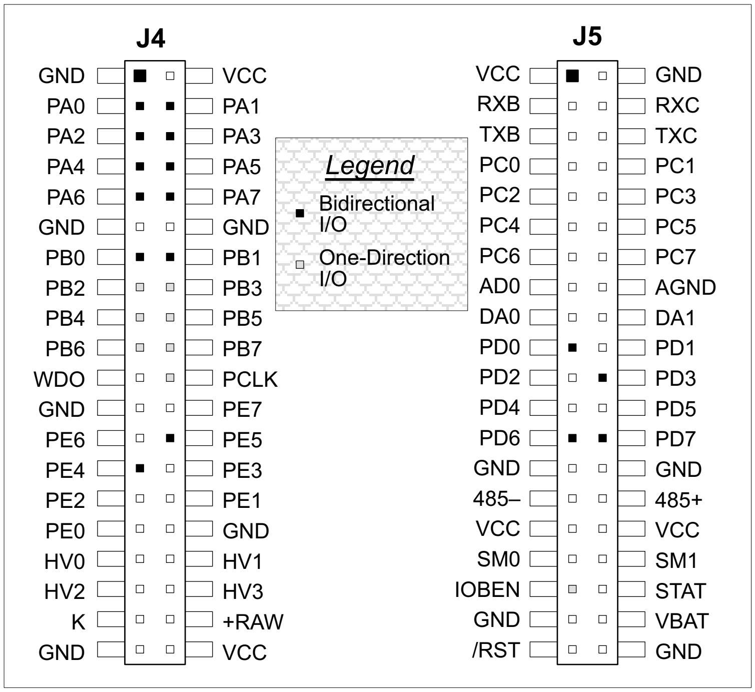


Figure 8. Jackrabbit I/O Pins

As shown in Table 2, 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.

3.3.1 Inputs

The four input-only are located on PB2–PB5. These pins can instead be used with the slave port of the Rabbit 2000 microprocessor.

3.3.2 Outputs

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.

3.4 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.

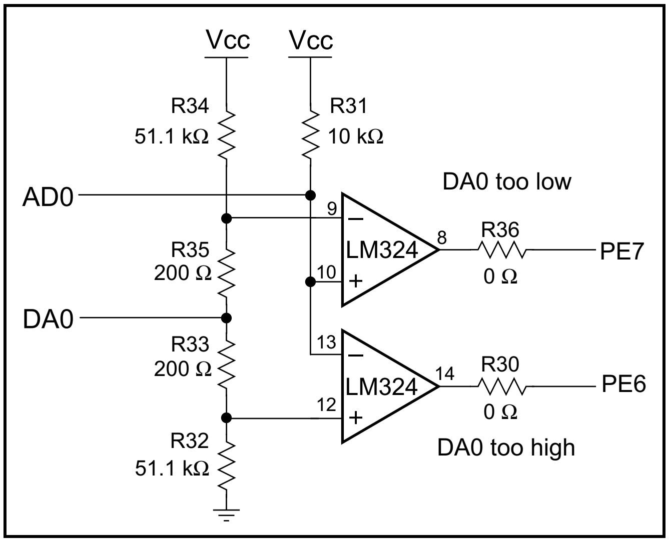


Figure 9. Schematic Diagram of A/D Converter

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.

3.5 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.

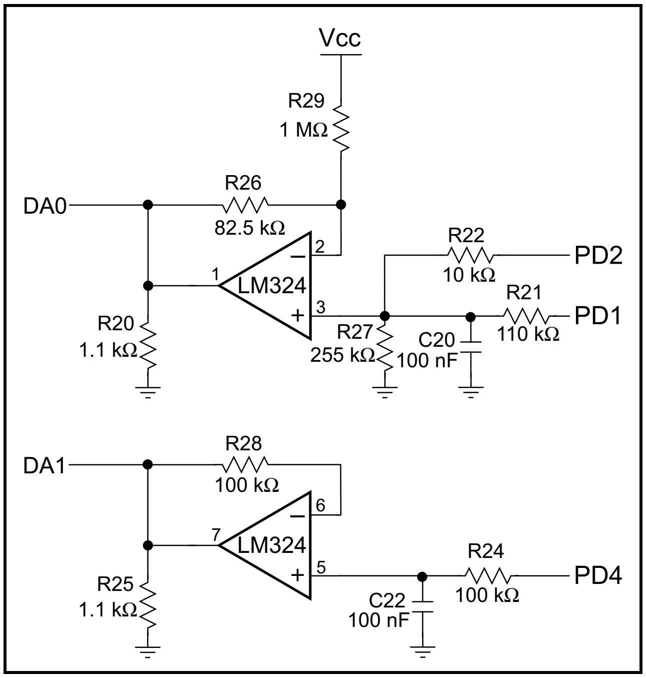


Figure 10. Schematic Diagram of D/A Converters

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.

3.5.1 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

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

where RC = 0.01 s for 14.7 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 14.7 MHz clock is (14.7 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 14.7 MHz Jackrabbit has t = 70 µs. Based on the standard capacitor discharge formula, this means that the maximum voltage change will be

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

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

Table 3. 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

$$V = V_P - (V_P - V_{DA1})e^{(-t/RC)}$$

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.

3.5.2 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 4 lists typical DA0 voltages measured for various duty cycle values with a load larger than 1 MΩ.

Table 4. 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

$$V = V_P - (V_P - V_{DA0})e^{(-t/RC)}$$

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 14.7 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.

3.6 High-Power Outputs

The Jackrabbit board has four high-power outputs, HV0–HV3, connected to the Rabbit 2000 pins E0–E3 (Port E). Three high-power outputs, HV0–HV2, are always sinking; HV3 is factory-configured to be a sourcing output, but can be reconfigured to provide a fourth sinking output.

3.6.1 High-Power Sinking Outputs (HV0–HV2)

Three high-power outputs, HV0–HV2, are always sinking. The sinking configuration is shown schematically in Figure 11.

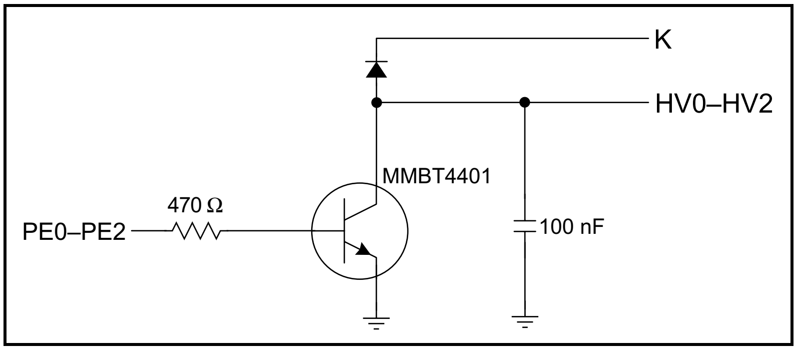


Figure 11. High-Power Sinking Outputs

The transistor simply shorts the high-voltage output to ground when the respective PE0–PE2 signal goes high. The current-sinking capability is limited only by the current gain of the transistor. The Rabbit 2000 will source about 8 mA per channel and, given a conservative transistor current gain of 25, this allows a normal 4401 NPN transistor, which is installed on the BL1810 and BL1820, to sink up to 200 mA. A high-performance transistor such as the Zetex FMMT619, which is installed on the BL1800, can sink up to 1 A. The collectors of these transistors are protected against inductive loads by a diode-capacitor combination.

K is limited to a maximum of 30 V by the power dissipation of resistors R51 and R52 in the sourcing output since a common K is used for all four high-power outputs.

3.6.2 Configurable High-Power Output (HV3)

HV3, shown schematically in Figure 12, is factory-configured to be a sourcing output.

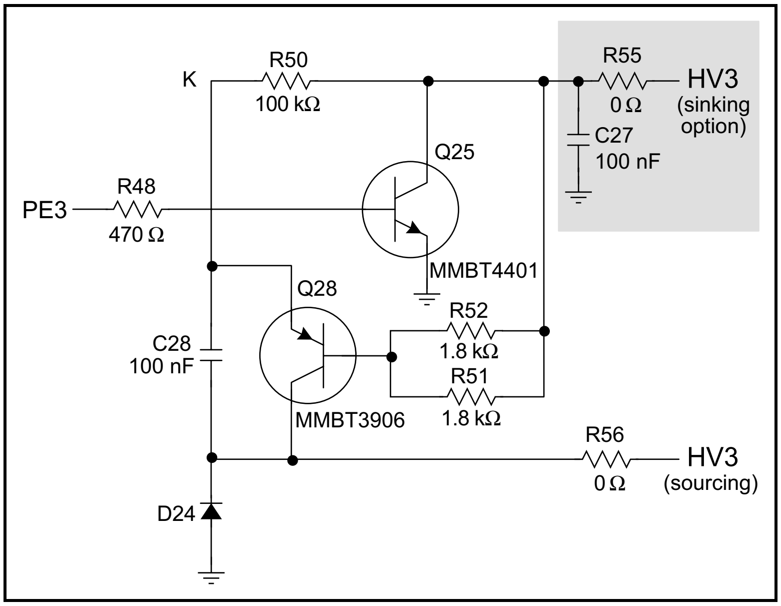


Figure 12. Configurable High-Current Output

When used as a sourcing output, HV3 is shorted to K when PE3 on the Rabbit 2000 goes high, and the two transistors shown in Figure 12 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 13 shows the location of these components.

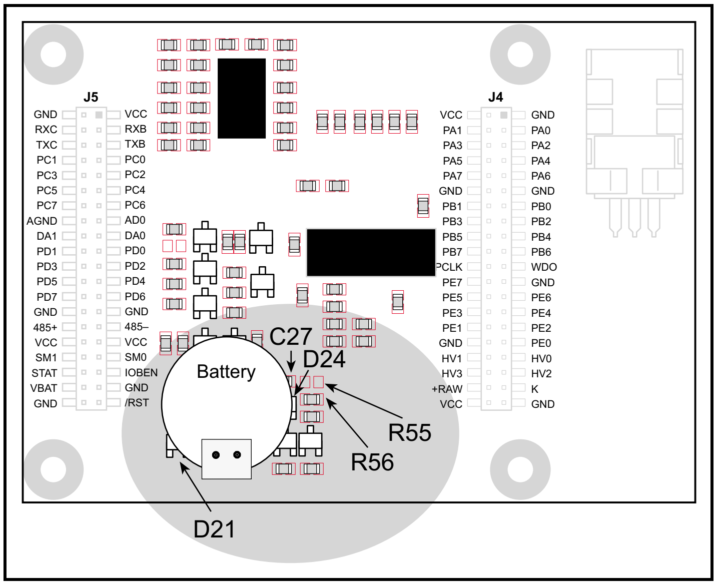


Figure 13. Changing HV3 to a Sinking Output

3.6.3 Connecting a Load to the High-Power 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-current output circuits.
2. A diode-capacitor combination in the circuit “snubs” voltage transients when inductive loads such as relays and solenoids are driven.

Note that the same K supply is used for all four high-power outputs. Figure 14 shows the connection of a load to the sinking and sourcing outputs. The load’s supply has a common ground with the rest of the Jackrabbit.

