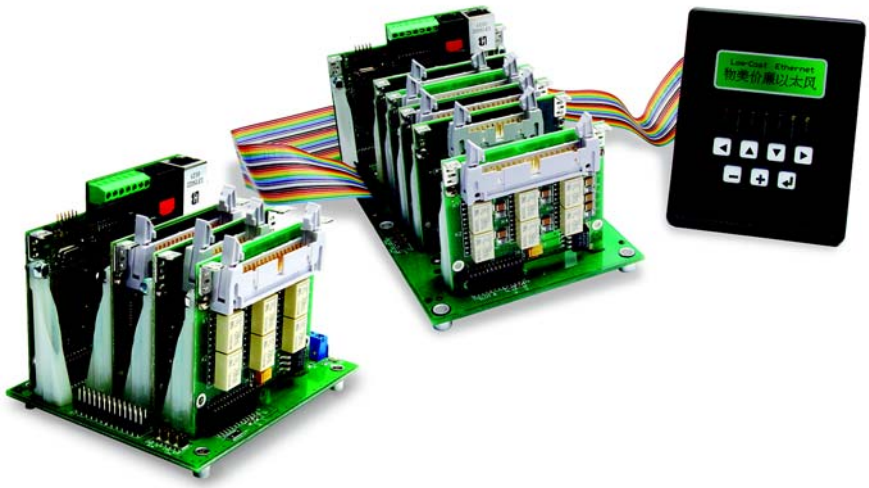
PRODUCT MANUAL



# Smart Star (SR9000)

Modular C-Programmable Control System

## User's Manual

019–0107 • 090519–L

# Smart Star (SR9000) User's Manual

Part Number 019-0107 • 090519–L • Printed in U.S.A.

© 2002–2009 Digi International Inc. • All rights reserved.

No part of the contents of this manual may be reproduced or transmitted in any form or by any means without the express written permission of Digi International.

Permission is granted to make one or more copies as long as the copyright page contained therein is included. These copies of the manuals may not be let or sold for any reason without the express written permission of Digi International.

Digi International reserves the right to make changes and improvements to its products without providing notice.

## Trademarks

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

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

**Digi International Inc.**

www.rabbit.com

# TABLE OF CONTENTS

## Part I. CPU/Backplane


**Chapter 1. Introduction** **9**

1.1 Features ....9
1.2 User Connections ....11
1.3 Optional Add-Ons ....11
1.4 Development and Evaluation Tools ....12
1.4.1 Tool Kit ....12
1.4.2 Software ....12
1.5 CE Compliance ....13
1.5.1 Design Guidelines ....15
1.5.2 Interfacing the Smart Star to Other Devices ....15

**Chapter 2. Getting Started** **17**

2.1 Attach the CPU Card to the Backplane ....18
2.2 Connect the Power Supply ....19
NOTE: Notice to Customers Outside North America ....19
2.3 Programming Cable Connections ....20
2.4 Installing Dynamic C ....21
2.5 Starting Dynamic C ....22
2.6 PONG.C ....23
2.7 Installing I/O Cards ....24
2.8 Where Do I Go From Here? ....25

**Chapter 3. Hardware Features** **27**

3.1 Backplane Features ....28
3.1.1 Power Distribution ....28
3.1.2 I/O Card Slots ....31
3.2 Smart Star CPU Card Features ....32
3.2.1 Serial Communication ....32
3.2.1.1 RS-232 ....32
3.2.1.2 RS-485 ....33
3.2.1.3 Programming Port ....35
3.2.1.4 Ethernet Port (SR9150 only) ....36
3.3 Programming Cable ....37
3.3.1 Changing Between Program Mode and Run Mode ....37
3.3.2 Memory ....38
3.3.2.1 SRAM ....38
3.3.2.2 Flash EPROM ....38
3.3.3 Other Connectors ....39
3.4 Other Hardware ....41
3.4.1 Clock Doubler ....41
3.4.2 Spectrum Spreader ....41

**Chapter 4. Software** **43**
4.1 Running Dynamic C ... 43
4.1.1 Upgrading Dynamic C ... 45
4.1.1.1 Patches and Bug Fixes ... 45
4.1.1.2 Upgrades ... 45
4.2 Sample Programs ... 46
4.3 Dynamic C Libraries ... 47
4.4 Smart Star Backplane Function Calls ... 48
4.4.1 Board Reset ... 48
4.4.2 Board Initialization ... 48
4.5 Serial Communication Calls ... 49

**Chapter 5. Using the TCP/IP Features** **51**
5.1 Ethernet Connections ... 51
5.2 TCP/IP Sample Programs ... 53
5.2.1 How to Set IP Addresses in the Sample Programs ... 53
5.2.2 How to Set Up Your Computer for Direct Connect ... 54
5.2.3 Run the PINGME.C Demo ... 55
5.2.4 Additional Demo Programs ... 55
5.2.5 LCD/Keypad Sample Programs Showing TCP/IP Features ... 56
5.3 Where Do I Go From Here? ... 57

**Chapter 6. Smart Star Specifications** **59**
6.1 Electrical and Mechanical Specifications ... 60
6.1.1 Smart Star Backplane ... 60
6.1.2 CPU Card ... 62
6.2 Jumper Configurations ... 64
6.3 Conformal Coating ... 66
6.4 Use of Rabbit 2000 Parallel Ports ... 67
6.5 Exclusion Zone ... 68

# Part II. Digital I/O Cards

**Chapter 7. Digital I/O Cards** **71**
7.1 Features ... 71
7.2 User Interface ... 72
7.3 User FWT Connections ... 73
7.3.1 Pinouts ... 73
7.4 Digital Inputs and Outputs ... 74
7.4.1 Digital Inputs ... 75
7.4.2 Digital Outputs ... 77
7.5 Software ... 79
7.5.1 Sample Programs ... 79
7.5.1.1 Running Sample Programs ... 79
7.5.2 Dynamic C Libraries ... 79
7.5.3 Smart Star Digital I/O Card Function Calls ... 80
7.6 Electrical and Mechanical Specifications ... 82

# Part III. A/D Converter Cards


**Chapter 8. A/D Converter Cards** **87**

8.1 A/D Converter Card Features ....87
8.2 User Interface ....88
8.3 User FWT Connections ....89
8.3.1 Pinouts ....89
8.4 Power Distribution ....90
8.5 Software ....91
8.5.1 Sample Programs ....91
8.5.1.1 Running Sample Programs .... 91
8.5.2 Dynamic C Libraries ....91
8.5.3 Smart Star A/D Converter Card Function Calls ....92
8.6 Electrical and Mechanical Specifications ....96


# Part IV. D/A Converter Cards


**Chapter 9. D/A Converter Cards** **101**

9.1 D/A Converter Card Features ....101
9.2 User Interface ....102
9.3 User FWT Connections ....104
9.3.1 Pinouts ....104
9.4 Power Distribution ....105
9.5 Software ....106
9.5.1 Sample Programs ....106
9.5.1.1 Running Sample Programs .... 106
9.5.2 Dynamic C Libraries ....106
9.5.3 Smart Star D/A Converter Card Function Calls ....107
9.6 Electrical and Mechanical Specifications ....113


# Part V. Relay Cards


**Chapter 10. Relay Cards** **117**

10.1 Relay Card Features ....117
10.2 User Interface ....118
10.3 User FWT Connections ....119
10.3.1 Pinouts ....119
10.4 Power Distribution ....120
10.5 Relay Cards Software ....121
10.5.1 Sample Programs ....121
10.5.2 Running Sample Programs ....121
10.5.3 Dynamic C Libraries ....121
10.5.4 Smart Star Relay Card Function Calls ....122
10.6 Electrical and Mechanical Specifications ....123

# Part VI. Appendices


**Appendix A. Field Wiring Terminals** **127**
A.1 Selecting and Installing a Field Wiring Terminal ... 128
A.2 Dimensions ... 129

**Appendix B. LCD/Keypad Module** **131**
B.1 Specifications ... 131
B.2 Contrast Adjustments for All Boards ... 133
B.3 Keypad Labeling ... 134
B.4 Header Pinouts ... 135
B.4.1 I/O Address Assignments ... 135
B.5 Mounting LCD/Keypad Module ... 136
B.5.1 Installation Guidelines ... 136
B.5.2 Mounting Instructions ... 137
B.5.2.1 Bezel-Mount Installation ... 137
B.6 Connecting LCD/Keypad Module to Smart Star Backplane ... 139
B.7 Sample Programs ... 141
B.8 LCD/Keypad Module Function Calls ... 143
B.8.1 LEDs ... 143
B.8.2 LCD Display ... 144
B.8.3 Keypad ... 160
B.9 Font and Bitmap Converter ... 163

**Appendix C. Power Management** **165**
C.1 Current Requirements ... 166
C.2 Batteries and External Battery Connections ... 166
C.2.1 Replacing the Backup Battery ... 167
C.2.2 Battery-Backup Circuit ... 167
C.2.3 Power to VRAM Switch ... 168
C.2.4 Reset Generator ... 169
C.2.5 External Battery ... 170
C.3 Chip Select Circuit ... 171

**Appendix D. Smart Star Slot Address Layout** **173**
D.1 Digital I/O Card Channel Layout ... 175
D.2 A/D Converter Card Channel Layout ... 176
D.3 D/A Converter Card Channel Layout ... 177
D.4 Relay Card Channel Layout ... 178

**Index** **179**

**Schematics** **183**

# PART I. CPU/BACKPLANE

# 1. INTRODUCTION

Chapter 1 introduces the Smart Star embedded control system and describes the features associated with the backplane chassis and the CPU Card. The Tool Kit containing the hardware essentials to begin using the Smart Star is described, and the software highlights are presented.

The Smart Star is a modular and expandable embedded control system whose configuration of Digital I/O, A/D Converter, D/A Converter, and Relay Cards can be tailored to a large variety of demanding real-time control and data acquisition applications.

The typical Smart Star system consists of a rugged backplane with a built-in voltage regulator, a CPU Card, and one or more I/O cards. The CPU Card plugs into a designated slot on the backplane chassis, which has additional slots available for any combination of I/O cards. A high-performance Rabbit 2000 microprocessor on the CPU Card operates at 22.1 MHz to provide fast data processing.

## 1.1 Features

- C-programmable to create a custom user interface
- Flexible functionality—modular configuration allows interchanging or replacing individual I/O cards
- Expandable—up to 168 I/O ports
- Choice of two backplanes—with either 3 or 7 slots for I/O cards
- Choice of CPU cards—with or without one RJ-45 10/100-compatible Ethernet port with 10Base-T Ethernet interface
- RS-232 and RS-485 serial ports allow networking to other Smart Star units, single-board computers, or enterprise computing centers
- 128K SRAM and 512K flash memory, optional 512K SRAM
- Real-time clock
- Watchdog supervisor
- Backup battery
- Optional backlit 122 × 32 graphic display/keypad module
- RabbitLink Ethernet gateway available for remote download/debug, Web serving, and e-mail

Table 1 lists the backplanes, CPU cards, and the I/O cards that are available for the Smart Star control system. Appendix A provides detailed specifications for the Smart Star backplanes and the CPU cards.

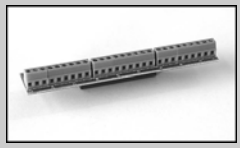
***Table 1. Smart Star Backplanes and Cards***

| Card | | Model | Features |
|---|---|---|---|
| Backplane | | SR9010 | 7 I/O card slots, 1 CPU card slot, header connections for optional LCD/keypad module |
| | | SR9050 | 3 I/O card slots, 1 CPU card slot, header connections for optional LCD/keypad module |
| CPU | | SR9150 | Full-featured CPU card ***with*** RJ-45 Ethernet port |
| | | SR9160 | Full-featured CPU card ***without*** RJ-45 Ethernet port |
| I/O Cards | Digital I/O | SR9200 | 16 digital inputs, 8 digital sinking outputs |
| | | SR9210 | 8 digital inputs, 16 digital sinking outputs |
| | | SR9220 | 8 digital inputs, 8 digital sinking outputs |
| | | SR9205 | 16 digital inputs, 8 digital sourcing outputs |
| | | SR9215 | 8 digital inputs, 16 digital sourcing outputs |
| | | SR9225 | 8 digital inputs, 8 digital sourcing outputs |
| | A/D Converter | SR9300 | 12-bit A/D converter, 11 channels, 0 V – 10 V |
| | | SR9310 | 12-bit A/D converter, 11 channels, -10 V – +10 V |
| | | SR9320 | 12-bit A/D converter, 11 channels, 4 mA – 20 mA |
| | D/A Converter | SR9400 | 12-bit D/A converter, 8 channels, 0 V – 10 V |
| | | SR9410 | 12-bit D/A converter, 8 channels, -10 V – +10 V |
| | | SR9420 | 12-bit D/A converter, 8 channels, 4 mA – 20 mA |
| | Relay | SR9500 | 5 SPST relays and 1 SPDT relay, each protected with onboard snubbers |
| | | SR9510 | 8 SPDT relays (no snubbers) |

## 1.2 User Connections

Connections to the I/O cards are made via a ribbon cable connector or optional field wiring terminals that are either pluggable or have screw terminals. Three different Field Wiring Terminals (FWTs) are available. Table 2 lists the I/O cards and the Rabbit part numbers for the corresponding FWTs.

***Table 2. Guide to FWT Selection***

| FWT Description | I/O Cards | Rabbit Part Number | |
|---|---|---|---|
| | | Pluggable Terminals | Screw Terminals |
| | | |  |
| FWT27 | Digital I/O Relay (SR9510) | 101-0420 | 101-0514 |
| FWT18 | A/D Converter D/A Converter | 101-0421 | 101-0515 |
| FWT18R | Relay (SR9500) | 101-0422 | 101-0516 |

**NOTE:** Appendix A, “Field Wiring Terminals,” provides further information on FWTs, including their dimensions.

## 1.3 Optional Add-Ons

The LCD/keypad module is the only available optional add-on. Further details on the LCD/keypad module are provided in Appendix B.