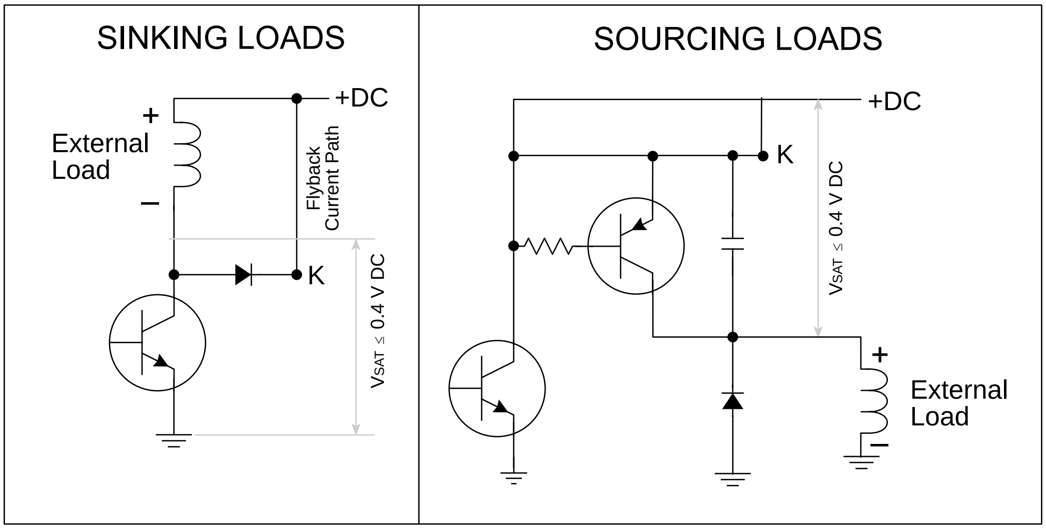


Figure 14. Connecting Loads to Sinking and Sourcing Outputs

3.7 Serial Communication

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

3.7.1 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.

3.7.2 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 15.

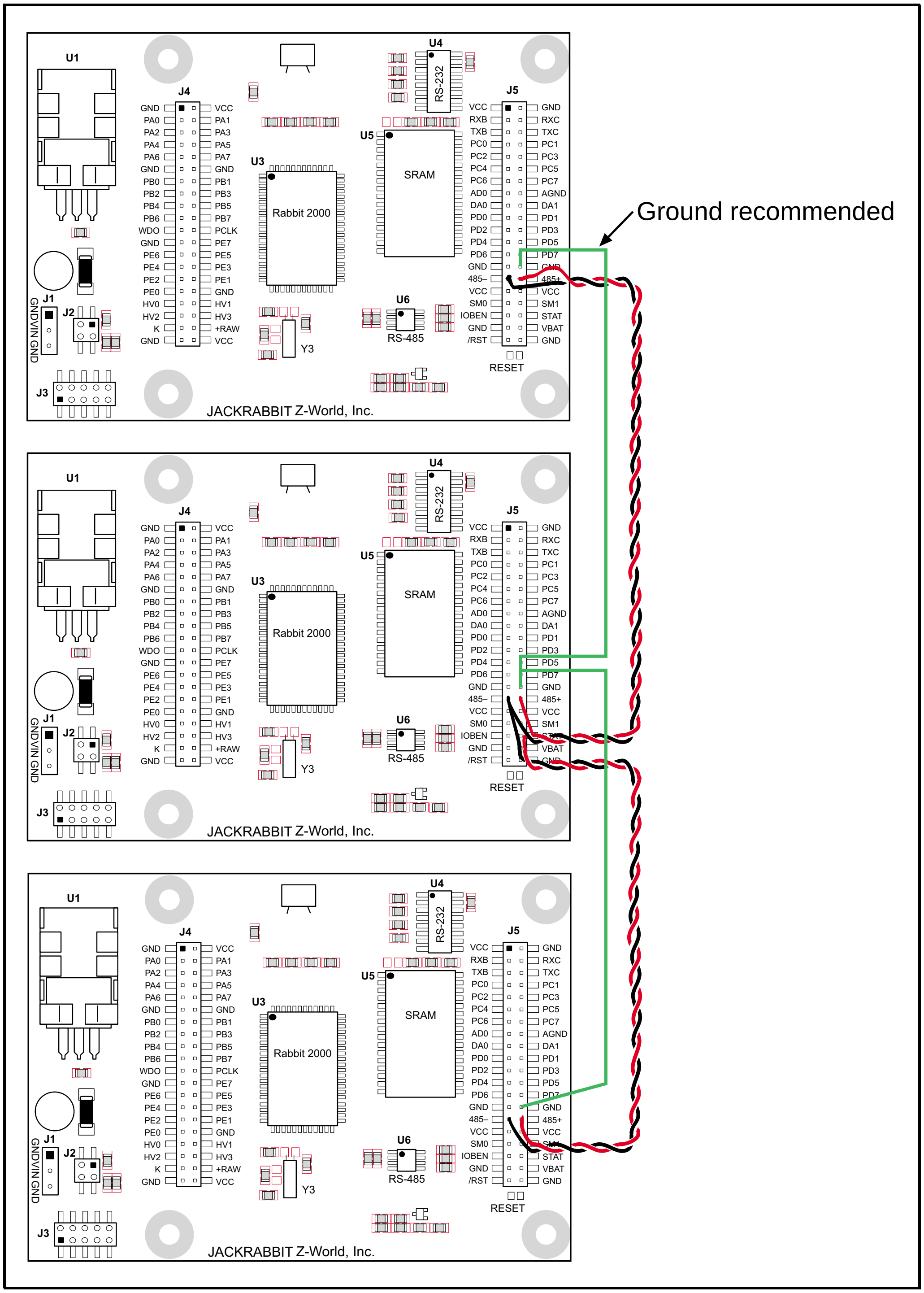


Figure 15. Multidrop Jackrabbit Network

The Jackrabbit comes with a 220 Ω termination resistor and 681 Ω bias resistors already installed, as shown in Figure 16.

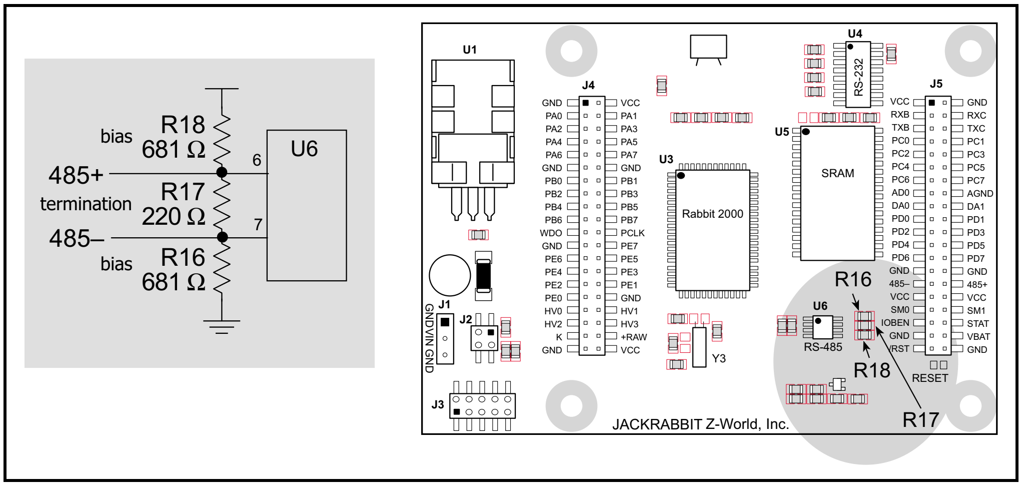

bias
R18
681 Ω
485+
termination
R17
220 Ω
485–
bias
R16
681 Ω
6
7
U6
U1
J4
GND PA0 PA2 PA4 PA6 GND PB0 PB2 PB4 PB6 WDO GND PE6 PE4 PE2 PE0 HV0 HV2 K GND
VCC PA1 PA3 PA5 PA7 GND PB1 PB3 PB5 PB7 PCLK PE7 PE5 PE3 PE1 GND HV1 HV3 +RAW VCC
U3
Rabbit 2000
U4
RS-232
U5
SRAM
J5
VCC RXB TXB PC0 PC2 PC4 PC6 AD0 DA0 PD0 PD2 PD4 PD6 GND 485– VCC SM0 IOBEN GND /RST
GND RXC TXC PC1 PC3 PC5 PC7 AGND DA1 PD1 PD3 PD5 PD7 GND 485+ VCC SM1 STAT VBAT GND
J1
GND VIN GND
J2
J3
Y3
U6
RS-485
R16
R18
R17
RESET
JACKRABBIT Z-World, Inc.

Figure 16. 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.

3.7.3 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. The Rabbit 2000 startup-mode pins (SMODE0, SMODE1) are presented to the programming port so that an externally connected device can force the Jackrabbit to start up in an external bootstrap mode.

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

The Jackrabbit can be reset from the programming port.

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

The clock line for serial port A is presented to the programming port, which makes fast serial communication possible.

3.8 Memory

3.8.1 SRAM

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

The existing standard models of the Jackrabbit Series come with 128K of SRAM. A factory-installed option for 512K of SRAM is available. Figure 17 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.

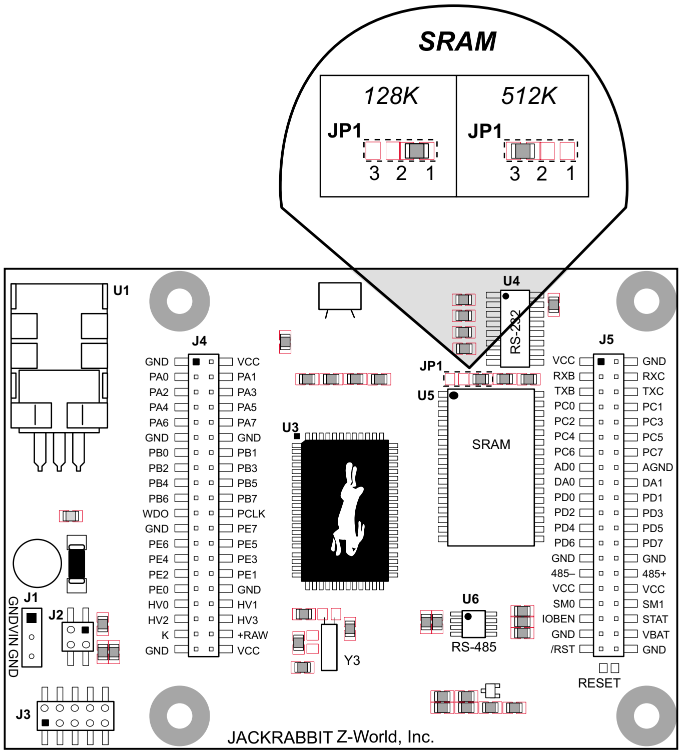

SRAM
128K
JP1
3 2 1
512K
JP1
3 2 1
U1
U4
RS-232
J4
GND PA0 PA2 PA4 PA6 GND PB0 PB2 PB4 PB6 WDO GND PE6 PE4 PE2 PE0 HV0 HV2 K GND
VCC PA1 PA3 PA5 PA7 GND PB1 PB3 PB5 PB7 PCLK PE7 PE5 PE3 PE1 GND HV1 HV3 +RAW VCC
JP1
U5
SRAM
U3
J5
VCC RXB TXB PC0 PC2 PC4 PC6 AD0 DA0 PD0 PD2 PD4 PD6 GND 485– VCC SM0 IOBEN GND /RST
GND RXC TXC PC1 PC3 PC5 PC7 AGND DA1 PD1 PD3 PD5 PD7 GND 485+ VCC SM1 STAT VBAT GND
U6
RS-485
Y3
J1
GND VIN GND
J2
J3
RESET
JACKRABBIT Z-World, Inc.

Figure 17. Jackrabbit Jumper Settings for SRAM Size

☎ For ordering or other information involving the factory-intsalled 512K SRAM option, call your Z-World Sales Representative at (530)757-3737.

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

3.8.2 Flash EPROM

The Jackrabbit is also designed to accept 128K to 512K of small-sector-size flash EPROM packaged in a TSOP case. Z-World recommends that Jackrabbit applications should not be constrained by the sector size of the flash EPROM since it may be necessary to use large-sector-size flash EPROM in the future.

4. Software Reference

4.1 More About Dynamic C

Dynamic C has been in use worldwide since 1989. Dynamic C is specially designed for programming embedded systems. Dynamic C features quick compile and interactive debugging in the real environment. A complete reference to Dynamic C is contained in the ***Dynamic C Reference Manual***.

Dynamic C for Rabbit™ processors uses the standard Rabbit programming interface. This is a 10-pin connector that connects to the Rabbit serial port A. It is possible to reset and cold-boot a Rabbit processor via the programming port. No software needs to be present in the target system. More details are available in the ***Rabbit 2000 Microprocessor User's Manual***.

Dynamic C cold-boots the target system and compiles the BIOS. The BIOS is a basic program of a few thousand bytes in length that provides the debugging and communication facilities that Dynamic C needs. Once the BIOS has been compiled, the user can compile his own program and test it. If the BIOS fails because of a crash, a new cold boot and BIOS compile can be done at any time.

Each type of Rabbit microprocessor system can have a different BIOS, or the BIOS program can be customized by using **`#define`** options. The Jackrabbit board is supplied with several different BIOS, one for using flash memory to hold the program, and one to use RAM memory to hold the program. RAM memory is useful for holding a program while debugging is being done because it is more flexible than flash memory.

Dynamic C does not use include files, rather it has libraries which are used for the same purpose, that is, to supply function prototypes to programs before they are compiled. Libraries are much easier to use compared to include files.

Dynamic C supports assembly language, either as separate programs or as fragments embedded in C programs. Interrupt routines may be written in Dynamic C or in assembly language.

4.1.1 Operating System Framework

Dynamic C does not include an operating system in the usual sense of a complex software system that is resident in memory. The user has complete control of what is loaded as a part of his program, other than those routines that support loading and debugging and which are inactive at embedded run time. However, certain routines are very basic and normally should always be present and active.

- Periodic interrupt routine. This interrupt routine is driven by the Rabbit periodic interrupt facility, and when enabled creates an interrupt every 16 ticks of the 32.768 kHz oscillator, or every 488 µs. This routine drives three long global variables that keep track of the time: **`SEC_TIMER`**, **`MS_TIMER`**, and **`TICK_TIMER`** that respectively count seconds, milliseconds, and 488 µs ticks. These variables are needed by virtually all functions that measure time. The **`SEC_TIMER`** is set to seconds elapsed since 1 Jan 1980, and thus also keeps track of the time and date. The periodic interrupt routine must be disabled when the microprocessor enters sleepy mode and the processor clock is operating at 32.768 kHz. The interrupt routine cannot complete at this slow speed before the next tick of the periodic interrupt. In this situation, the hardware real-time clock can be read directly to provide the time.

- Watchdog support routines. Although the Rabbit watchdog can be disabled, this is not recommended since the watchdog is an essential facility for recovering from crashes. Very few systems are crash-free in real life.

4.2 I/O Drivers

The Jackrabbit board 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.

4.2.1 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.

4.2.2 Digital Output

The Jackrabbit board 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 BL1810 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.

`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).

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

4.2.3 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 below 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.

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 board can be more accurately calibrated for each individual Jackrabbit board 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:

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

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

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

4.2.4 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.

📖 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`**.

📖 See sample program **`JRIO_COF.C`** for an example of the use of **`cof_anaIn`**.

4.3 RS-232 Serial Communication Drivers

The interface to the Rabbit serial library, **`RS232.LIB`**, is designed to provide users with a set of functions that send and receive entire blocks of data without yielding to other tasks, a set of single user cofunctions that send and receive data but yield to other tasks, and a set of circular buffer functions.

The naming convention is **`serXfn`**:

ser - serial

X - the port being used: A, B, C, or D

fn - the function being implemented

For example, **`serBgetc()`** is the serial port B function **`getc()`**, which returns a character.

The Rabbit serial functions are listed in the following groups.

Open and Close Functions

Non-Cofunction Blocking Input Functions

Non-Cofunction Blocking Output Functions

Single-User Cofunction Input Functions

Single-User Cofunction Output Functions

Circular Buffer Functions

4.3.1 Open and Close Functions

The open and close functions enable and disable serial communication over the specified port.

`int serXopen (long baud);`

Currently only 8N1 transmission (8 data bits, no parity, 1 stop bit) is supported. The **`open`** function sets up the interrupt service routine vector.

Parameters

`baud`—desired baud rate in bits per second

Return Value

1—The baud rate set on the Rabbit is the same as the input baud rate.

0—The baud rate set on the rabbit does not match the input baud rate.

`int serXclose ( );`

Disables the serial port interrupt service routine.

Parameters

None.

Return Value

1

4.3.2 Non-Cofunction Blocking Input Functions

These are simple functions that do not use Dynamic C costatements. If no input data are available when called, they return immediately with appropriate status information in their return value. Once they begin to receive characters, they do not yield to other tasks until they complete their operation or until a character-to-character timeout period elapses.

`int serXgetc ( );`

Gets a single character. Always returns immediately, either with the next available input byte, or with –1 if none is available.

Parameters

None

Return Value

An integer with return character in the low byte. No character is represented by a return of –1.

`int serXread (void *data, int length, unsigned long tmout);`

Reads a block of characters. Returns the number of bytes read from an input serial stream. The stream is considered to be ended when all **`length`** bytes have been read or when the timeout period elapses waiting for data to appear in the input buffer.

Parameters

`data`—Destination data structure. The user must ensure data is allocated for at least length bytes.

`length`—The number of bytes to read.

`tmout`—The number of milliseconds to wait for receipt of each byte before timing out.

Return Value

The number of bytes read into data until timed out or until all length bytes have been read.

4.3.3 Non-Cofunction Blocking Output Functions

These are simple functions that do not use Dynamic C costatements. They immediately begin to perform their task, not yielding to other tasks until all characters have been written.

`int serXputc (char c);`

Writes a character to the serial port.

Parameters

`c`—Character to write

Return Value

1 for success, 0 if the character could not be written to the port.

`int serXputs (char *s);`

Calls `serXwrite (s, strlen (s))`.

Parameters

`s`—Null-terminated character string source to write to the serial port.

Return Value

The number of characters written.

`int serXwrite (void *data, int length);`

Writes a block of `length` bytes to the serial port.

Parameters

`data`—Destination data structure. The user must ensure data is allocated for at least length bytes.

`length`—The number of bytes to read.

Return Value

The number of bytes written to the serial port.

4.3.4 Single-User Cofunction Input Functions

These are Dynamic C cofunctions. If the input buffer they use is locked or becomes full during the course of their operation, they yield to other tasks, but do not return to execute the next statement within their own costatement block until they have completed their operation.

`scofunc int cof_serXgetc ( );`

Reads a single character from the serial port, yielding when not successful, and only returning when a character is successfully read.

Parameters

None

Return Value

An integer with the character read in the low byte.

`scofunc int cof_serXgets(char *s, int length, unsigned long tmout);`

Reads a null-terminated string, completes its execution when a carriage return is read, **`length`** number of characters are read, or the character to character timeout period elapses after the first character is read. It yields to other tasks while the input buffer is locked or becomes empty during its execution, and only returns control to the following statement in its own costatement block when it completes.

Parameters

`data`—Destination data structure. The user must ensure data is allocated for at least length bytes.

`length`—The number of bytes to read.

`tmout`—The number of milliseconds to wait for each character after the first character is read.

Return Value

1—CR or **`length`** bytes read into **`s`**.

0—Function times out before reading CR or **`length`** bytes.

`scofunc int cof_serXread(void *data, int length, unsigned long tmout);`

Reads a block of characters, completes its execution when `length` number of characters are read, or the character-to-character timeout period elapses after the first character is read. It yields to other tasks while the input buffer is locked or becomes empty during its execution and only returns control to the following statement in its own costatement block when it completes.

Parameters

`data`—Destination data structure. The user must ensure data is allocated for at least length bytes.

`length`—The number of bytes to read.

`tmout`—The number of milliseconds to wait for each character after the first.

Return Value

The number of bytes read.

4.3.5 Single-User Cofunction Output Functions

These are Dynamic C cofunctions. If the output buffer they use is locked or becomes empty during the course of their operation, they yield to other tasks, but do not return to execute the next statement within their own costatement block until they have completed their operation.

`scofunc void cof_serXputc (char c);`

Writes a single character to the serial port, yielding to other tasks when unsuccessful, and returning only when the character is successfully written.

Parameters

`c`—Character to write to the serial port.

Return Value

None

`scofunc void cof_serXputs(char *s);`

Writes a null-terminated character string to the serial port, yielding to other tasks when unsuccessful or whenever the buffer is full, returning only when the string is successfully written.

Parameters

`s`—Null-terminated character string written to the serial port.

Return Value

None

`scofunc void cof_serXwrite (void *data, int length);`

Writes a block of characters to the serial port, yielding to other tasks when unsuccessful or whenever the buffer is full, returning only when all the data is successfully written.

Parameters

`data`—Source data structure to write to the serial port.

`length`—Number of characters in data to write.

Return Value

None

4.3.6 Circular Buffer Functions

These functions act on or report status of the circular transmit/receive buffers.

Macro definitions are used to establish the buffer sizes:

xINBUFSIZE—read buffer size, where **x** is A, B, C, or D

xOUTBUFSIZE—write buffer size where **x** is A, B, C, or D

The user must define each buffer size for each port being used to be a power of 2 minus 1 with a macro. The size of 2^n - 1 enables masking for fast rollover calculations. If no value or an illegal value is defined, a default size of 31 will be used and a compiler warning will be given. When using cofunctions, smaller buffer sizes can yield more frequently to other tasks, but have the risk of a large input data stream overrunning the buffer and losing data if the other task executes for too long relative to the baud rate.

int serXpeek ( );

Returns the first character in the receive buffer, if any are available, without removing it from the buffer.