Visit our Web site for up-to-date information about additional add-ons and features as they become available. The Web site also has the latest revision of this user’s manual.

## 1.4 Development and Evaluation Tools

### 1.4.1 Tool Kit

The Tool Kit has the hardware essentials that you need to create and use your own Smart Star control system.

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

- ***Smart Star (SR9000) Getting Started*** instructions.
- ***Dynamic C*** CD-ROM, with complete product documentation on disk.
- Programming cable, used to connect your PC serial port to the Smart Star CPU Card to write and debug C programs that run on the Smart Star control system.
- FWT27 pluggable field wiring terminal.
- Screwdriver.
- DC power supply, used to power the backplane, which in turn supplies power to the CPU card and the I/O cards. The DC power supply accepts an AC input of 100 V to 240 V at up to 0.6 A, and delivers a DC output up to 1.1 A at 24 V.
- ***Rabbit 2000 Processor Easy Reference*** poster.
- Registration card.

### 1.4.2 Software

The Smart Star control system is programmed using Rabbit's Dynamic C. A compatible version is included on the Tool Kit CD-ROM.

Rabbit also offers add-on Dynamic C modules containing the popular µC/OS-II real-time operating system, as well as PPP, Advanced Encryption Standard (AES), and other select libraries. In addition to the Web-based technical support included at no extra charge, a one-year telephone-based technical support module is also available for purchase. Visit our Web site at www.rabbit.com or contact your Rabbit sales representative or authorized distributor for further information.

## 1.5 CE Compliance

Equipment is generally divided into two classes.

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

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

The CPU card, I/O cards, and backplane in the Smart Star embedded control system have been tested and were found to be in conformity with the following applicable immunity and emission standards as described in Table 3.

***Table 3. CE Compliance of Smart Star Backplanes and Cards***

| Card | | Model | Description | | Used for CE Compliance Testing |
|---|---|---|---|---|---|
| Backplane | | SR9010 | 7 I/O card slots, 1 CPU card slot, header connections for optional LCD/keypad module* | Full-featured | × |
| | | SR9050 | 3 I/O card slots, 1 CPU card slot, header connections for optional LCD/keypad module* | Sub-version | |
| CPU | | SR9150 | 22.1 MHz CPU card ***with*** Ethernet | Full-featured | × |
| | | SR9160 | 22.1 MHz CPU card ***without*** Ethernet | Sub-version | |
| I/O Cards | Digital I/O | SR9200 | 16 inputs, 8 sinking outputs | Full-featured | × |
| | | SR9205 | 16 inputs, 8 sourcing outputs | Full-featured | × |
| | | SR9210 | 8 inputs, 16 sinking outputs | Sub-version | |
| | | SR9220 | 8 inputs, 8 sinking outputs | Sub-version | |
| | A/D Converter | SR9300 | Eleven 12-bit analog inputs (0–10 V) | Sub-version | |
| | | SR9310 | Eleven 12-bit analog inputs (±10 V) | Full-featured | × |
| | | SR9320 | Eleven 12-bit analog inputs (4–20 mA) | Sub-version | |
| | Relay | SR9500 | 5 SPST relays and 1 SPDT relay, each protected with onboard snubbers | Full-featured | × |
| | | SR9510 | 8 SPDT relays (no snubbers) | Full-featured | × |

* No CE compliance testing was done with the LCD/keypad module connected to a Smart Star embedded control system. A system consisting of Smart Star boards and an LCD/keypad module therefore ***cannot*** be considered to be CE-compliant.

The sub-versions of the boards are also CE-compliant. All boards that are CE-compliant have the CE mark.

Several Smart Star boards are not yet CE-compliant. These boards are listed in Table 4.

***Table 4. Smart Star Backplanes and Cards Not CE-Compliant***

| Card | Model | Description | Comments |
|---|---|---|---|
| Backplane | SR9000 | 7 I/O card slots, 1 CPU card slot | Legacy product |
| CPU | SR9100 | 25.8 MHz CPU card | Legacy product |
| D/A Converter | SR9400 | Eight analog outputs (0–10 V) | Passed emissions tests, immunity tests pending |
| | SR9410 | Eight analog outputs (±10 V) | |
| | SR9420 | Eight analog outputs (4–20 mA) | |

## Immunity

The CE-compliant Smart Star boards meet the following EN55024/1998 immunity standards.

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

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

## Emissions

The CE-compliant Smart Star boards meet the following emission standards when used with a Smart Star embedded control system that contains a Rev. C or higher version of the Rabbit 2000 microprocessor with its spectrum spreader turned on and set to the normal mode. This microprocessor is used in all Smart Star CPU boards that carry the CE mark.

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

**NOTE:** The Smart Star embedded control system satisfied the Class A limits but not the Class B limits. Such equipment need not be restricted in its sale, but the following warning must be included in the instructions for its use.

**Warning**

This is a class A product. In a domestic environment this product may cause radio interference, in which case the user may be required to take adequate measures.

Additional shielding or filtering may be needed to meet Class B emissions standards.

### 1.5.1 Design Guidelines

Note the following requirements for incorporating a Smart Star embedded control system into your application to comply with CE requirements.

#### General

- The power supply provided with the Tool Kit is for development purposes only. It is the customer's responsibility to provide a CE-compliant power supply for the end-product application.
- When connecting the Smart Star embedded control system to outdoor cables, the customer is responsible for providing CE-approved surge/lighting protection.
- Rabbit recommends placing digital I/O or analog cables that are 3 m or longer in a metal conduit to assist in maintaining CE compliance and to conform to good cable design practices.
- When installing or servicing the Smart Star embedded control system, it is the responsibility of the end-user to use proper ESD precautions to prevent ESD damage to the Smart Star.

#### Safety

- All inputs and outputs to and from the Smart Star embedded control system must not be connected to voltages exceeding SELV levels (42.4 V AC peak, or 60 V DC).
- The lithium backup battery circuit on the CPU card in the Smart Star embedded control system has been designed to protect the battery from hazardous conditions such as reverse charging and excessive current flows. Do not disable the safety features of the design.

### 1.5.2 Interfacing the Smart Star to Other Devices

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

# 2. GETTING STARTED

Chapter 2 explains how to connect the power supply to the Smart Star backplane, how to install the CPU Card on the backplane, and how to connect the programming cable to the CPU Card. Once you run a sample program to demonstrate that you have connected everything correctly, you will be ready to go on to install I/O cards and finish developing your system.

## 2.1 Attach the CPU Card to the Backplane

1. Orient the backplane with the **PROCESSOR SLOT** facing away from you as shown in Figure 1.

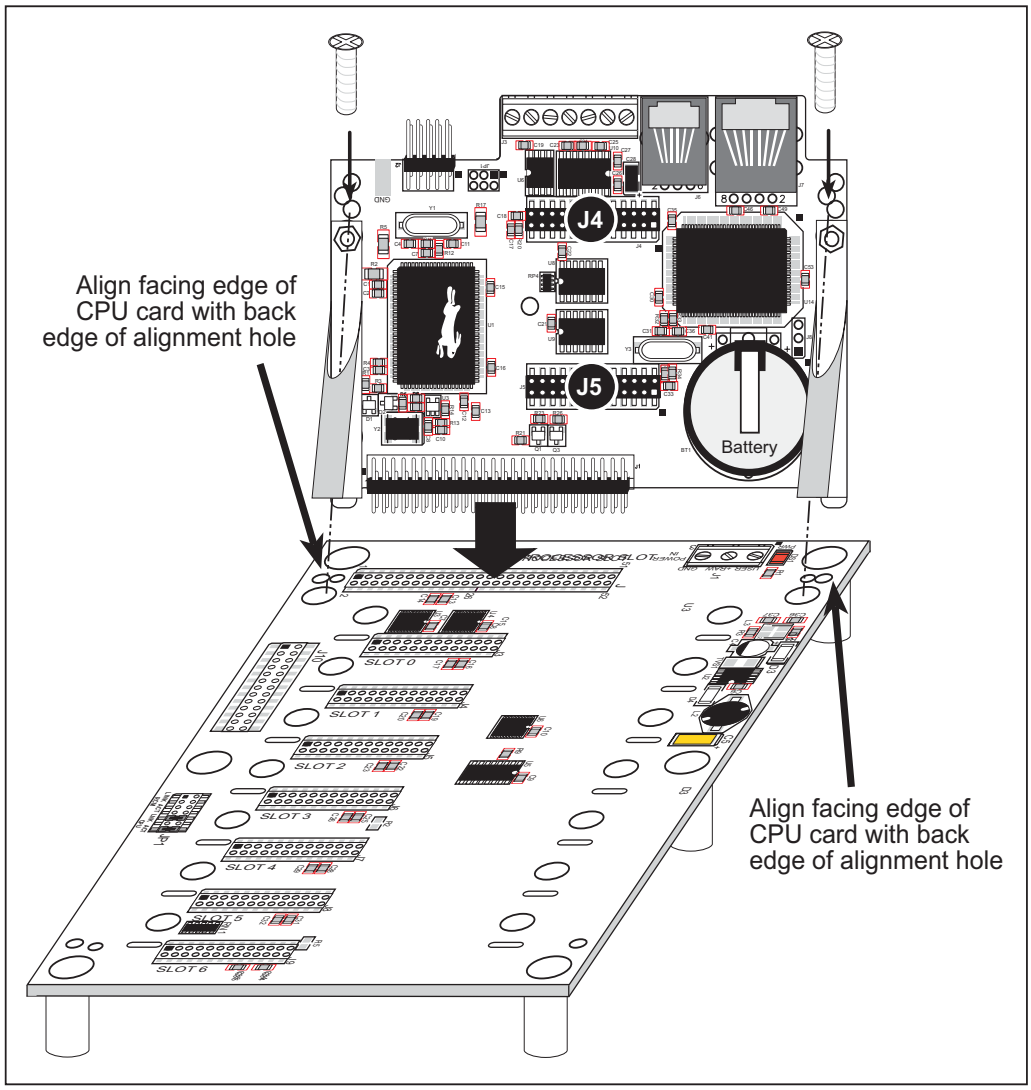



***Figure 1. Attach the CPU Card to the Backplane***

2. Position the CPU Card above the backplane as shown in Figure 1.
3. Carefully insert the CPU Card header into the **PROCESSOR SLOT** on the backplane and line up the facing edge of the CPU Card with the back edge of the alignment holes on the backplane as shown in Figure 1.

   **NOTE:** Be careful to line up the pins on the CPU Card with the socket on the backplane when installing the CPU Card. The CPU Card can be damaged once power is applied if the CPU Card is not installed correctly.

4. Use the two 4-40 screws supplied with the CPU Card to anchor the plastic brackets so that they hold the CPU Card firmly in place on the backplane.

## 2.2 Connect the Power Supply

Connect the power supply to the **POWER IN** connector on the backplane—the red (positive) wire to **+RAW** and the black (negative) wire to **GND**, as shown in Figure 2.

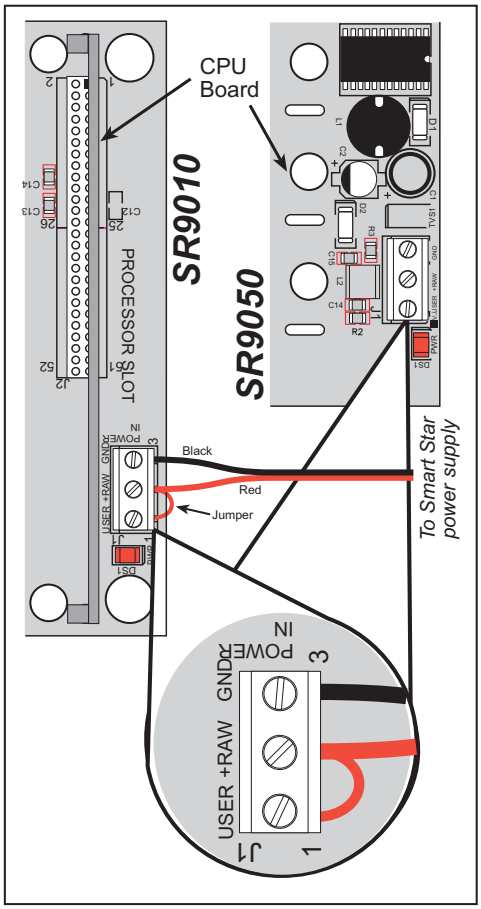



***Figure 2. Power Supply Connections (North America)***

> **NOTE:** Be careful to hook up the positive and negative leads ***exactly*** as described. Only the +5 V circuitry is protected against reverse polarity.

A **USER** connection is supplied on the backplane to allow an independent power supply to be used for future development. For now, use a wire jumper to connect **USER** to **+RAW** so that they share the same power supply.

### Notice to Customers Outside North America

The power supply included with the Smart Star Tool Kit may be used worldwide. Customers outside North America simply need to exchange the line cord and plug from the power supply to their wall outlet with one available locally.

1. To exchange the line cord and plug, first remove the existing line cord. To access the screws, use a screwdriver to gently lift up and remove the plastic insulating cover.

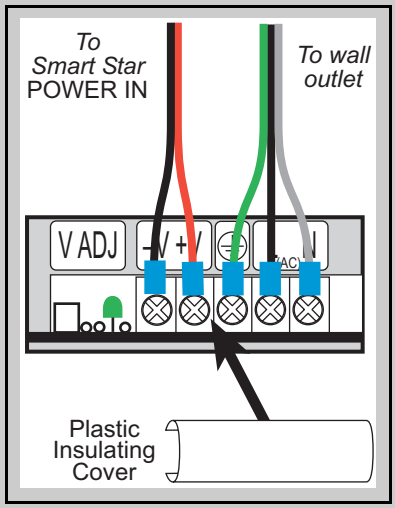



***Figure 3. Power Supply Connections (overseas)***

2. Unscrew the wires at the ground, **L**, and **N** terminals.
3. Attach the line cord that you obtained locally to the power supply. Be sure to follow any color-coding conventions, for example, green/yellow to ground, brown to **L**, and blue to **N** terminals.
4. Ensure that the wires are attached securely and are not touching each other. Snap on the plastic insulating cover.