Parameters

None

Return Value

An integer with return character in the low byte. No character is represented by a return of –1.

void serXrdFlush ( );

Flushes the serial port receive buffer.

Parameters

None

Return Value

None

void serXwrFlush ( );

Flushes the serial port transmit buffer.

Parameters

None

Return Value

None

`int serXrdFree ( );`

Calculates the free space in the serial port receive buffer.

Parameters

None

Return Value

The number of characters the serial port receive buffer can accept before becoming full.

`int serXwrFree ( );`

Calculates the free space in the serial port transmit buffer.

Parameters

None

Return Value

The number of characters the serial port transmit buffer can accept before becoming full.

`int serXrdUsed ( );`

Calculates the number of characters ready to read from the serial port receive buffer.

Parameters

None

Return Value

The number of characters currently in the serial port receive buffer.

4.4 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.

Jr485Init()

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

Jr485Tx()

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

Jr485Rx()

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

The following sample program illustrates the use of the RS-485 serial drivers. The sample program shows a byte being trasmitted, 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 );
      }
   }
}
```

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

APPENDIX A. SPECIFICATIONS

Table A-1 to Table A-3 list the electrical, mechanical, and environmental specifications for the Jackrabbit boards.

Table A-1. Jackrabbit (BL1800) Board Specifications

Parameter	Specification
Board Size	2.50" × 3.50" × 0.76" (64 mm × 89 mm × 19 mm)
Operating Temperature	–40°C to +70°C
Humidity	5% to 95%, noncondensing
Input Voltage and Current	8 V to 40 V DC, 49 mA typical at 24 V DC, 91 mA typical at 12 V DC, switching regulator
General-purpose I/O	16 bidirectional, 4 inputs, 4 outputs, CMOS compatible
Analog Inputs	One low-grade A/D input—input range 0.1 V to 2.8 V, 9-bit resolution, 8-bit accuracy, average acquisition time 75 ms (83 ms maximum)
Analog Outputs	Two filtered and buffered PWM outputs
Digital Outputs	Four high-current, high-voltage outputs—3 sink up to 1 A and 30 V standoff, 1 sources up to 500 mA
Microprocessor	Rabbit 2000
Clock	29.49 MHz
SRAM	128K (supports 32K–512K)
Flash EPROM	256K (supports 128K–512K)
Timers	Five 8-bit timers, one 10-bit timer with two match registers, five timers are cascadable
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 • one 5 V CMOS-compatible programming port Two serial ports (A and B) can be clocked.
Serial Rate	RS-232 up to 115,200 bps, RS-485 up to 250,000 bps, CMOS up to 7.37 Mbps
Watchdog/Supervisor	Yes
Time/Date Clock	Yes
Backup Battery	Yes, 3 V lithium coin type, 950 mA-h, rated at –40°C to +85°C

Table A-2. Jackrabbit (BL1810) Board Specifications

Parameter	Specification
Board Size	2.50" × 3.50" × 0.94" (64 mm × 89 mm × 24 mm)
Operating Temperature	–40°C to +70°C
Humidity	5% to 95%, noncondensing
Input Voltage and Current	7.5 V to 25 V DC, 100 mA typical, linear regulator
General-purpose I/O	16 bidirectional, 4 inputs, 4 outputs, CMOS compatible
Analog Inputs	One low-grade A/D input—input range 0.1 V to 2.8 V, 9-bit resolution, 8-bit accuracy, average acquisition time 150 ms (165 ms maximum)
Analog Outputs	Two filtered and buffered PWM outputs
Digital Outputs	Four high-current, high-voltage outputs—3 sink up to 200 mA and 30 V standoff, 1 sources up to 100 mA
Microprocessor	Rabbit 2000
Clock	14.732 MHz
SRAM	128K (supports 32K–512K)
Flash EPROM	128K (supports 128K–512K)
Timers	Five 8-bit timers, one 10-bit timer with two match registers, five timers are cascadable
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 • one 5 V CMOS-compatible programming port Two serial ports (A and B) can be clocked.
Serial Rate	RS-232 up to 115,200 bps, RS-485 up to 250,000 bps, CMOS up to 3.69 Mbps
Watchdog/Supervisor	Yes
Time/Date Clock	Yes
Backup Battery	Yes, 3 V lithium coin type, 950 mA-h, rated at –20°C to +60°C

Table A-3. Jackrabbit (BL1820) Board Specifications

Parameter	Specification
Board Size	2.50" × 3.50" × 0.63" (64 mm × 89 mm × 16 mm)
Operating Temperature	–40°C to +70°C
Humidity	5% to 95%, noncondensing
Input Voltage and Current	7.5 V to 25 V DC, 100 mA typical , linear regulator
General-purpose I/O	17 bidirectional, 5 inputs, 5 outputs, CMOS compatible
Analog Inputs	One low-grade A/D input—input range 0.1 V to 2.8 V, 9-bit resolution, 8-bit accuracy, average acquisition time 150 ms (165 ms maximum)
Analog Outputs	Two filtered and buffered PWM outputs
Digital Outputs	Four high-current, high-voltage outputs—3 sink up to 200 mA and 30 V standoff, 1 sources up to 100 mA
Microprocessor	Rabbit 2000
Clock	14.732 MHz
SRAM	128K (supports 32K–512K)
Flash EPROM	128K (supports 128K–512K)
Timers	Five 8-bit timers, one 10-bit timer with two match registers, five timers are cascadable
Serial Ports	Up to four serial ports: • two RS-232 or one RS-232 (with CTS/RTS) rated at 15 kV ESD • two 5 V CMOS-compatible serial ports (one used as programming port) Two serial ports (A and B) can be clocked.
Serial Rate	RS-232 up to 115,200 bps, CMOS up to 3.69 Mbps
Watchdog/Supervisor	Yes
Time/Date Clock	Yes
Backup Battery	No

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

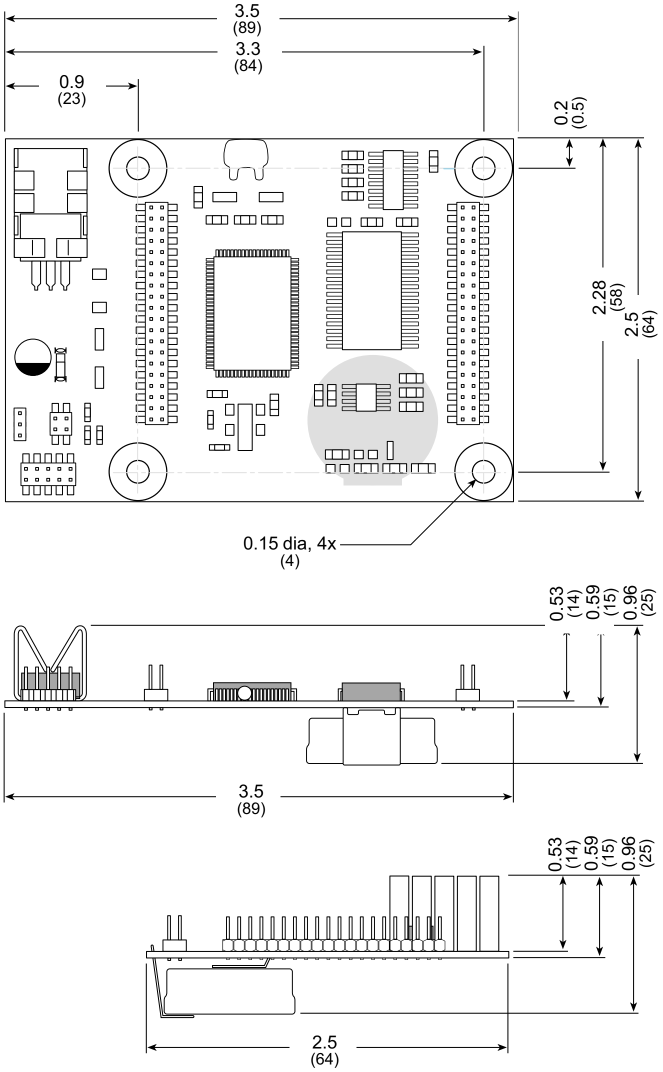


Figure A-1. Jackrabbit (BL1810) Dimensions

Table A-4 provides the pin 1 locations for the Jackrabbit headers.

Table A-4. Jackrabbit Header Pin 1 Locations

Header	Description	Pin 1 (x,y) Coordinates