> **NOTE:** The power supply included with the Smart Star Tool Kit is intended for development purposes only.

## 2.3 Programming Cable Connections

### 1. Connect the programming cable to the CPU Card.

Connect the 10-pin **PROG** connector of the programming cable to header J2 on the CPU Card as shown in Figure 4. Connect the other end of the programming cable to a COM port on your PC. Note that COM1 on the PC is the default COM port in the Dynamic C installation.

**NOTE:** Be sure to use the programming cable (Part No. 101-0513) supplied with the Smart Star Tool Kit—the programming cable has red shrink wrap around the RS-232 converter section located in the middle of the cable. Programming cables from other Rabbit kits are not designed to work with the Smart Star.

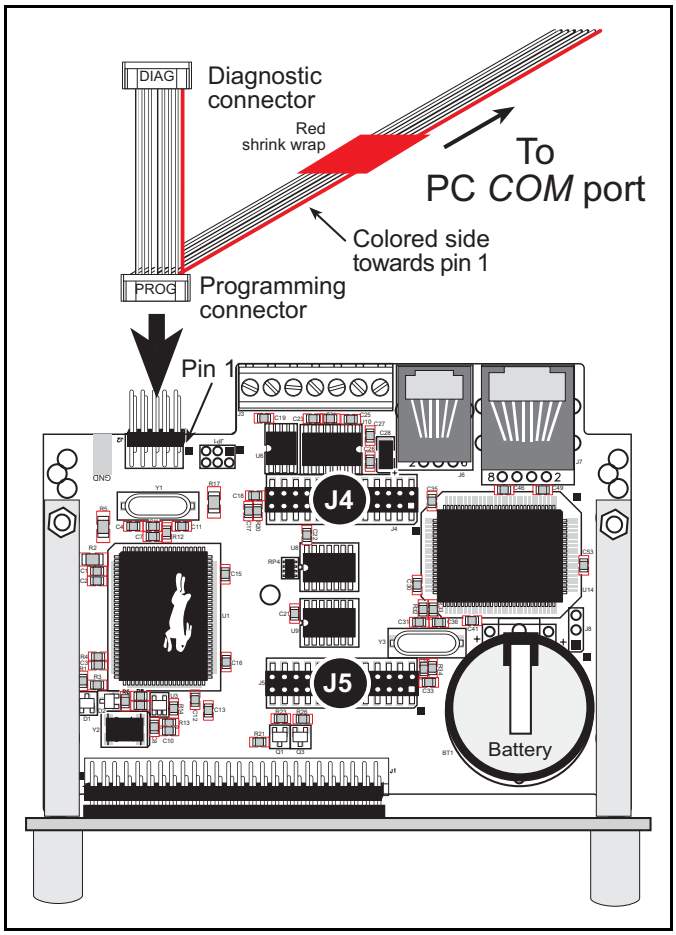



***Figure 4. Programming Cable Connections***

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

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

**2. Apply power.**

Plug the power supply in to a nearby outlet. The CPU Card is now ready to be used.

> **NOTE:** A hardware RESET is accomplished by unplugging the power supply, then plugging it back in.

To power down the SmartStar, unplug the power supply. You should disconnect power before making any circuit adjustments or changing any connections to the SmartStar.

## 2.4 Installing Dynamic C

If you have not yet installed Dynamic C version 7.06P3 (or a later version), do so now by inserting the Dynamic C CD from the Smart Star Tool Kit in your PC's CD-ROM drive. The CD will auto-install unless you have disabled auto-install on your PC.

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

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

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

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

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

## 2.5 Starting Dynamic C

Once the CPU Card is installed and connected as described above, start Dynamic C by double-clicking on the Dynamic C icon or by double-clicking on **`dcrab_XXXX.exe`** in the Dynamic C root directory, where **`XXXX`** are version-specific characters.

Dynamic C defaults to using the serial port on your PC that you specified during installation. If the port setting is correct, Dynamic C should detect the CPU Card and go through a sequence of steps to cold-boot the CPU Card and to compile the BIOS. (Some versions of Dynamic C will not do the initial BIOS compile and load until the first time you compile a program.)

If you receive the message **`No Rabbit Processor Detected`**, the programming cable may be connected to the wrong COM port, a connection may be faulty, or the target system may not be powered up. First, check both ends of the programming cable to ensure that it is firmly plugged into the PC and the programming port.

If there are no faults with the hardware, select a different COM port within Dynamic C. From the **Options** menu, select **Communications**. Select another COM port from the list, then click **OK**. Press **<Ctrl-Y>** to force Dynamic C to recompile the BIOS. If Dynamic C still reports it is unable to locate the target system, repeat the above steps until you locate the active COM port. You should receive a **`Bios compiled successfully`** message once this step is completed successfully.

If Dynamic C appears to compile the BIOS successfully, but you then receive a communication error message when you compile and load a sample program, it is possible that your PC cannot handle the higher program-loading baud rate. Try changing the maximum download rate to a slower baud rate as follows.

- Locate the **Serial Options** dialog in the Dynamic C **Options > Communications** menu. Select a slower Max download baud rate.

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

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

## 2.6 PONG.C

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

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

This program does not test the serial ports on the CPU Card, but does ensure that the CPU is basically functional.

## 2.7 Installing I/O Cards

1. Orient the backplane with the CPU Card already installed and facing towards you as shown in Figure 5.

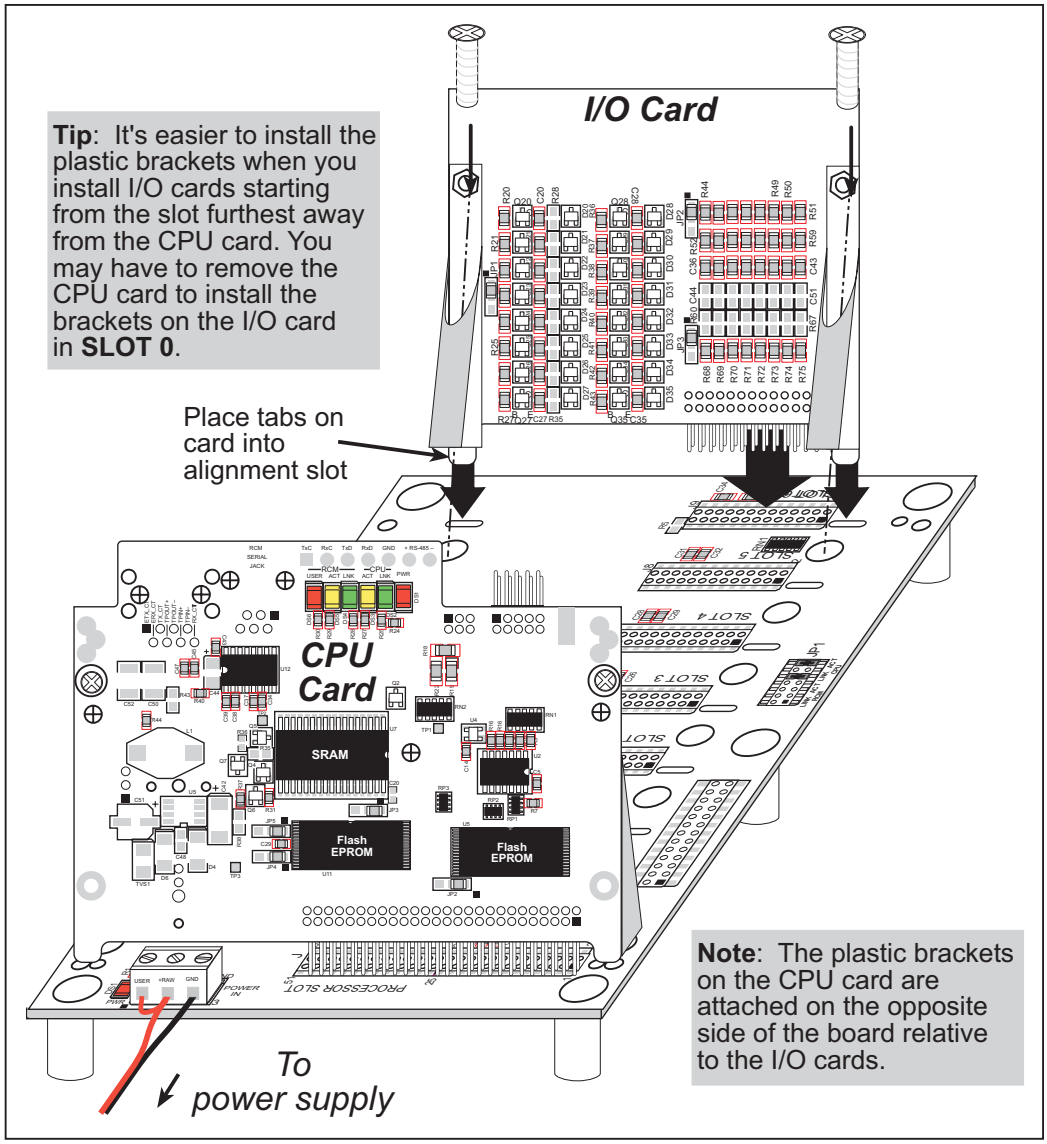



***Figure 5. Installing I/O Cards on the Backplane***

2. Position the new I/O card above the backplane over any unused slot position (**SLOT 0** to **SLOT 6**) as shown in Figure 5. Note the slot number and the type of I/O card since Dynamic C addresses the I/O cards by slot number.

3. Carefully insert the I/O card header into the slot on the backplane and line up the tabs on the I/O cards with the slots on the backplane as shown in Figure 5.

4. Use the two 4-40 screws supplied with the I/O card to anchor the plastic brackets on the CPU Card or the I/O card firmly on the backplane. Tighten the screws as needed using a Phillips screwdriver whose shaft is at least 3" (7 cm) long, but is no thicker than 0.16" (4 mm).

## 2.8 Where Do I Go From Here?

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

If there are any problems at this point:

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

If the sample program ran fine, you are now ready to go on to install I/O cards, explore other Smart Star features, and develop your own applications.

Chapter 3, "Hardware Features," provides detailed information about the CPU Card, and how to install the I/O cards. Be sure to take the total current consumption of the individual cards into account when selecting a power supply. Appendix C.1, "Current Requirements," provides more detailed information. Chapter 4, "Software," describes the Dynamic C software libraries and introduces some sample programs for use with the CPU Card. Chapter 6, "Smart Star Specifications," provides specifications for the backplanes and the CPU cards, including mounting and clearance recommendations.

Separate sections in this manual have been prepared for the various I/O cards, and include complete information about their pinouts and Dynamic C software libraries, including sample programs.

Once you have developed your application and bench-tested the finished system, you may install the finished system.

# 3. HARDWARE FEATURES

Chapter 3 describes the principal features for the Smart Star backplanes and CPU cards.

- Power Distribution
  - Power Distribution
  - I/O Card Slots
- Smart Star CPU Card Features
  - Serial Communication
  - Memory
  - Other Connectors

## 3.1 Backplane Features

### 3.1.1 Power Distribution

Power is supplied to the Smart Star control system from an external source through header J1 on the backplane. The +5 V circuitry on the Smart Star control system is protected against reverse polarity by a Schottky diode as shown in Figure 6.

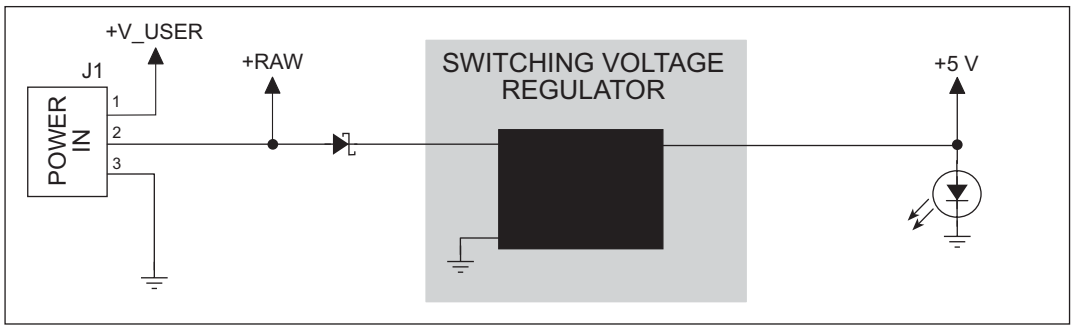



***Figure 6. Smart Star Control System Power Supply Schematic***

A capacitor provides surge current protection for the voltage regulator, and allows the external power supply to be located some distance away from the Smart Star control system. A switching power regulator is used. The **+RAW** input voltage may range from 9 V to 30 V (15 V to 30 V you plan to use a D/A Converter Card).

The backplane has inputs for two separate power supplies on header J1, **+RAW** and **V_USER**. The **+RAW** power supply goes to the switching power regulator, which outputs the +5 V DC used by the CPU Card and by the I/O cards plugged into the backplane. The **V_USER** connection allows a different voltage to be available on the I/O cards for future development.

> **NOTE:** Always connect **V_USER** to **+RAW** with a jumper wire between terminals 1 and 2 on header J1 for the development activities described in the Smart Star manuals.

Figure 7 shows how the power supplies are distributed on the backplane and on the CPU Card.

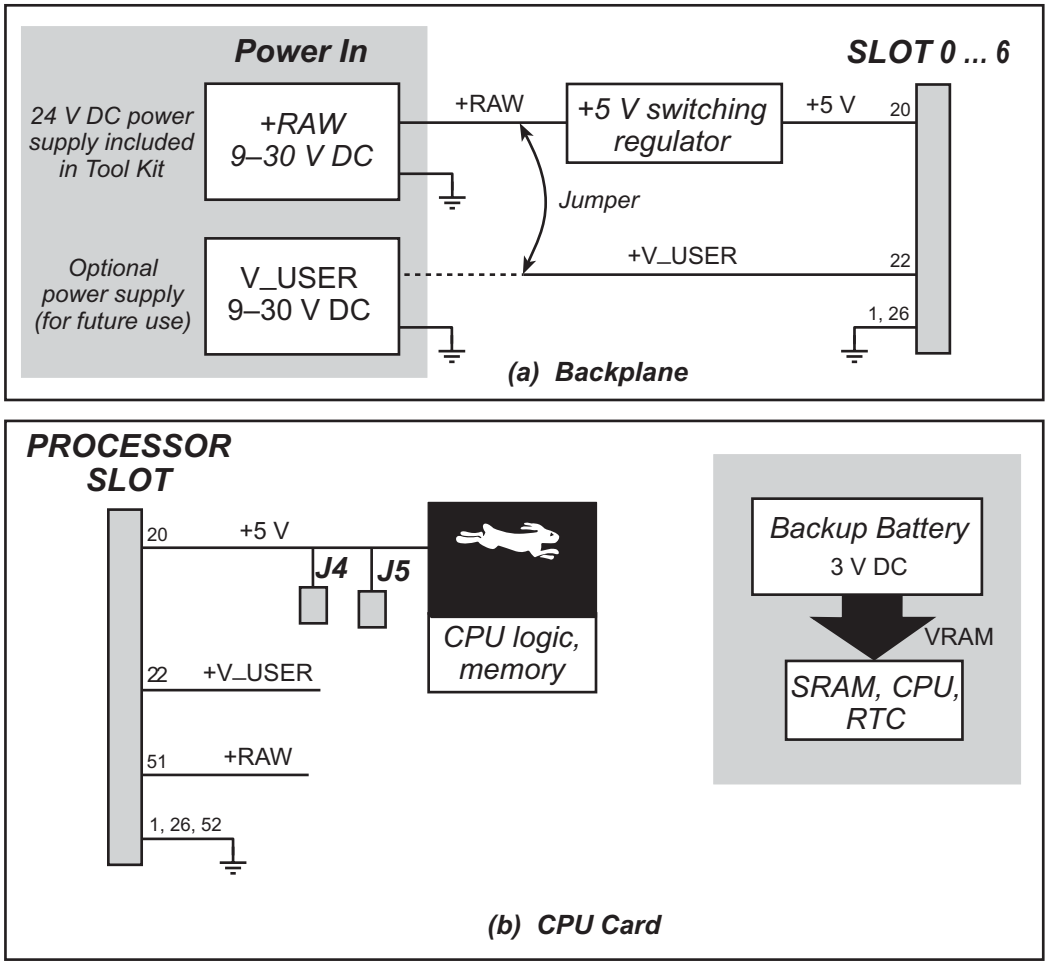



***Figure 7. Smart Star Power Supplies—Backplane and CPU Card***

Figure 8 shows how the power supplies are distributed on the I/O cards.

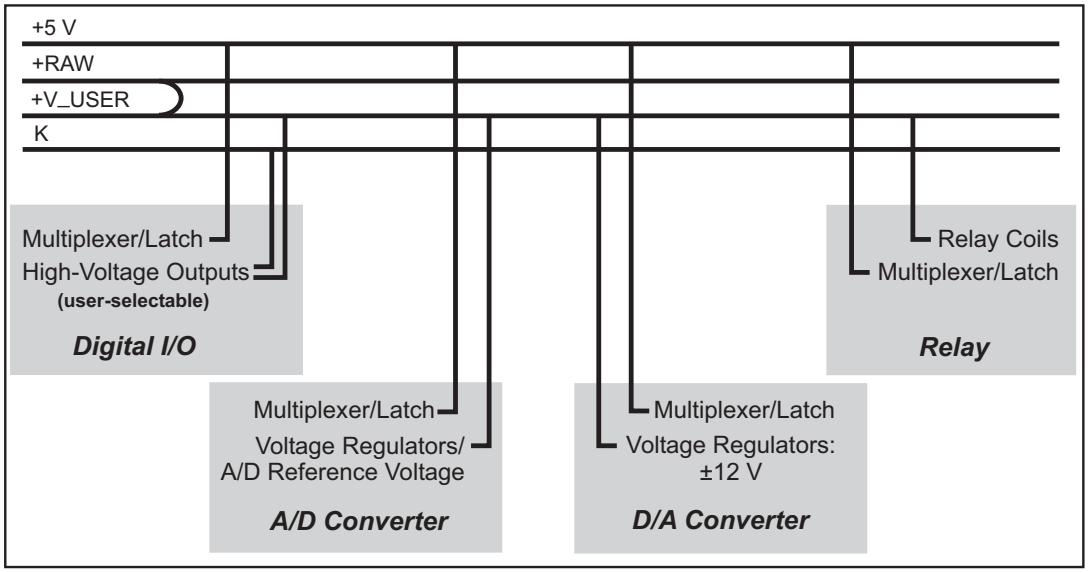



***Figure 8. Smart Star Power Distribution on I/O Cards***

**NOTE:** Note that Rabbit recommends tying **+RAW** to **+V_USER** as explained in Section 2.2, "Connect the Power Supply."

The user has the option of using a separate power supply to K when configuring the high-power outputs for the digital I/O cards. The connection to K is through the user interface on the digital I/O card. Further details are provided in Chapter 7, "Digital I/O Cards."

### 3.1.2 I/O Card Slots

The backplane serves to make the CPU Card accessible to up to seven I/O cards plugged in to **SLOT 0** through **SLOT 6** on the backplane. Figure 9 shows the pinout for **SLOT 0** through **SLOT 6** (headers J3–J9) on the backplane.

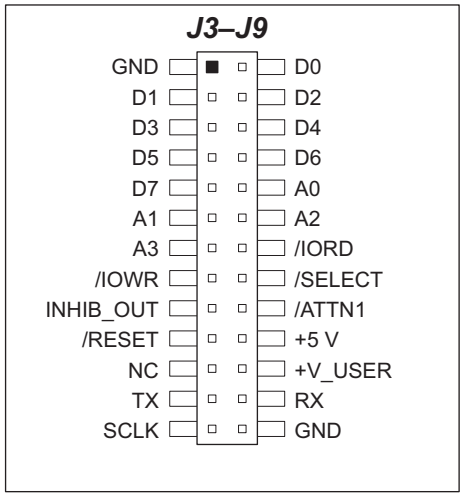



***Figure 9. Pinout for SLOT 0 Through SLOT 6 (Headers J3–J9) on the Backplane***

**NOTE:** The SR9050 backplane can accommodate up to three I/O cards plugged in to **SLOT 0** through **SLOT 2** (headers J3–J5).

## 3.2 Smart Star CPU Card Features

### 3.2.1 Serial Communication

The CPU Card has one screw terminal header for RS-232/RS-485 serial communication (J3) and one RJ-45 Ethernet jack (J7, SR9150 only). The RJ-12 jack, J6, is reserved for future use and therefore has no signals. The pinouts are shown in Figure 10.

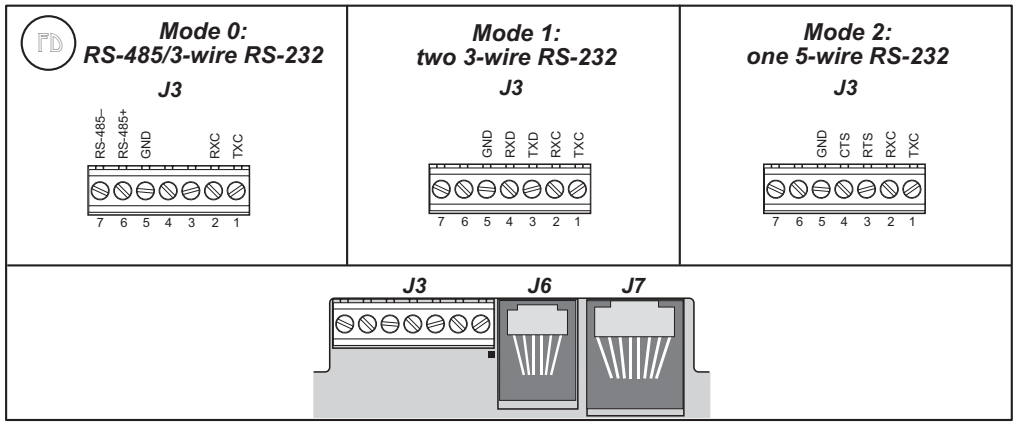



***Figure 10. Smart Star CPU Card Serial Pinout***

The factory default for the CPU Card is one RS-232 (3-wire) and one RS-485 serial channel, corresponding to Mode 0 in Figure 10. The other modes shown in Figure 10 are set in software via the Dynamic C **`serMode`** function call (see Section 4.5, “Serial Communication Calls”).

#### 3.2.1.1 RS-232

The CPU Card’s RS-232 serial channel is connected to an RS-232 transceiver. The transceiver provides the voltage output, slew rate, and input voltage immunity required to meet the RS-232 serial communication protocol. Basically, the chip translates the Rabbit 2000’s 0 V to +Vcc signals to RS-232 signal levels. Note that the polarity is reversed in an RS-232 circuit so that +5 V is output as approximately -10 V and 0 V is output as approximately +10 V. The transceiver also provides the proper line loading for reliable communication.

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

The Rabbit 2000 serial port C TXD and RXD signals are presented either as RS-232 TX and RX or as RTS/CTS handshaking, depending on the mode selected with the Dynamic C function **`serMode`**. The RS-232 signals are available on screw terminal header J3.

#### 3.2.1.2 RS-485

The CPU Card has one RS-485 serial channel, which is connected to the Rabbit 2000 serial port C through an RS-485 transceiver. The chip's slew rate limiters provide for a maximum baud rate of 250,000 bps, and allows networking over a distance of up to 300 m (or 1000 ft.). The half-duplex communication uses the Rabbit 2000's PD4 pin to control the data enable on the communication line.

The RS-485 signals are available on the CPU Card through screw terminal header J3.

The Smart Star control system can be used in an RS-485 multidrop network. Connect the 485+ to 485+ and 485– to 485– using single twisted-pair wires on the CPU Card's header J4 as shown in Figure 11. Note that a common ground is recommended.

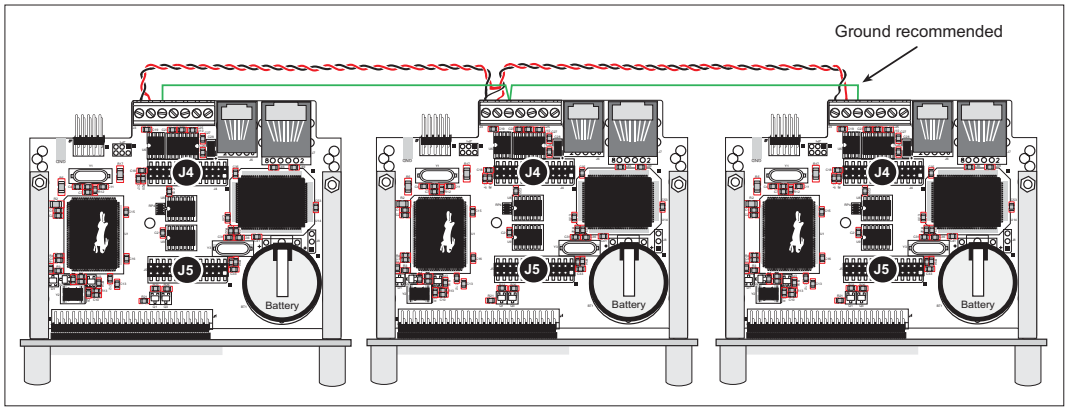



***Figure 11. Multidrop Smart Star Network***

The CPU Card comes with a 220 Ω termination resistor and 681 Ω bias resistors already installed and enabled with jumpers across pins 1–2 and 5–6 on header JP1, as shown in Figure 12.

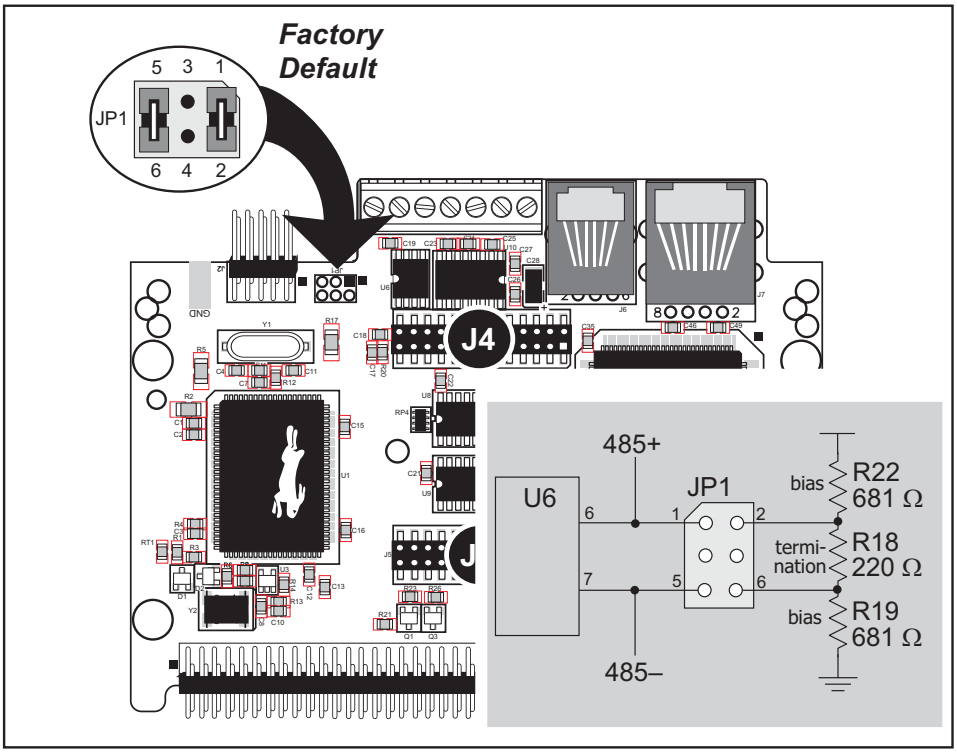



***Figure 12. RS-485 Termination and Bias Resistors***

For best performance, the bias and termination resistors in a multidrop network should only be enabled on both end nodes of the network. Disable the termination and bias resistors on any intervening Smart Star units in the network by removing both jumpers from header JP1.