J1	Power supply input	(0.110, 0.700)
J2	External battery	(0.415, 0.638)
J3	Programming port	(0.145, 0.149)
J4	Jackrabbit subsystems	(0.984, 2.023)
J5	Jackrabbit subsystems	(3.184, 2.023)

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 Mechanical Dimensions and Layout

Figure B-1 shows the mechanical dimensions and layout for the Jackrabbit Prototyping Board.

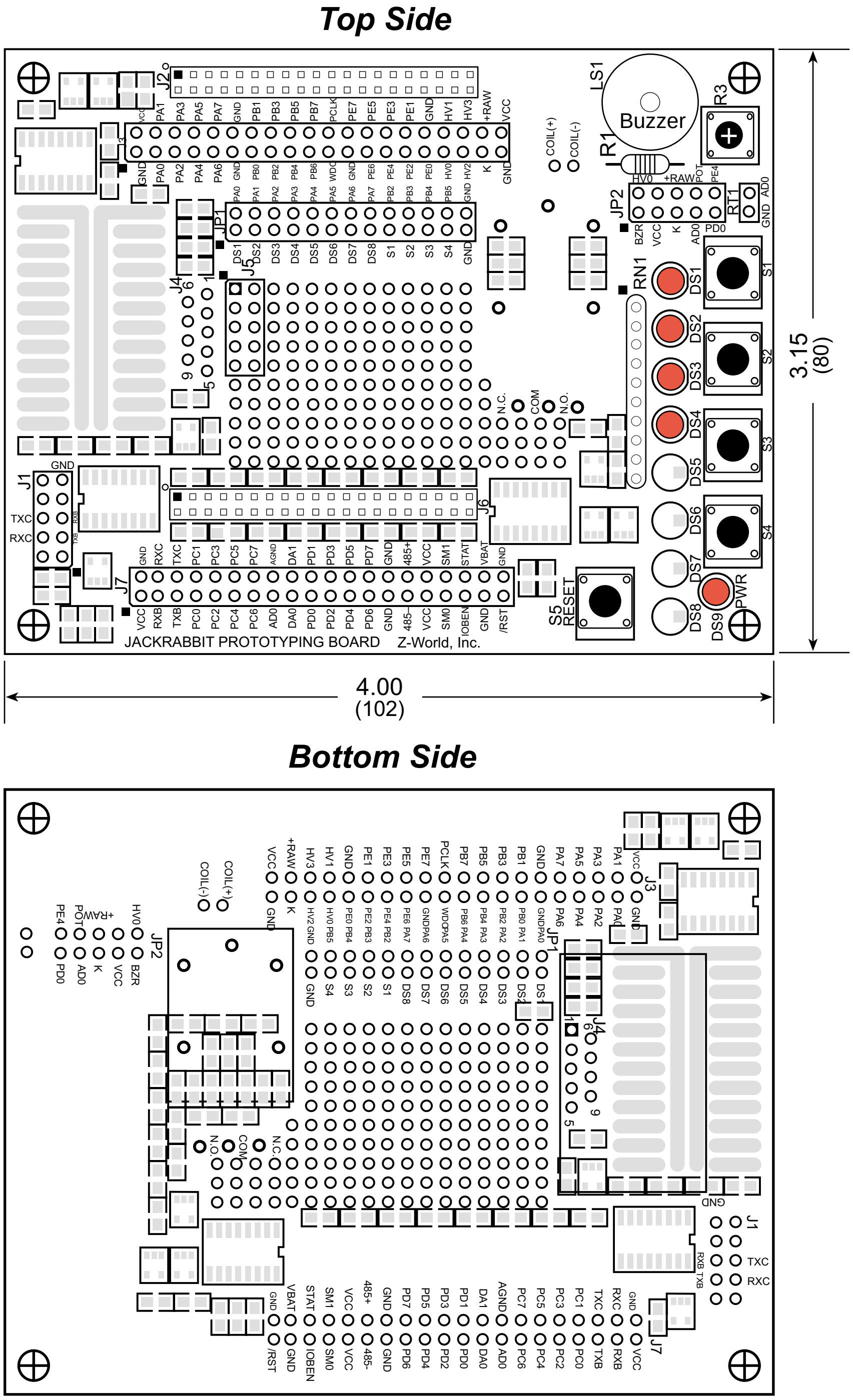


Figure B-1. Jackrabbit Prototyping Board

B.2 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 ***Using the Jackrabbit Tutorial***, 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-2 onto the Bottom Side.

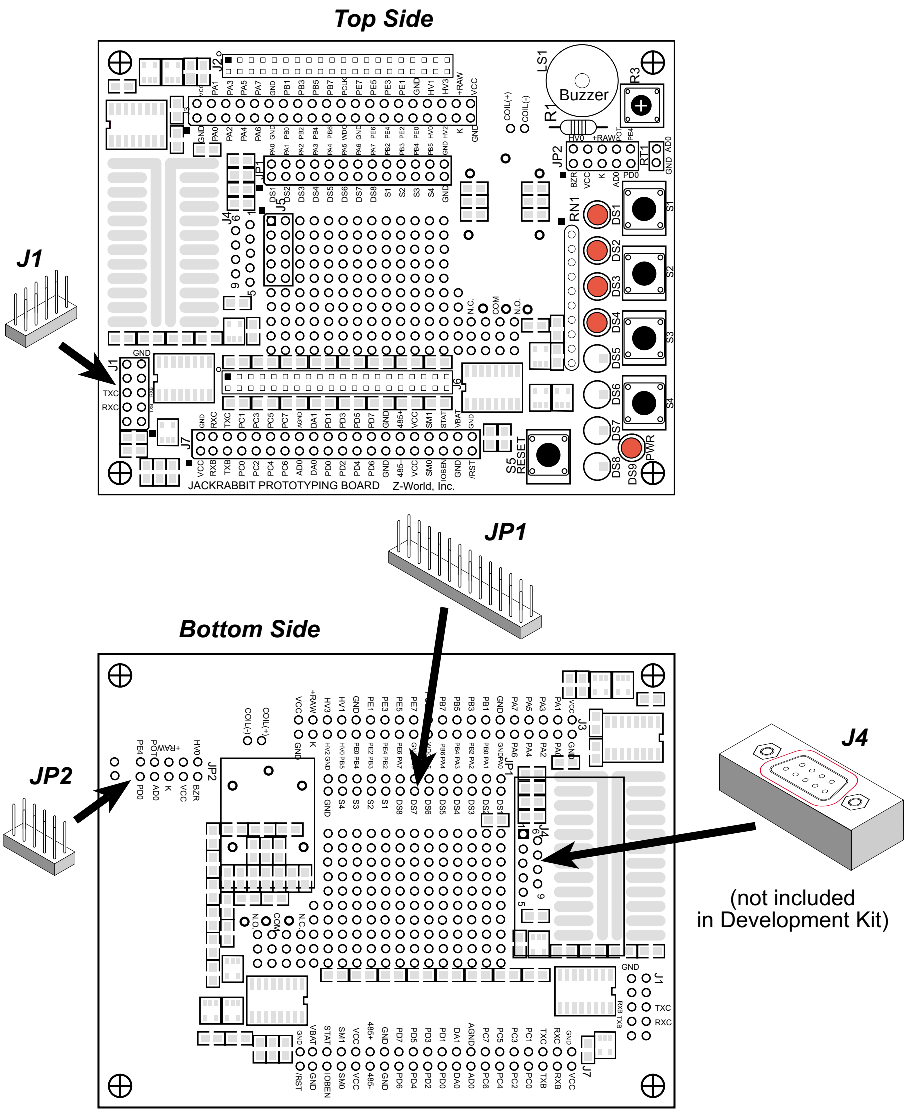
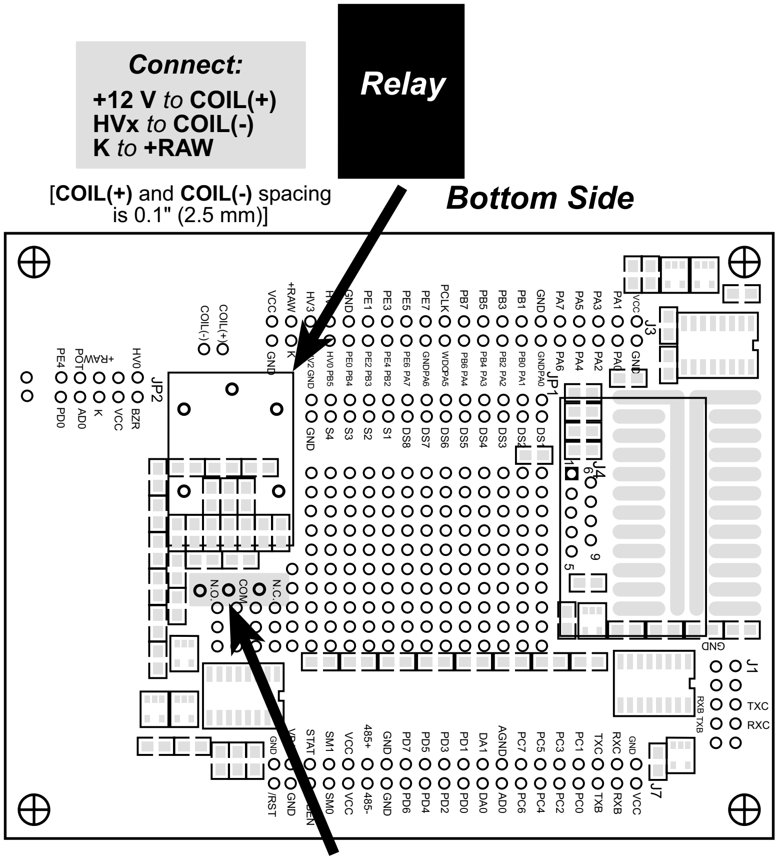


Figure B-2. Where to Solder Additional Headers

B.2.1 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 Bottom 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-3.



Connect:
+12 V *to* **COIL(+)**
HVx *to* **COIL(-)**
K *to* **+RAW**

[**COIL(+)** and **COIL(-)** spacing is 0.1" (2.5 mm)]

Relay Contacts
[0.156" (4 mm) spacing]

Figure B-3. Installation of Relay on Prototyping Board

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.

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-4) on the Prototyping Board, the A/D readings on AD0 will change with temperature.

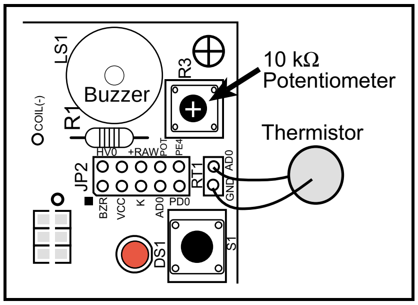


Figure B-4. Thermistor and Potentiometer Locations

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

The LEDs can be mounted in positions DS5–DS8, shown in Figure B-5, to display the complete status for Parallel Port A.

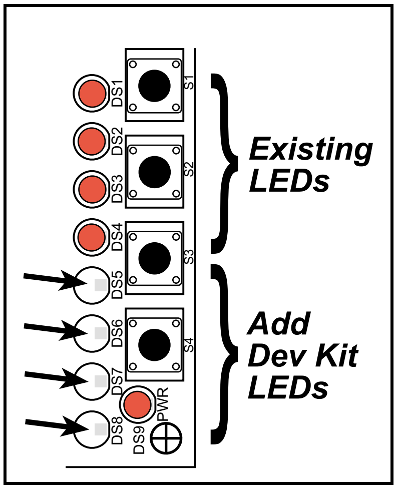


Figure B-5. LED Location

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 DE9 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.2.2 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.

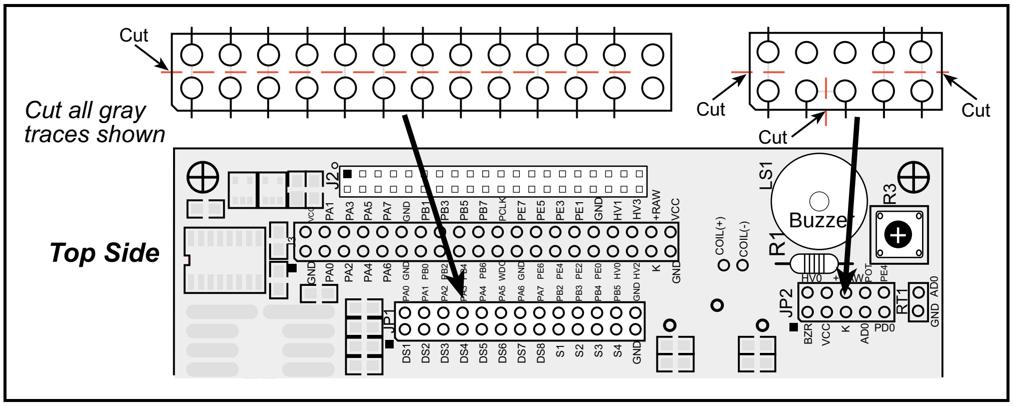

Cut
Cut all gray traces shown
Cut
Cut
Cut
Top Side
J2
VCC PA1 PA3 PA5 PA7 GND PB1 PB3 PB5 PB7 PCLK PE7 PE5 PE3 PE1 GND HV1 HV3 +RAW VCC
GND PA0 PA2 PA4 PA6 GND PB0 PB2 PB4 PB6 WDO GND PE6 PE4 PE2 PE0 HV0 HV2 K GND
PA0 PA1 PA2 PA3 PA4 PA5 PA6 PA7 PB2 PB3 PB4 PB5 GND
JP1
DS1 DS2 DS3 DS4 DS5 DS6 DS7 DS8 S1 S2 S3 S4 GND
COIL(+)
COIL(-)
LS1
Buzzer
R3
R1
HV0 +RAW POT PE4
JP2
BZR VCC K AD0 PD0
RT1
AD0 GND

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 Bottom Side of the Prototyping Board (shown in Figure B-7) are a mirror image of the Jackrabbit board pinouts.

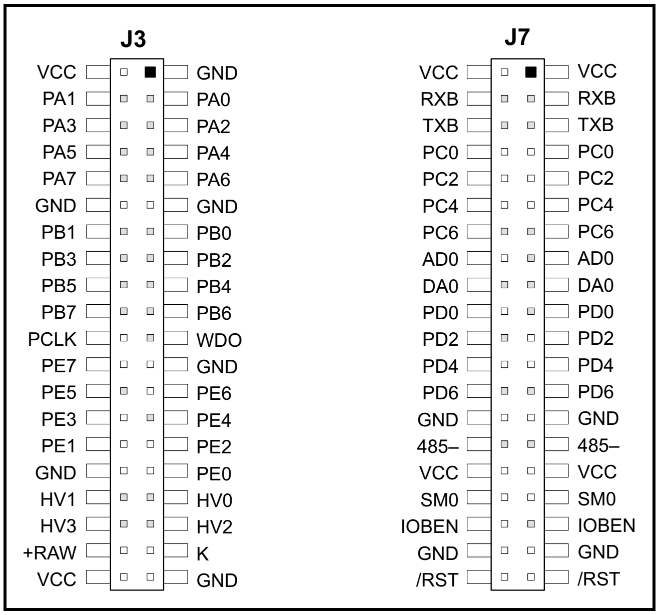


Figure B-7. Jackrabbit I/O Pinout on Prototyping Board (Bottom Side)

A user-supplied DE9 connector can be added as shown in Figure B-2. 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 Bottom 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.

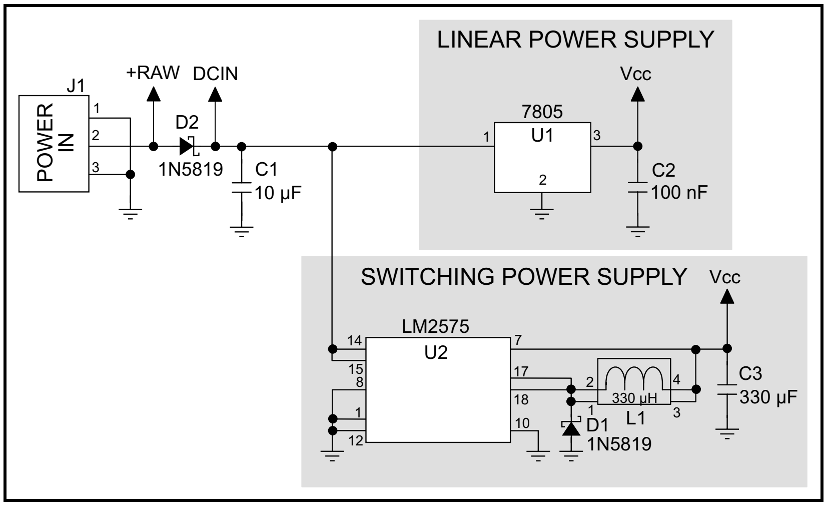


Figure C-1. Jackrabbit Power Supply Schematic