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

#### 3.2.1.3 Programming Port

The CPU Card has a 10-pin programming header labeled J2. The programming port uses the Rabbit 2000's Serial Port A for communication. Dynamic C uses the programming port to download and debug programs.

The programming port is also used for the following operations.

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

**Alternate Uses of the Serial Programming Port**

All three clocked Serial Port A signals are available as

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

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

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

The two startup mode pins determine what happens after a reset—the Rabbit 2000 is either cold-booted or the program begins executing at address 0x0000.

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

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

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

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

#### 3.2.1.4 **Ethernet Port** (SR9150 only)

Figure 13 shows the pinout for the Ethernet port (J2 on the CPU Card). Note that there are two standards for numbering the pins on this connector—the convention used here, and numbering in reverse to that shown. Regardless of the numbering convention followed, the pin positions relative to the spring tab position (located at the bottom of the RJ-45 jack in Figure 13) are always absolute, and the RJ-45 connector will work properly with off-the-shelf Ethernet cables.

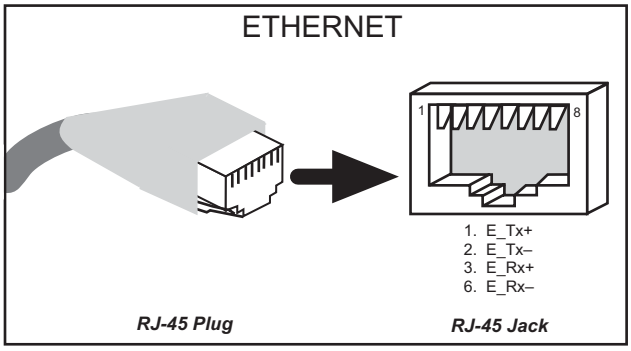



***Figure 13. RJ-45 Ethernet Port Pinout***

RJ-45 pinouts are sometimes numbered opposite to the way shown in Figure 13.

Two LEDs are placed behind the RJ-45 Ethernet jack, one to indicate an Ethernet link (**LNK**) and one to indicate Ethernet activity (**ACT**). Only the CPU LEDs are functional at this time since the RCM LEDs were added for future enhancements to the CPU Card.

The transformer/connector assembly ground is connected to the CPU Card digital ground via a 0 Ω resistor "jumper," R43, as shown in Figure 14.

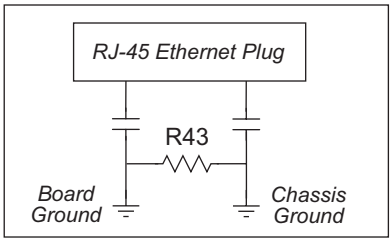



***Figure 14. Isolation Resistor R43***

The factory default is for the 0 Ω resistor "jumper" at R43 to be installed. In high-noise environments, remove R43 and ground the transformer/connector assembly directly through the chassis ground. This will be especially helpful to minimize ESD and/or EMI problems.

## 3.3 Programming Cable

The programming cable is used to connect the programming port of the Smart Star CPU Card to a PC serial COM port. The programming cable converts the RS-232 voltage levels used by the PC serial port to the TTL voltage levels used by the Rabbit 2000.

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

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

### 3.3.1 Changing Between Program Mode and Run Mode

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

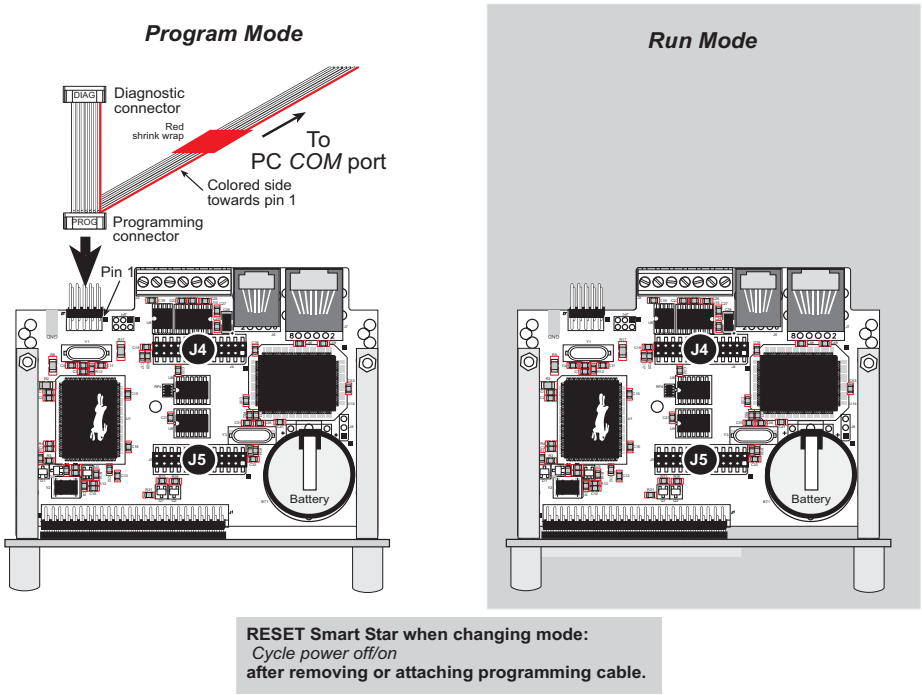



*Figure 15. Smart Star Program Mode and Run Mode Set-Up*

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

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

## 3.3.2 Memory

### 3.3.2.1 SRAM

The Smart Star CPU Cards are designed to accept 128K or 512K of static RAM packaged in an SOIC case. Standard CPU Cards come with 128K of SRAM.

### 3.3.2.2 Flash EPROM

The Smart Star CPU Card are also designed to accept 128K to a total of 512K of flash memory packaged in a TSOP case. The CPU cards come with two 256K flash memory chips.

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

A Flash Memory Bank Select jumper configuration option based on 0 Ω surface-mounted resistors exists at header JP5 on the CPU Card. This option, used in conjunction with some configuration macros, allows Dynamic C to compile two different co-resident programs for the upper and lower halves of the 256K flash in such a way that both programs start at logical address 0000. This is useful for applications that require a resident download manager and a separate downloaded program. See Technical Note 218, ***Implementing a Serial Download Manager for a 256K Flash***, for details.

### 3.3.3 Other Connectors

The connectors labeled J4 and J5 in Figure 16 are reserved for future use and should not be used in customer applications at this time.

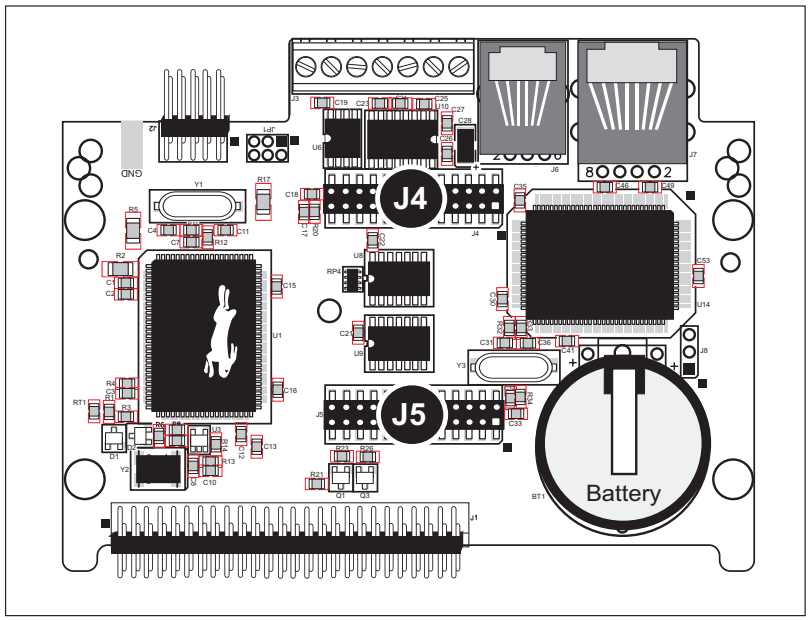



*Figure 16. CPU Card Connectors J4 and J5*

Jumpers across pins 9–10 and 13–14 on header JP1 on the backplane are used to bring out the **ACT** and **LNK** LED signals to header J6, which is used to connect the optional LCD/keypad module. Remove these jumpers (you may park them across pins 7–8 and 11–12 on header JP1) if you do not wish to use the **ACT** and **LNK** signals on the LCD/keypad module.

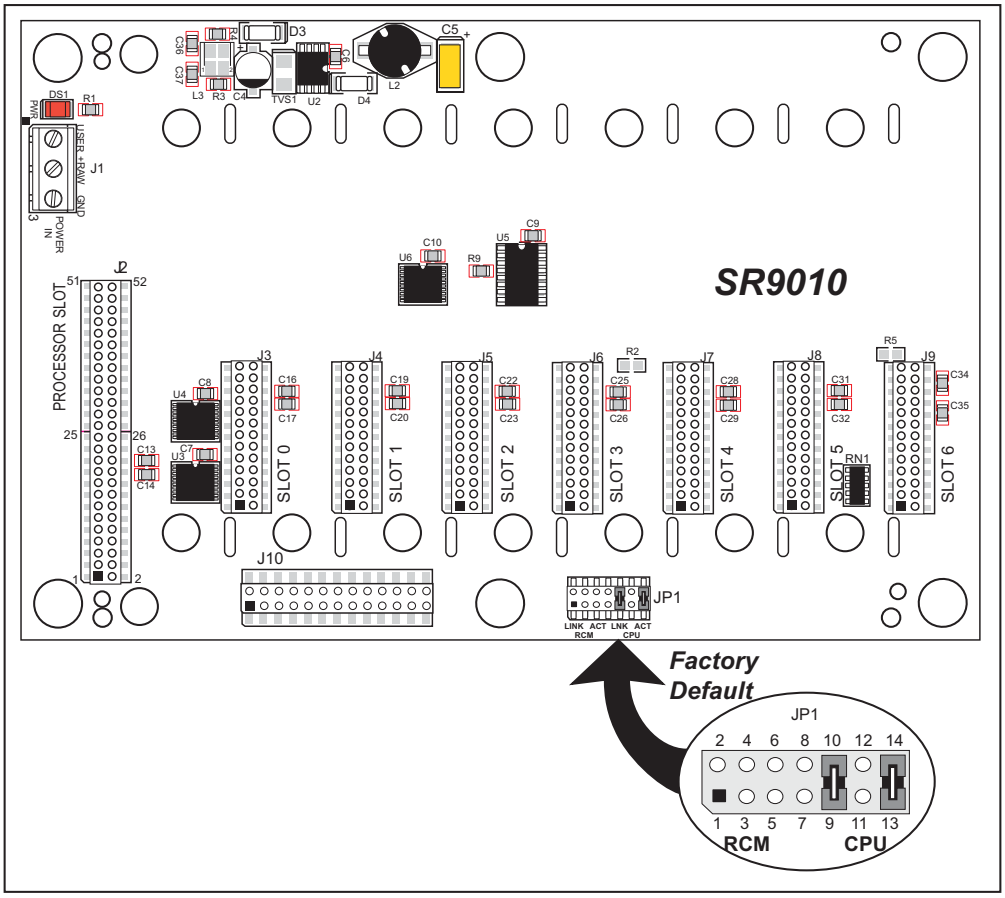



***Figure 17. Header JP1 Configurations for* ACT *and* LNK *Signals***

**NOTE:** The RCM positions for pins 1–2 and 5–6 on header JP1 are reserved for future use and should not be used in customer applications at this time.

## 3.4 Other Hardware

### 3.4.1 Clock Doubler

The Smart Star CPU cards take advantage of the Rabbit 2000 microprocessor's internal clock doubler. A built-in clock doubler allows half-frequency crystals to be used to reduce radiated emissions. The 22.1 MHz frequency is generated using an 11.0592 MHz crystal. The clock doubler is disabled automatically in the BIOS for crystals with a frequency above 12.9 MHz.

The clock doubler may be disabled if 22.1 MHz clock speeds are not required. Disabling the Rabbit 2000 microprocessor's internal clock doubler will reduce power consumption and further reduce radiated emissions. The clock doubler is disabled with a simple configuration macro as shown below.

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

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

### 3.4.2 Spectrum Spreader

Smart Star CPU cards that carry the CE mark have a Rabbit 2000 microprocessor that features a spectrum spreader, which helps to mitigate EMI problems. By default, the spectrum spreader is on automatically for CPU cards that carry the CE mark when used with Dynamic C 7.32 or later versions, but the spectrum spreader may also be turned off or set to a stronger setting. The means for doing so is through a simple configuration macro as shown below.
.

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

   ```
   ENABLE_SPREADER=1
   ```

   For strong spreading, add the line

   ```
   ENABLE_SPREADER=2
   ```

   To disable the spectrum spreader, add the line

   ```
   ENABLE_SPREADER=0
   ```

   **NOTE:** The strong spectrum-spreading setting is unnecessary for the Smart Star.
3. Click **OK** to save the macro. The spectrum spreader will now remain off whenever you are in the project file where you defined the macro.

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

# 4. SOFTWARE

Dynamic C is an integrated development system for writing embedded software. It runs on an IBM-compatible PC and is designed for use with Rabbit controllers and other controllers based on the Rabbit microprocessor.

Chapter 4 provides the libraries, function calls, and sample programs related to the Smart Star backplane and CPU cards.

## 4.1 Running Dynamic C

You have a choice of doing your software development in the flash memory or in the static RAM included on the Smart Star CPU cards. The flash memory and SRAM options are selected with the **Options > Program Options > Compiler** menu.

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

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

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

Developing software with Dynamic C is simple. Users can write, compile, and test C and assembly code without leaving the Dynamic C development environment. Debugging occurs while the application runs on the target. Alternatively, users can compile a program to an image file for later loading. Dynamic C runs on PCs under Windows 95 or later. Programs can be downloaded at baud rates of up to 460,800 bps after the program compiles.

Dynamic C has a number of standard features.

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

### 4.1.1 Upgrading Dynamic C

#### 4.1.1.1 Patches and Bug Fixes

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

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

#### 4.1.1.2 Upgrades

Dynamic C installations are designed for use with the board they are included with, and are included at no charge as part of our low-cost kits. Dynamic C is a complete software development system, but does not include all the Dynamic C features. Rabbit also offers add-on Dynamic C modules containing the popular µC/OS-II real-time operating system, as well as PPP, Advanced Encryption Standard (AES), and other select libraries. In addition to the Web-based technical support included at no extra charge, a one-year telephone-based technical support module is also available for purchase.

## 4.2 Sample Programs

Sample programs are provided in the Dynamic C **SAMPLES** folder. The sample program **PONG.C** demonstrates the output to the **STDIO** window. The various directories in the **SAMPLES** folder contain specific sample programs that illustrate the use of the corresponding Dynamic C libraries.

The **SAMPLES\SMRTSTAR** folder provides sample programs specific to the Smart Star control system. Each sample program has comments that describe the purpose and function of the program. Follow the instructions at the beginning of the sample program.

To run a sample program, open it with the **File** menu (if it is not still open), compile it using the **Compile** menu, and then run it by selecting **Run** in the **Run** menu. The BL2500 must be in **Program** mode (see Section 3.3, "Programming Cable") and must be connected to a PC using the programming cable as described in Section 2.2, "Connect the Power Supply."

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

Let's take a look at sample programs for the backplane and the CPU Card in the **SMRTSTAR** folder.

The **RS232** directory contains two sample programs to illustrate RS-232 serial communication.

- **SSTAR232.C**—Demonstrates a simple RS-232 loopback using both serial ports C and D.
- **SSTAR5W.C**—Demonstrates simple 5-wire RS-232 communication with flow control.

The **RS485** directory contains two sample programs to illustrate RS-485 serial communication.

- **MASTER.C**—Demonstrates a simple RS-485 transmission of lower case letters to a slave controller. The slave will send converted upper case letters back to the master controller for display in the **STDIO** window. Use **SLAVE.C** to program the slave controller.
- **SLAVE.C**—Demonstrates a simple RS-485 transmission of alphabetic characters to a master controller. The slave will send converted upper case letters back to the master controller for display in the **STDIO** window. Use **MASTER.C** to program the master controller.

## 4.3 Dynamic C Libraries

One library directory contains software that is unique to the Smart Star.

- **`SMRTSTAR.LIB`**—This library supports all the functions needed by the Smart Star systems including Digital I/O Cards, Relay Cards, D/A Converter and A/D Converter Cards, and serial communication.

Functions dealing with the backplane and the CPU Card are described in this chapter. Functions relevant to the individual I/O cards are described in the chapter specific to the I/O card.

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

## 4.4 Smart Star Backplane Function Calls

### 4.4.1 Board Reset

```
void brdResetBus();
```

Resets all cards on the bus.

**RETURN VALUE**

None.

### 4.4.2 Board Initialization

```
void brdInit();
```

Initializes slot addressing, disables card enable/disable line, resets card slot bus and LED latch, and turns all LEDS OFF. Call this function at the beginning of the application.

**RETURN VALUE**

None.

## 4.5 Serial Communication Calls

Library files included with Dynamic C provide a full range of serial communications support. The **RS232.LIB** library provides a set of circular-buffer-based serial functions. The **PACKET.LIB** library provides packet-based serial functions where packets can be delimited by the 9th bit, by transmission gaps, or with user-defined special characters. Both libraries provide blocking functions, which do not return until they are finished transmitting or receiving, and nonblocking functions, which must be called repeatedly until they are finished. For more information, see the ***Dynamic C Function Reference Manual*** and Technical Note 213, ***Rabbit 2000 Serial Port Software***.

Use the following function calls with the Smart Star.

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

User interface to set up serial communication lines for the Smart Star control system. Call this function after **serXOpen()**.

**PARAMETERS**

**mode** is the defined serial port configuration of the CPU Card.

| Mode | Serial Port | | Parallel Port |
|---|---|---|---|
| | C (PC2 and PC3) | D (PC0 and PC1) | D (PD0 and PD1) |
| 0 | RS-232, 3-wire | RS-485 | |
| 1 | RS-232, 3-wire | RS-232, 3-wire | |
| 2 | RS-232, 5-wire | | RTS/CTS |
| 3 | RS-232, 5-wire | RS-485 | RTS/CTS |

**RETURN VALUE**

0 if correct mode, 1 if not.

```
ser485Tx();
```

Enables RS-485 transmission (disables receive) on serial port D.

**RETURN VALUE**

None.

**SEE ALSO**

**ser485Rx**

## ser485Rx();

Disables RS-485 transmission (enables receive) on serial port D.

**RETURN VALUE**

None.

**SEE ALSO**

**ser485Tx**

# 5. USING THE TCP/IP FEATURES

Chapter 5 discusses using the TCP/IP features on the CPU cards. Note that the TCP/IP feature is available only on the SR9150 CPU Card.

## 5.1 Ethernet Connections

Before proceeding you will need to have the following items.

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

The Ethernet cables and Ethernet hub are available from Rabbit in a TCP/IP tool kit. More information is available at www.rabbit.com.

1. **Install the CPU Card on the backplane, and connect the power supply and the programming cable as shown in Chapter 2, "Getting Started."**
2. **Ethernet Connections**

- If you do not have access to an Ethernet network, use a crossover Ethernet cable to connect the installed CPU Card to a PC that at least has a 10Base-T Ethernet card.
- If you have an Ethernet connection, use a straight-through Ethernet cable to establish an Ethernet connection to the installed CPU Card from an Ethernet hub. These connections are shown in Figure 18.

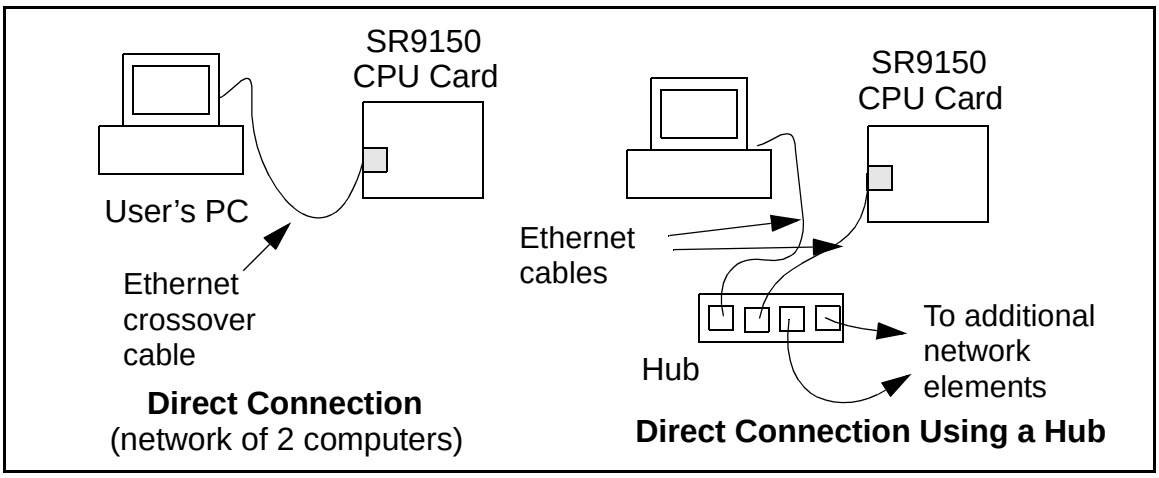



*Figure 18. Ethernet Connections*

### 3. Apply Power

Plug in the power supply. The Smart Star is now ready to be used.

> **NOTE:** A hardware RESET is accomplished by unplugging the power supply, then plugging it back in.

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

## 5.2 TCP/IP Sample Programs

We have provided a number of sample programs demonstrating various uses of TCP/IP for networking embedded systems. These programs require that you connect your PC and the Smart Star together on the same network. This network can be a local private network (preferred for initial experimentation and debugging), or a connection via the Internet.

### 5.2.1 How to Set IP Addresses in the Sample Programs

With the introduction of Dynamic C 7.30 we have taken steps to make it easier to run many of our sample programs. You will see a **TCPCONFIG** macro. This macro tells Dynamic C to select your configuration from a list of default configurations. You will have three choices when you encounter a sample program with the **TCPCONFIG** macro.

1. You can replace the **TCPCONFIG** macro with individual **MY_IP_ADDRESS**, **MY_NETMASK**, **MY_GATEWAY**, and **MY_NAMESERVER** macros in each program.
2. You can leave **TCPCONFIG** at the usual default of 1, which will set the IP configurations to **10.10.6.100**, the netmask to **255.255.255.0**, and the nameserver and gateway to **10.10.6.1**. If you would like to change the default values, for example, to use an IP address of **10.1.1.2** for the CPU Card, and **10.1.1.1** for your PC, you can edit the values in the section that directly follows the "General Configuration" comment in the **TCP_CONFIG.LIB** library. You will find this library in the **LIB\TCPIP** directory.
3. You can create a **CUSTOM_CONFIG.LIB** library and use a **TCPCONFIG** value greater than 100. Instructions for doing this are at the beginning of the **TCP_CONFIG.LIB** library in the **LIB\TCPIP** directory.

There are some other "standard" configurations for **TCPCONFIG** that let you select different features such as DHCP. Their values are documented at the top of the **TCP_CONFIG.LIB** library in the **LIB\TCPIP** directory. More information is available in the ***Dynamic C TCP/IP User's Manual***.

#### IP Addresses Before Dynamic C 7.30

Most of the sample programs use macros to define the IP address assigned to the CPU Card and the IP address of the gateway, if there is a gateway. Instead of the **TCPCONFIG** macro, you will see a **MY_IP_ADDRESS** macro and other macros.

```
#define MY_IP_ADDRESS "10.10.6.170"
#define MY_NETMASK "255.255.255.0"
#define MY_GATEWAY "10.10.6.1"
#define MY_NAMESERVER "10.10.6.1"
```

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

```
#define MY_IP_ADDRESS "10.1.1.2"
#define MY_NETMASK "255.255.255.0"
// #define MY_GATEWAY "10.10.6.1"
// #define MY_NAMESERVER "10.10.6.1"
```

In this case, the gateway and nameserver are not used, and are commented out. The IP address of the CPU Card is defined to be **10.1.1.2**. The IP address of you PC can be defined as **10.1.1.1**.

### 5.2.2 How to Set Up Your Computer for Direct Connect

Follow these instructions to set up your PC or notebook. Check with your administrator if you are unable to change the settings as described here since you may need administrator privileges. The instructions are specifically for Windows 2000, but the interface is similar for other versions of Windows.

> **TIP:** If you are using a PC that is already on a network, you will disconnect the PC from that network to run these sample programs. Write down the existing settings before changing them to facilitate restoring them when you are finished with the sample programs and reconnect your PC to the network.

1. Go to the control panel (**Start > Settings > Control Panel**), and then double-click the Network icon.
2. Select the network interface card used for the Ethernet interface you intend to use (e.g., **TCP/IP Xircom Credit Card Network Adapter**) and click on the "Properties" button. Depending on which version of Windows your PC is running, you may have to select the "Local Area Connection" first, and then click on the "Properties" button to bring up the Ethernet interface dialog. Then "Configure" your interface card for a "10Base-T Half-Duplex" or an "Auto-Negotiation" connection on the "Advanced" tab.

   > **NOTE:** Your network interface card will likely have a different name.

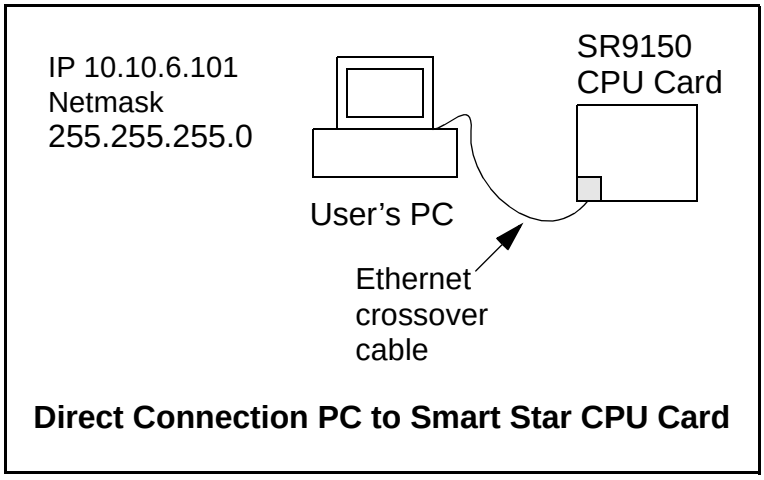
3. Now select the **IP Address** tab, and check **Specify an IP Address**, or select TCP/IP and click on "Properties" to assign an IP address to your computer (this will disable "obtain an IP address automatically"):

   IP Address : 10.10.6.101

   Netmask : 255.255.255.0

   Default gateway : 10.10.6.1

4. Click **<OK>** or **<Close>** to exit the various dialog boxes.




**Direct Connection PC to Smart Star CPU Card**

### 5.2.3 Run the `PINGME.C` Demo

Connect the crossover cable from your computer's Ethernet port to the CPU Card's RJ-45 Ethernet connector. Open this sample program from the **`SAMPLES\TCPIP\ICMP`** folder, compile the program, and start it running under Dynamic C. When the program starts running, the green **LNK** light on the CPU Card should be on to indicate that an Ethernet connection is made. (Note: If the **LNK** light does not light, you may not have a crossover cable, or if you are using a hub perhaps the power is off on the hub.)