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.

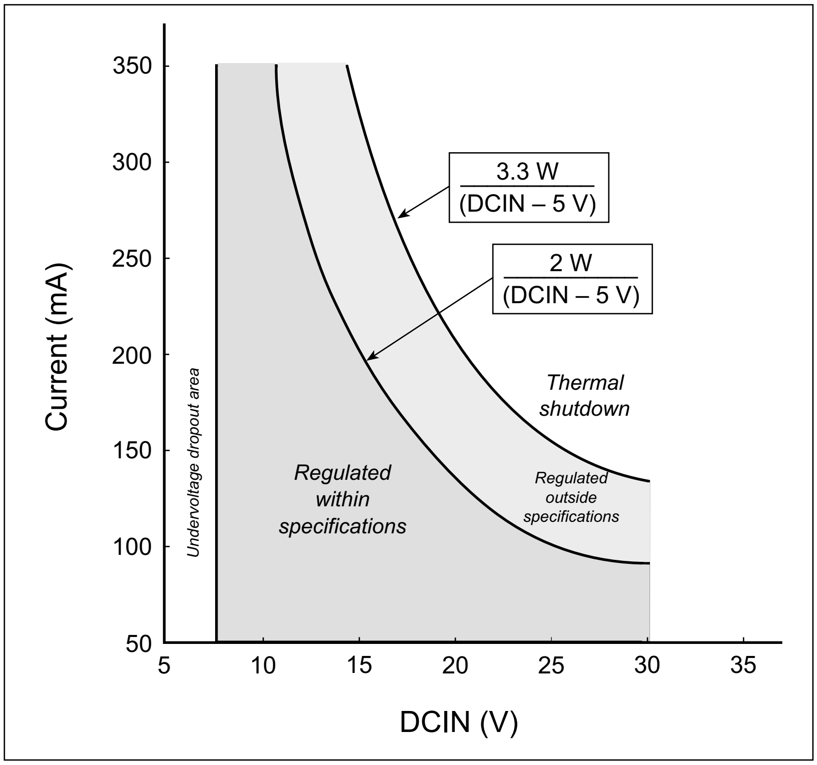


Figure C-2. 7805 Linear Regulator Power Operating Curve

The Jackrabbit operating at 14.7 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 14.7 MHz with the programming cable connected.

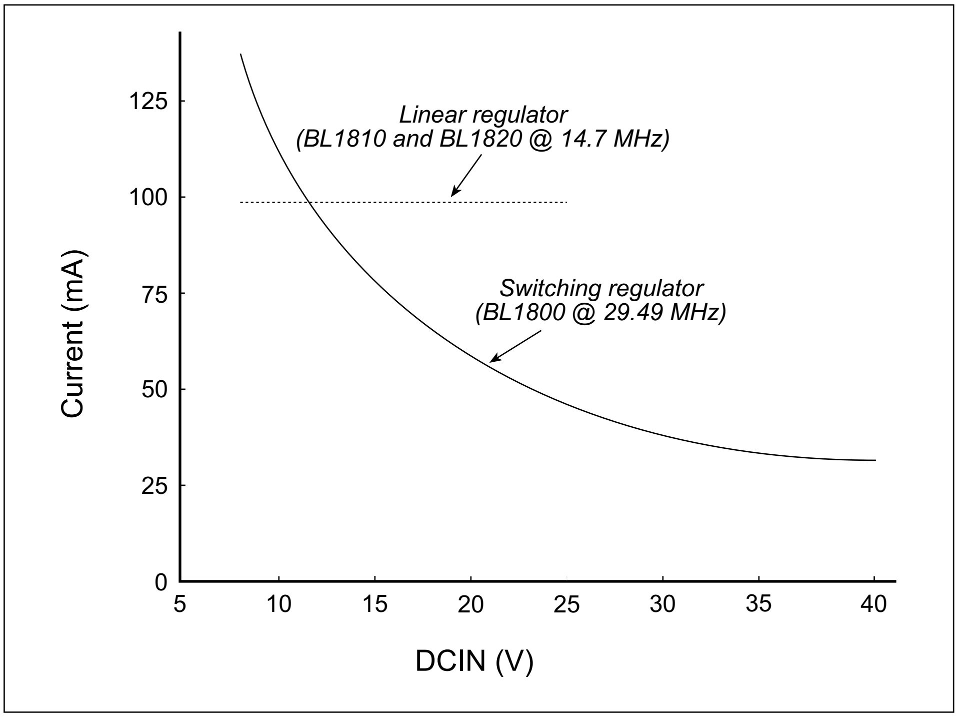


Figure C-3. Typical Jackrabbit Current Consumption

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:

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

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:

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

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.

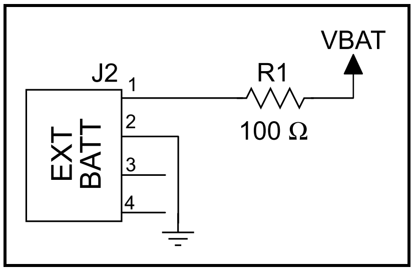


Figure C-4. External Battery Connections at Header J2

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.

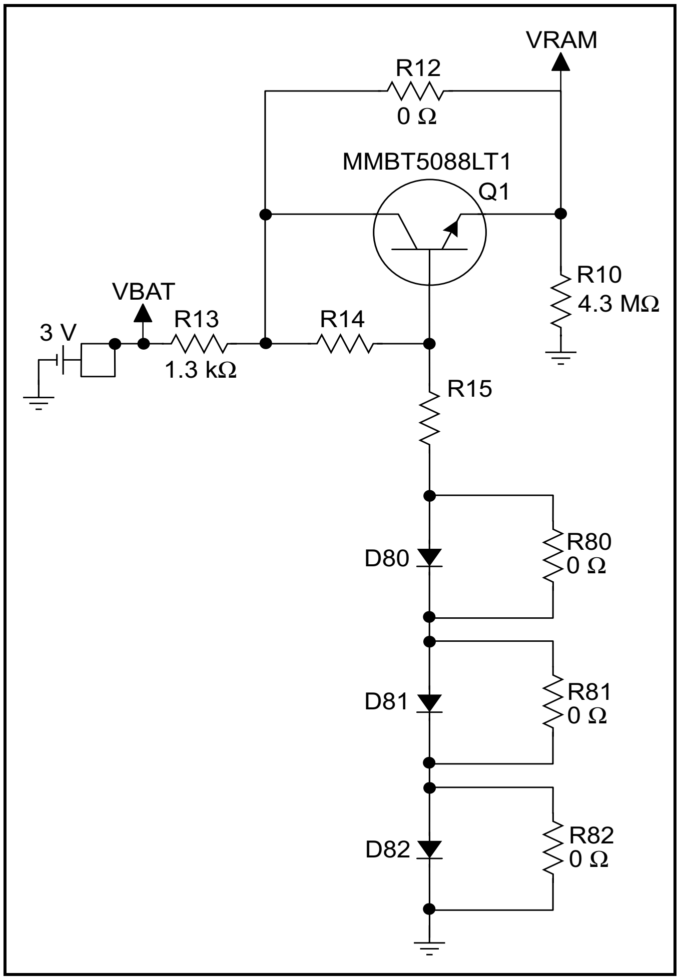


Figure C-5. Jackrabbit Battery Backup Circuit

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.

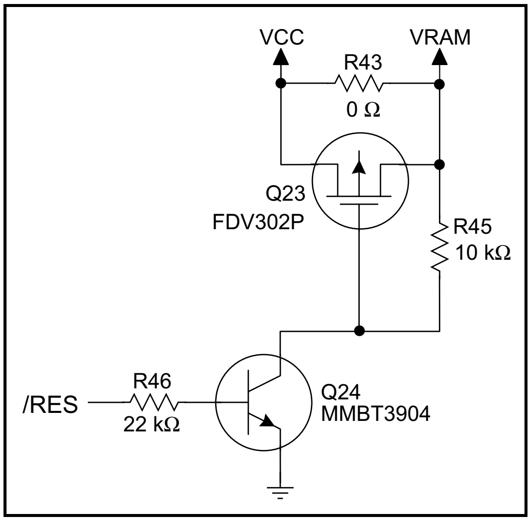


Figure C-6. VRAM Switch

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.

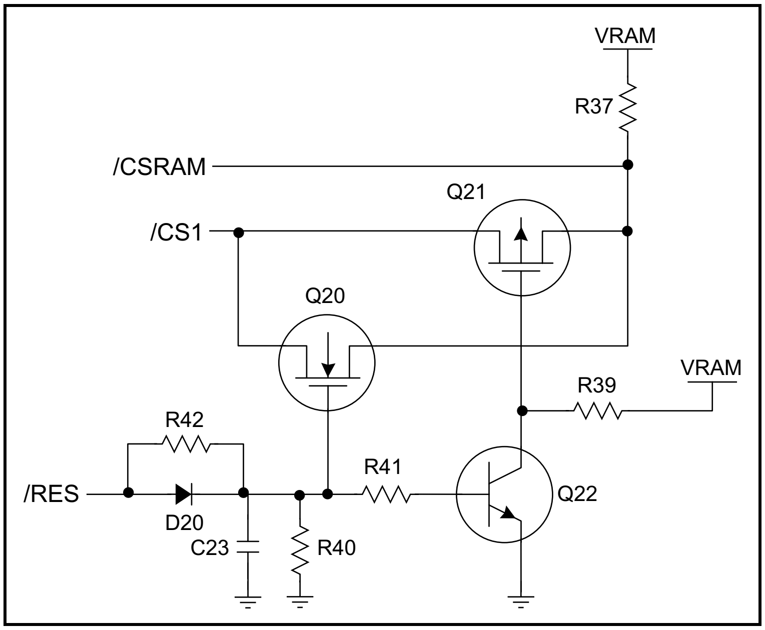


Figure C-7. Chip Select Circuit

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.

APPENDIX D. ALTERNATE USE OF THE PROGRAMMING PORT

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

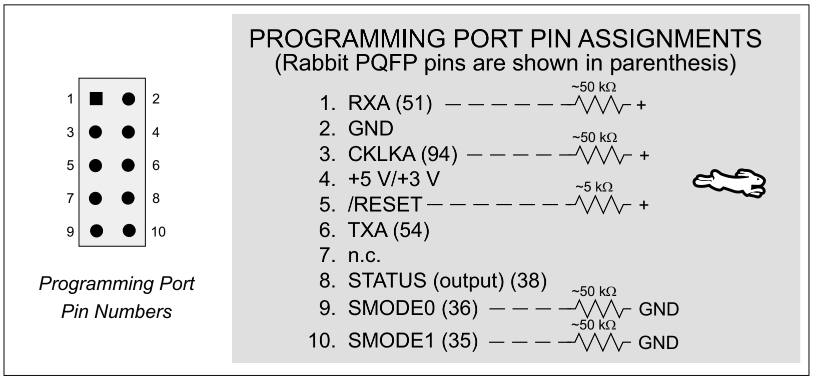


Figure D-1. Programming Port Pin Assignments

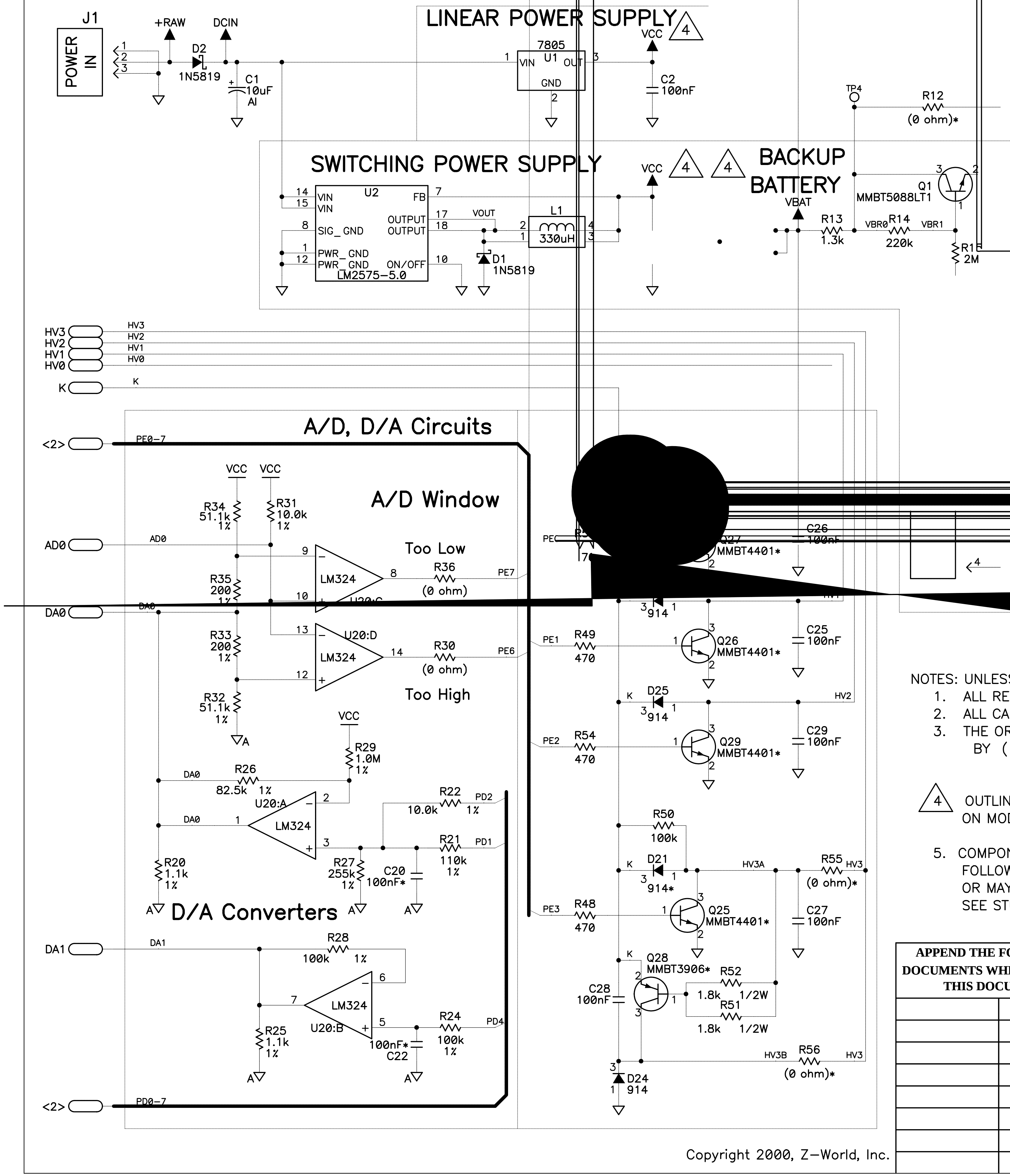
When the **PROG** connector is used, the /RESET line can be asserted by manipulating DTR and the STATUS line can be read as DSR on the serial port. The target can be restarted by pulsing reset and then, after a short delay, sending a special character string at 2400 bps. To simply restart the BIOS, the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

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

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

SCHEMATICS

REVISION HISTORY			REVISION APPROVAL			
REV	ECO	DESCRIPTION	PROJECT ENGINEER	APPROVAL DATE	DOCUMENT CONTROL	APPROVAL DATE



NOTES: UNLESS OTHERWISE SPECIFIED;

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

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

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

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

TITLE

Z-WORLD

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

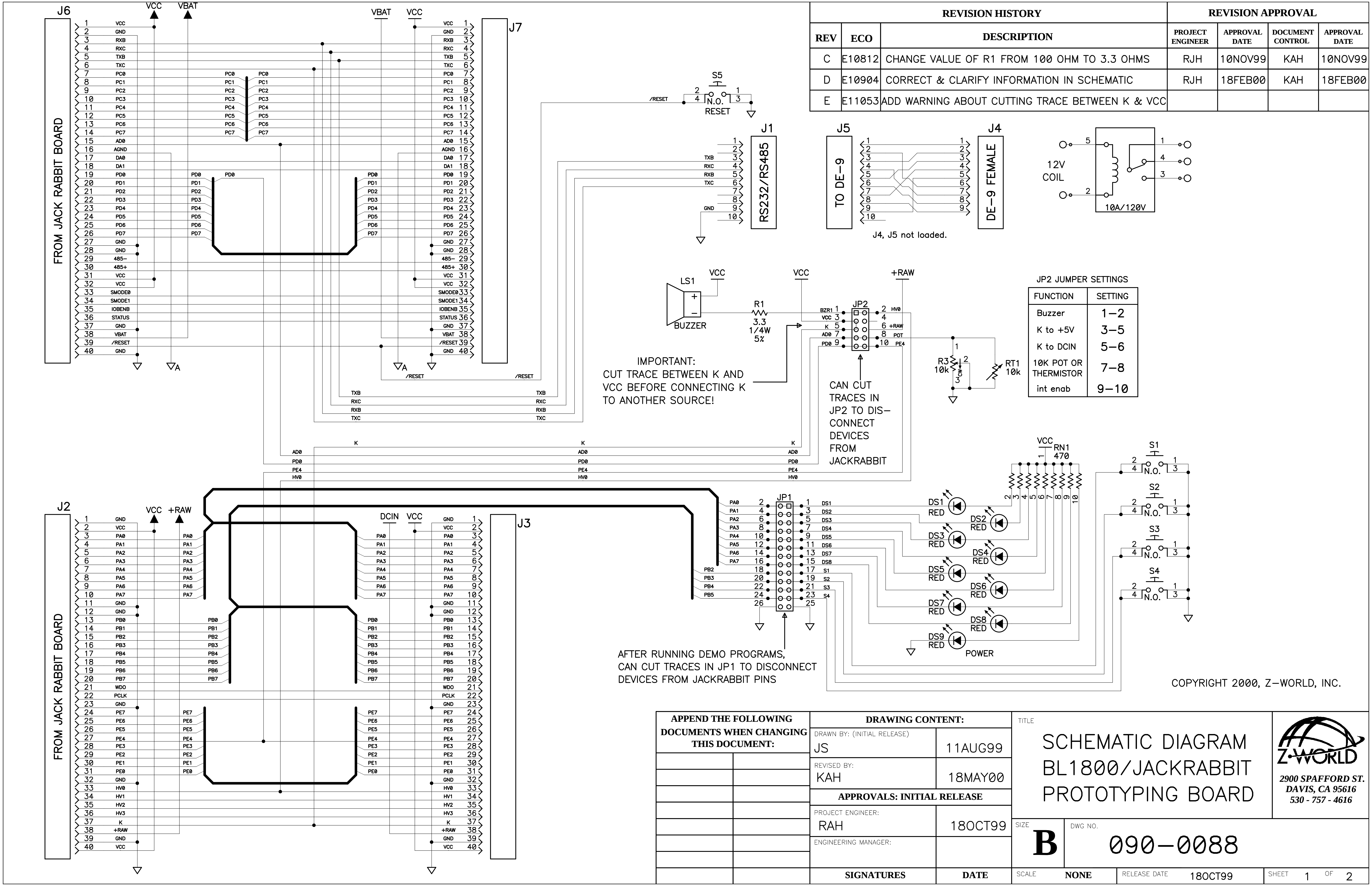
SIZE	DWG NO.		
B			
SCALE NONE	RELEASE DATE	SHEET	OF

	REGULATOR TEMPERATURE C				
CHIP SELECT CONTROL	W/BATTERY BACKUP	D20	914	914	NOT INSTALLED
		Q20	2N7002 n-ch	2N7002 n-ch	NOT INSTALLED
		Q21	FDV302P p-ch	FDV302P p-ch	NOT INSTALLED
		Q22	3904 npn	3904 npn	NOT INSTALLED
		R37	100k	100k	NOT INSTALLED
		R39	10k	10k	NOT INSTALLED
		R40	300k	300k	NOT INSTALLED
		R41	47k	47k	NOT INSTALLED