The next step is to ping the board from your PC. This can be done by bringing up the MS-DOS window and running the ping program:

```
ping 10.10.6.100
```

or by **Start > Run**

and typing the command

```
ping 10.10.6.100
```

Notice that the orange **ACT** light flashes on the CPU Card while the ping is taking place, and indicates the transfer of data. The ping routine will ping the board four times and write a summary message on the screen describing the operation.

### 5.2.4 Additional Demo Programs

The program **`SMTP.C`** (**`SAMPLES\SMRTSTAR\TCPIP\`**) demonstrates a basic Smart Star system using the SMTP library to send an e-mail when a keypress is detected on an LCD/keypad module. In order to run this sample program, edit the IP address as for the pingme program, edit the "mail to" e-mail address, compile the program, and start it executing. An e-mail corresponding to the keypad button that was pressed is sent.

The program **`SSI.C`** (**`SAMPLES\SMRTSTAR\TCPIP\`**) demonstrates how to make the Smart Star CPU Card a Web server. This program allows you to turn the LEDs on an attached LCD/keypad module on and off from a remote Web browser. In order to run these sample programs, edit the IP address as for the pingme program, compile the program, and start it executing. Then bring up your Web browser and enter the following server address: http://10.1.1.2. This should bring up the Web page served by the sample program.

The program **`SSI2.C`** (**`SAMPLES\SMRTSTAR\TCPIP\`**) demonstrates the use of I/O cards via instructions sent from a Web browser. You will need an A/D Converter Card, a D/A Converter Card, or a relay card installed on the backplane in order for the Web browser to be able to initiate changes on one or more of these I/O cards. Before you run this sample program, edit the IP address as for the pingme program, compile the program, and start it executing. The analog outputs will change or the relays will open and close in response to instructions sent from the Web browser.

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

The following sample programs, found in the **TCPIP** subdirectory in **SAMPLES/LCD_Keypad/122x32_1x7**, are demonstrate the features of the LCD/keypad module connected to the backplane. Remember to configure the IP address, netmask, and gateway as indicated in the sample programs.

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

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

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

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

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

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

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

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

## 5.3 Where Do I Go From Here?

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

If there are any problems at this point:

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

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

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

Refer to the ***Dynamic C TCP/IP User's Manual*** to develop your own applications. ***An Introduction to TCP/IP*** provides background information on TCP/IP, and is available on our Web site.

# 6. SMART STAR SPECIFICATIONS

This chapter provides the specifications for the Smart Star back-plane and CPU Card, and describes the conformal coating.

## 6.1 Electrical and Mechanical Specifications

### 6.1.1 Smart Star Backplane

Figure 19 shows the mechanical dimensions for the two Smart Star backplanes.

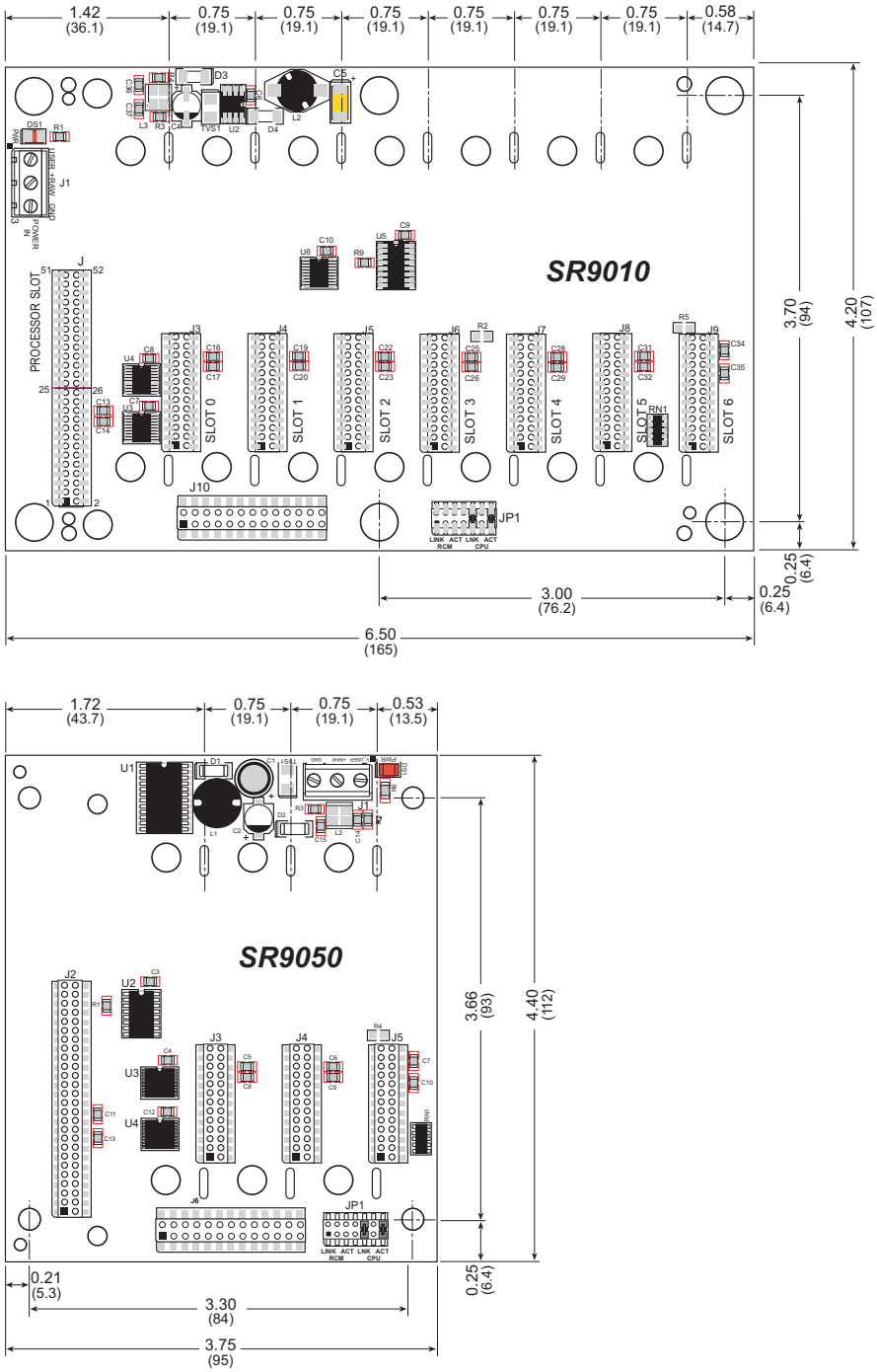



***Figure 19. Smart Star Backplane Dimensions***

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

Table 5 lists the electrical, mechanical, and environmental specifications for the Smart Star backplanes.

***Table 5. Smart Star Backplane Specifications***

| Parameter | Specification | |
|---|---|---|
| | **SR9010** | **SR9050** |
| Board Size | 6.50" × 4.20" × 0.75"<br>(165 mm × 107 mm × 19 mm) | 3.75" × 4.40" × 0.75"<br>(95 mm × 112 mm × 19 mm) |
| Connectors | one 2 × 26 (CPU card slot), 2 mm<br>seven 2 × 13 (I/O card slots), 2 mm | one 2 × 26 (CPU card slot), 2 mm<br>three 2 × 13 (I/O card slots), 2 mm |
| Slot Select | Each slot has a predefined dedicated set of addresses (see Appendix D and the software chapters in the individual I/O card manuals) | |
| Temperature | –40°C to +70°C | |
| Humidity | 5% to 95%, noncondensing | |
| External Input Voltage | 9 V to 30 V DC at 1 A typical for onboard +5 V regulated supply; provision for independent 9 V to 30 V DC (**V_USER**) voltage source for I/O cards—the exact voltage for the second supply depends on the requirements of the specific I/O cards used (Rabbit recommends tying **V_USER** to **+RAW** unless there is a specific need for an independent power supply) | |
| Onboard Voltage Regulator | Surface-mount switching regulator sources 5 V at 1 A | |
| Data Lines | Buffered bidrirectional data lines (D0–D7) | |
| Address Lines | Buffered address lines (A0–A3) | |
| Read/Write Control | Buffered IORD, IOWR | |
| Reset | I/O cards and CPU card can be reset independently | |

### 6.1.2 CPU Card

Figure 20 shows the mechanical dimensions for the CPU cards.

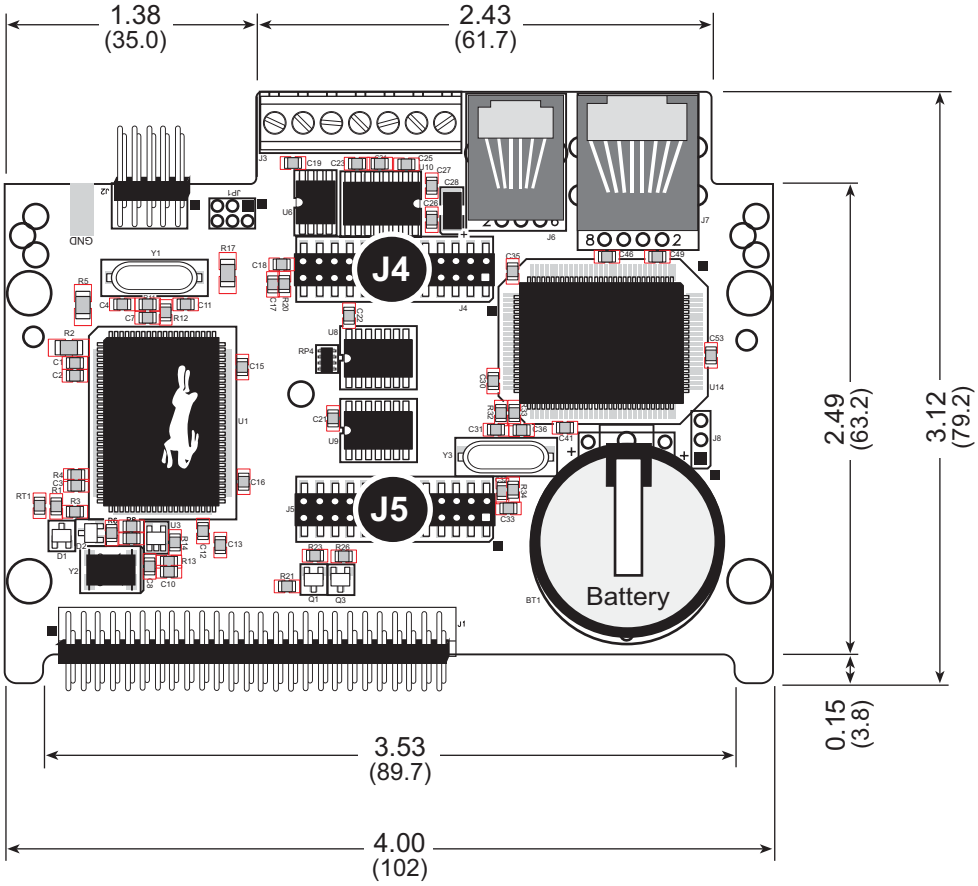



***Figure 20. CPU Card Dimensions***

Table 6 lists the electrical, mechanical, and environmental specifications for the CPU Card.

***Table 6. CPU Card Specifications***

| Parameter | Specification |
|---|---|
| Board Size (with optional backup battery board) | 4.00" × 3.12" × 1.00"<br>(102 mm × 79.2 mm × 25.4 mm) |
| Connectors | one RJ-45 (Ethernet) (SR9150 only)<br>one 2 × 5, 2 mm pitch (serial programming port)<br>one 0.9 mm × 0.5 screw-terminal connector strips (accept 14–30 AWG or 0.05–1.5 mm² wire) |
| Ethernet Interface (SR9150 only) | Direct connection to 10/100-compatible Ethernet networks with 10Base-T interface via RJ-45 connection |
| Temperature | –40°C to +70°C |
| Humidity | 5% to 95%, noncondensing |
| Input Voltage | 5 V DC at 190 mA typical |
| Microprocessor | Rabbit 2000® |
| Clock | 22.1 MHz |
| SRAM | 128K, surface mounted, 512K option |
| Flash EPROM | 2 × 256K, surface mounted |
| Timers | Five 8-bit timers cascadable in pairs, one 10-bit timer with 2 match registers that each have an interrupt |
| Serial Ports | Three serial ports:<br>• one CMOS-compatible programming port<br>• remaining ports software-configurable as two 3-wire RS-232, one 5-wire RS-232, or one 3-wire RS-232/ one RS-485 |
| Serial Rate | Selected baud rates up to 115, 200 bps<br>CMOS-compatible port supports up to 6.45 Mbps (synchronous) |
| Watchdog/Supervisor | Yes |
| Time/Date Clock | Yes |
| Expansion Port | Supports up to 7 I/O cards |
| Backup Battery | Yes: Panasonic CR2330 or equivalent 3 V lithium coin type, 265 mA·h standard using onboard battery holder; provision for external battery |

## 6.2 Jumper Configurations

Figure 21 shows the header locations used to configure the various CPU Card options via jumpers.