		R42	100k	100k	NOT INSTALLED
	W/O BATTERY BACKUP	R38	NOT INSTALLED	NOT INSTALLED	ZERO ohm
POWER TO VRAM SWITCH	W/BATTERY BACKUP	Q23	FDV302P p-ch	FDV302P p-ch	NOT INSTALLED
		Q24	3904 npn	3904 npn	NOT INSTALLED
		R45	10k	10k	NOT INSTALLED
		R46	22k	22k	NOT INSTALLED
	W/O BATTERY BACKUP	R43	NOT INSTALLED	NOT INSTALLED	ZERO ohm
POWER SUPPLY	LINEAR	C2	NOT INSTALLED	.1uF	.1uF
		U1	NOT INSTALLED	7805 LIN REG	7805 LIN REG
	SWITCHING	C3	330uF	NOT INSTALLED	NOT INSTALLED
		D1	1N5819	NOT INSTALLED	NOT INSTALLED
		L1	330uH	NOT INSTALLED	NOT INSTALLED
		U2	LM2575-5.0	NOT INSTALLED	NOT INSTALLED

D/A CONV		
HIGH POWER DRIVERS	MAIN	
	HV3=SINKING	
	HV3=SOURCING	
PROCESSOR RTC CRYSTAL		
PROCESSOR CRYSTAL		

SIZE B	DWG NO.		
SCALE NONE	REV LTR	SHEET	OF

REVISION HISTORY			REVISION APPROVAL			
REV	ECO	DESCRIPTION	PROJECT ENGINEER	APPROVAL DATE	DOCUMENT CONTROL	APPROVAL DATE
C	E10812	CHANGE VALUE OF R1 FROM 100 OHM TO 3.3 OHMS	RJH	10NOV99	KAH	10NOV99
D	E10904	CORRECT & CLARIFY INFORMATION IN SCHEMATIC	RJH	18FEB00	KAH	18FEB00
E	E11053	ADD WARNING ABOUT CUTTING TRACE BETWEEN K & VCC				



J4, J5 not loaded.

JP2 JUMPER SETTINGS

FUNCTION	SETTING
Buzzer	1–2
K to +5V	3–5
K to DCIN	5–6
10K POT OR THERMISTOR	7–8
int enab	9–10

IMPORTANT:
CUT TRACE BETWEEN K AND VCC BEFORE CONNECTING K TO ANOTHER SOURCE!

CAN CUT TRACES IN JP2 TO DISCONNECT DEVICES FROM JACKRABBIT

AFTER RUNNING DEMO PROGRAMS, CAN CUT TRACES IN JP1 TO DISCONNECT DEVICES FROM JACKRABBIT PINS

APPEND THE FOLLOWING DOCUMENTS WHEN CHANGING THIS DOCUMENT:		DRAWING CONTENT:	
		DRAWN BY: (INITIAL RELEASE) JS	11AUG99
		REVISED BY: KAH	18MAY00
		APPROVALS: INITIAL RELEASE	
		PROJECT ENGINEER: RAH	18OCT99
		ENGINEERING MANAGER:	
		SIGNATURES	DATE

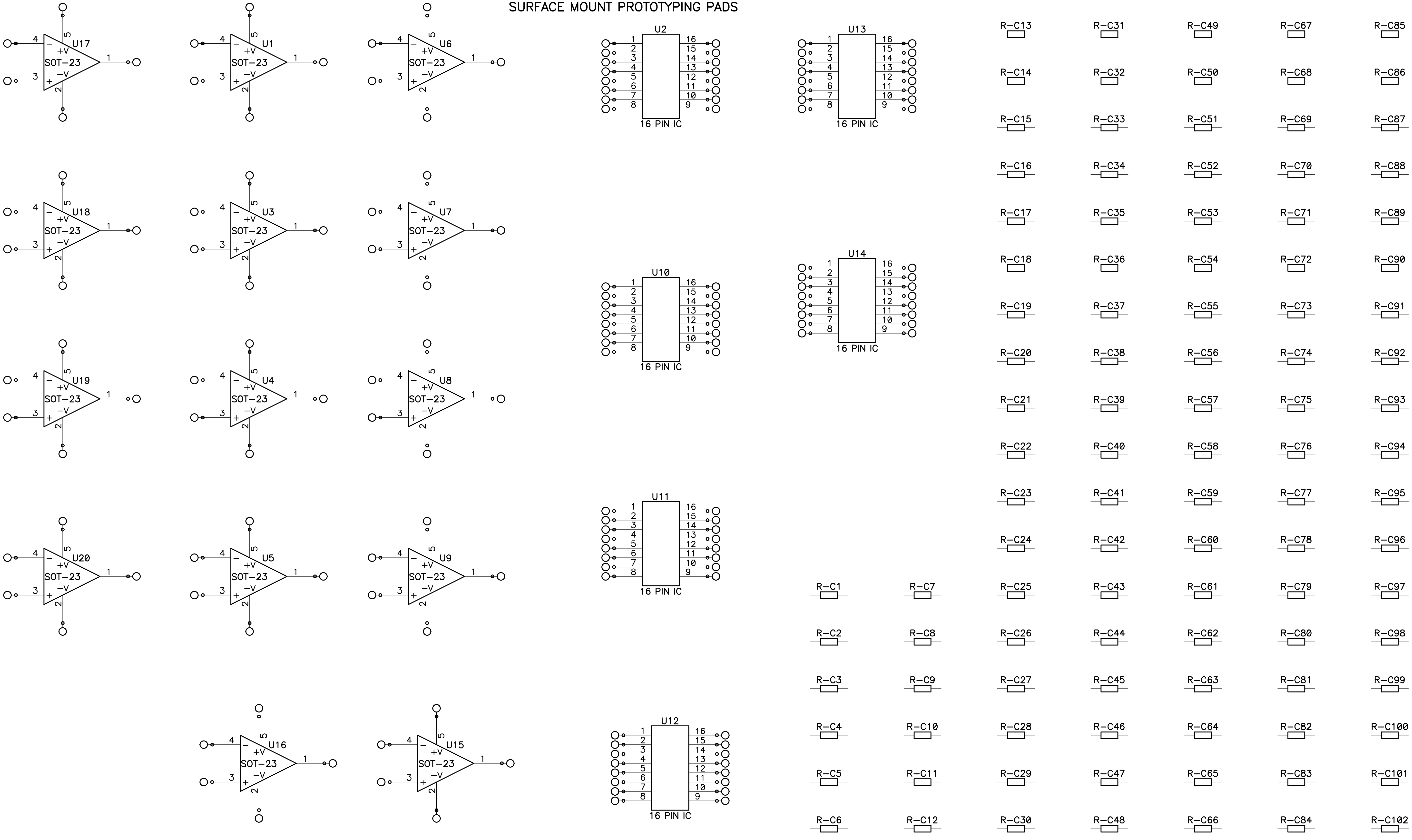
TITLE

SCHEMATIC DIAGRAM BL1800/JACKRABBIT PROTOTYPING BOARD

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

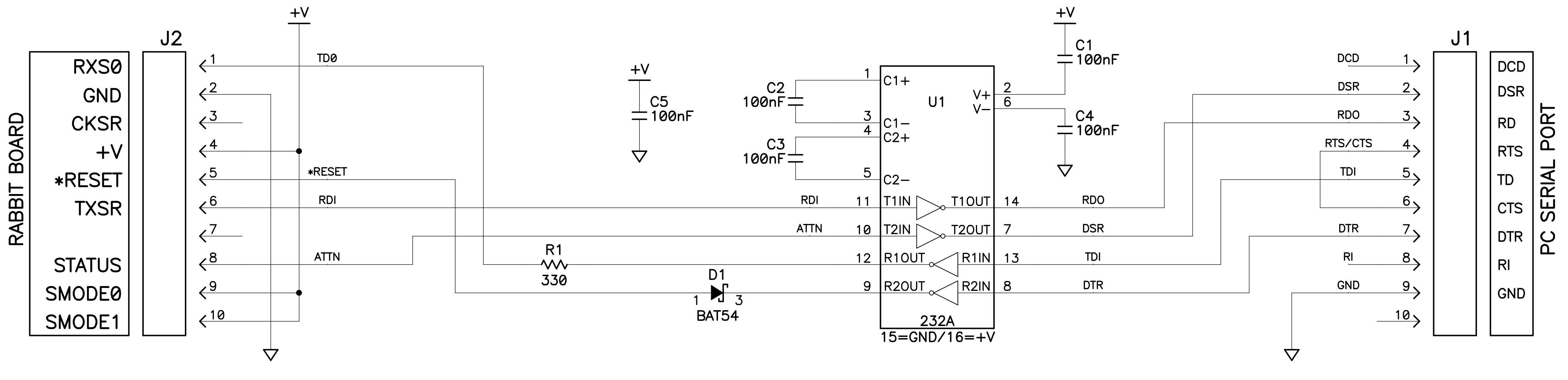
SIZE	DWG NO.		
B	090-0088		
SCALE NONE	RELEASE DATE 18OCT99	SHEET 1	OF 2

SURFACE MOUNT PROTOTYPING PADS



SIZE	DWG NO.		
B	090-0088		
SCALE NONE	REV LTR E	SHEET 2 OF 2	

REVISION HISTORY			REVISION APPROVAL			
REV	ECO	DESCRIPTION	PROJECT ENGINEER	APPROVAL DATE	DOCUMENT CONTROL	APPROVAL DATE
X1	----	Engineering Prototype Release A/W Rev-A	RH	----	----	----
A	E10680	INITIAL RELEASE OF SCHEMATIC, PCB A/W @ REV-B				



NOTES: UNLESS OTHERWISE SPECIFIED;

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

APPEND THE FOLLOWING DOCUMENTS WHEN CHANGING THIS DOCUMENT:

DRAWING CONTENT:	
DRAWN BY: (INITIAL RELEASE) KAH	15MAR99
REVISED BY: KAH	13AUG99
APPROVALS: INITIAL RELEASE	
PROJECT ENGINEER:	
ENGINEERING MANAGER:	
SIGNATURES	DATE

TITLE

SCHEMATIC DIAGRAM
RABBIT SIB

Z WORLD

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

SIZE **B** DWG NO. 090-0085

SCALE NONE RELEASE DATE SHEET 1 OF 1