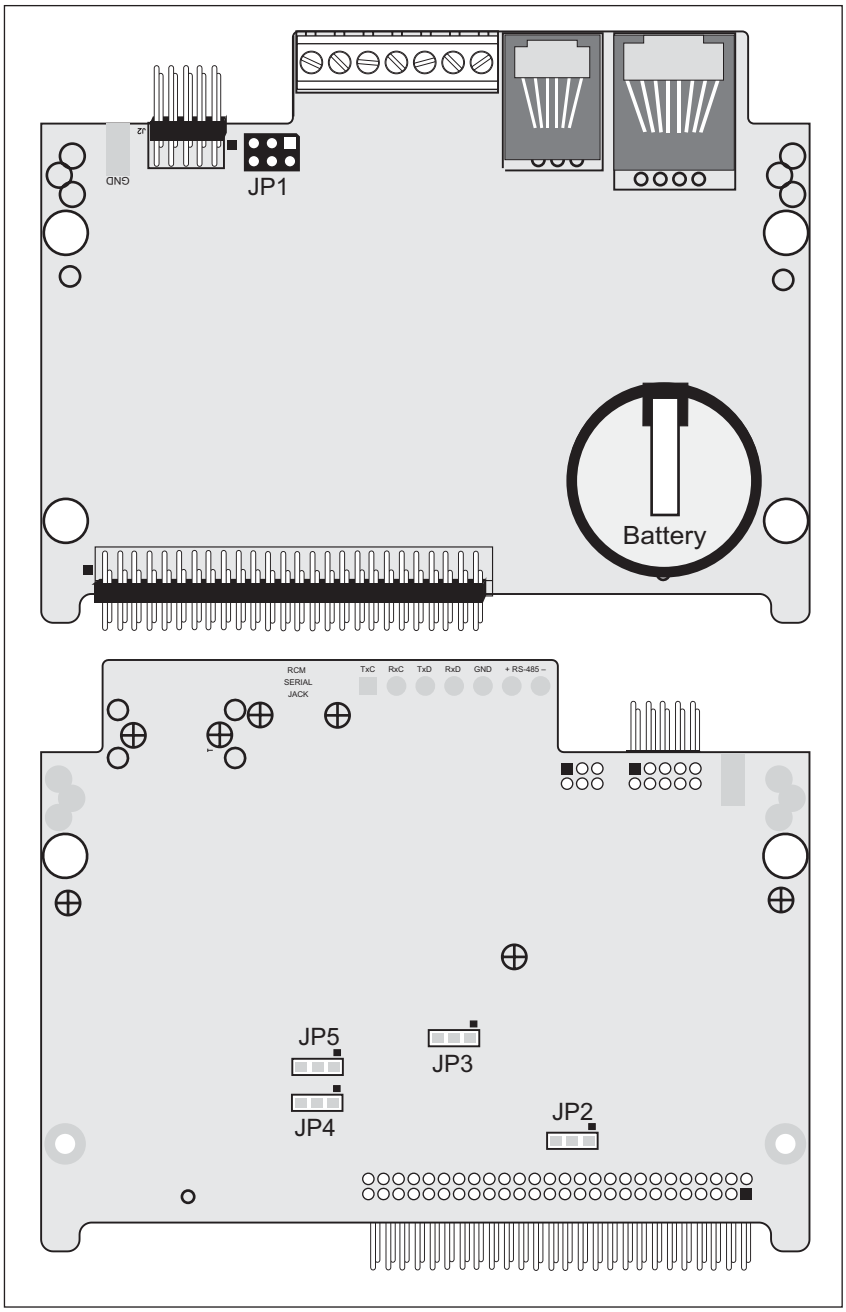



***Figure 21. Location of Smart Star CPU Card Configurable Positions***

Table 7 lists the configuration options.

***Table 7. Smart Star CPU Card Jumper Configurations***

| Header | Description | Pins Connected | | Factory Default |
|---|---|---|---|---|
| JP1 | RS-485 Bias and Termination Resistors | 1–2<br>5–6 | Bias and termination resistors connected | ✕ |
| | | 1–3<br>4–6 | Bias and termination resistors ***not*** connected* | |
| JP2 | U5 Flash Memory Size | 1–2 | 128K/256K | ✕ |
| | | 2–3 | 512K | |
| JP3 | SRAM Size | 1–2 | 128K | ✕ |
| | | 2–3 | 512K | |
| JP4 | U11 Flash Memory Size | 1–2 | 128K/256K | ✕ |
| | | 2–3 | 512K | |
| JP5 | Flash Memory Bank Select | 1–2 | Normal Mode | ✕ |
| | | 2–3 | Bank Mode | |

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

## 6.3 Conformal Coating

The areas around the crystal oscillator and the battery backup circuit on the CPU Card have had the Dow Corning silicone-based 1-2620 conformal coating applied. The conformally coated areas are shown in Figure 22. The conformal coating protects these high-impedance circuits from the effects of moisture and contaminants over time.

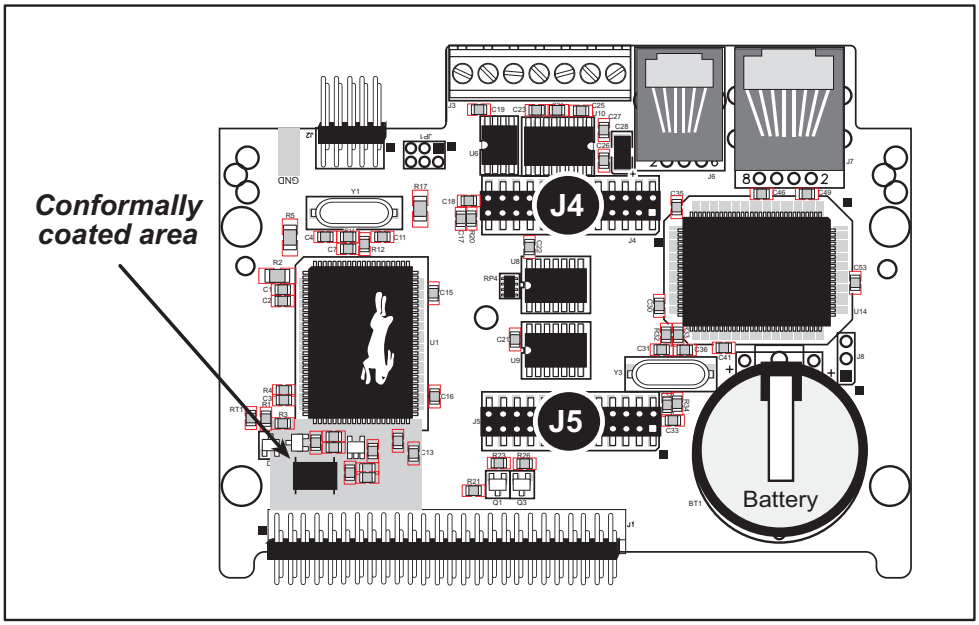



***Figure 22. CPU Card Areas Receiving Conformal Coating***

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

> **NOTE:** For more information on conformal coatings, refer to Rabbit Technical Note TN303, ***Conformal Coatings***.

## 6.4 Use of Rabbit 2000 Parallel Ports

Figure 23 shows the Rabbit 2000 parallel ports.

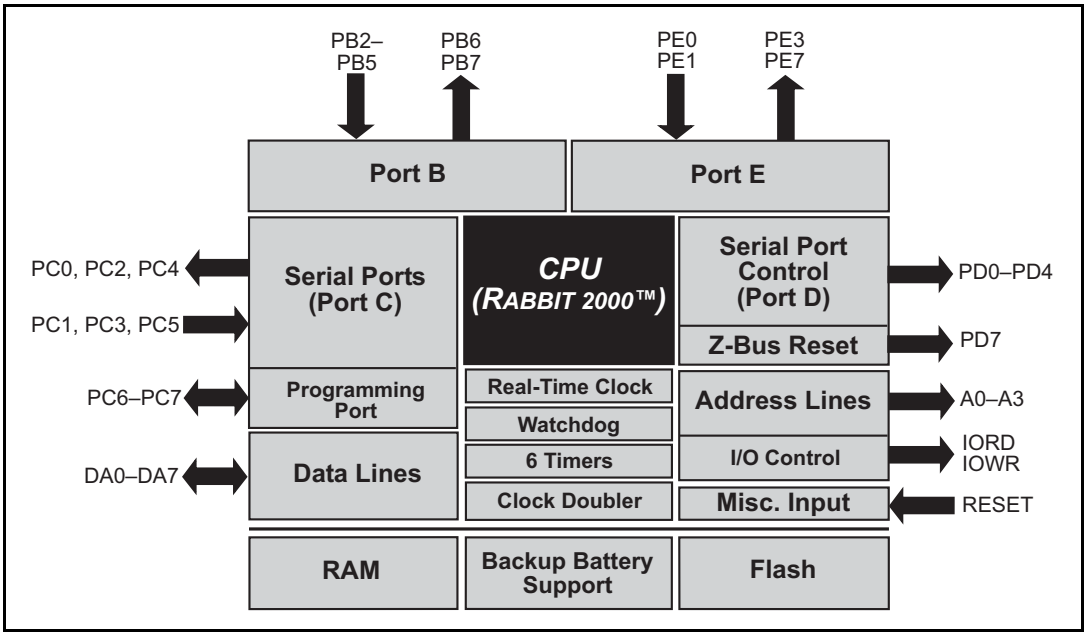



*Figure 23. Smart Star CPU Card Rabbit 2000 Systems*

## 6.5 Exclusion Zone

It is recommended that you allow for an “exclusion zone” of 3" (80 mm) around the Smart Star in all directions when the Smart Star is incorporated into an assembly that includes other components. This “exclusion zone” that you keep free of other components and boards will allow for sufficient air flow, and will help to minimize any electrical or EMI interference between adjacent boards.

# PART II. DIGITAL I/O CARDS

# 7. DIGITAL I/O CARDS

Chapter 7 describes the features of the Digital I/O Card, one of the I/O cards designed for the Smart Star embedded control system.The Smart Star is a modular and expandable embedded control system whose configuration of I/O, A/D Converter, D/A Converter, and Relay Cards can be tailored to a large variety of demanding real-time control and data acquisition applications.

The typical Smart Star system consists of a rugged backplane with a power supply, a CPU card, and one or more I/O cards. The CPU Card plugs into a designated slot on the backplane chassis, which has seven additional slots available for I/O cards to be used in any combination. A high-performance Rabbit 2000 microprocessor on the CPU Card provides fast data processing.

## 7.1 Features

The SR9200 Digital I/O Cards offer protected digital inputs and high-current driver outputs in three banks, each containing 8 I/O points. One bank's configuration is fixed as protected digital inputs, one bank's configuration is fixed as high-current driver outputs, and one bank may be configured either as protected digital inputs or as high-current driver outputs, depending on the model of Digital I/O Card selected. The high-current driver outputs are either all sinking or all sourcing, depending on the model of Digital I/O Card selected.

Table 8 lists the Digital I/O Cards that are available for the Smart Star control system.

***Table 8. Smart Star Digital I/O Cards***

| I/O Card | Model | Features |
|---|---|---|
| Digital I/O | SR9200 | 16 digital inputs, 8 digital sinking outputs |
| | SR9210 | 8 digital inputs, 16 digital sinking outputs |
| | SR9220 | 8 digital inputs, 8 digital sinking outputs |
| | SR9205 | 16 digital inputs, 8 digital sourcing outputs |
| | SR9215 | 8 digital inputs, 16 digital sourcing outputs |
| | SR9225 | 8 digital inputs, 8 digital sourcing outputs |

## 7.2 User Interface

Figure 24 shows the complete pinout for the user interface on header J2. Note that pin 1 is indicated by a small arrow on the ribbon cable connector.

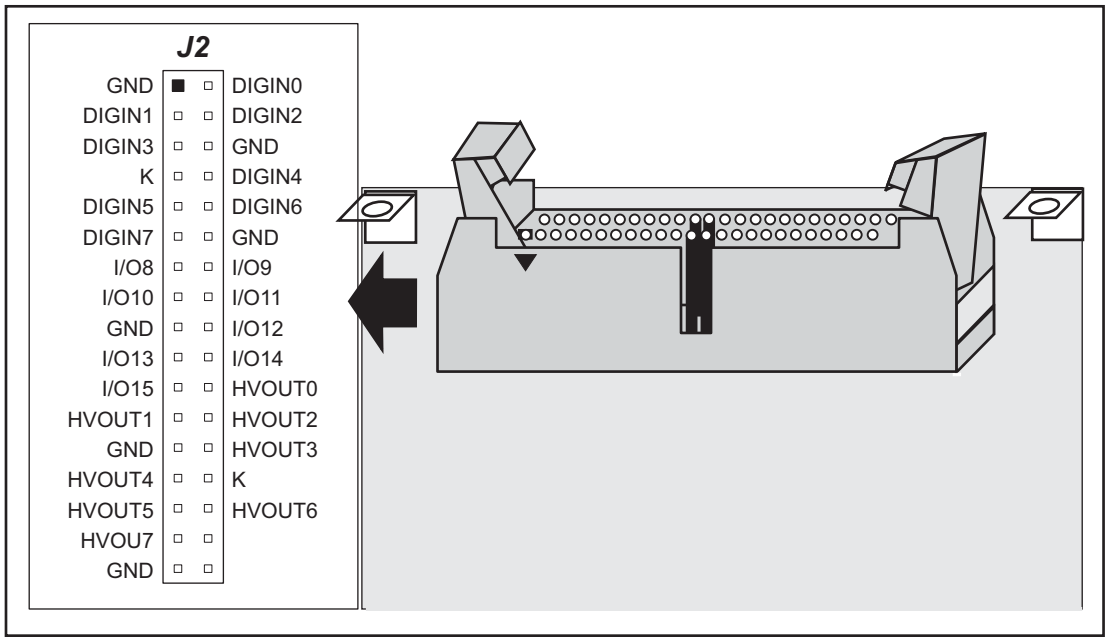
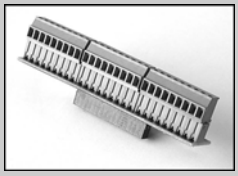
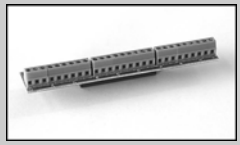



***Figure 24. Digital I/O Card User Interface Pinout***

## 7.3 User FWT Connections

Connections to the Digital I/O Cards are made via a ribbon cable connector or optional field wiring terminals that are either pluggable or have screw terminals. Table 9 lists the Rabbit part numbers for the FWTs.

***Table 9. Guide to FWT Selection***

| FWT Description | I/O Cards | Rabbit Part Number | |
|---|---|---|---|
| | | Pluggable Terminals | Screw Terminals |
| | |  |  |
| FWT27 | Digital I/O | 101-0420 | 101-0514 |

### 7.3.1 Pinouts

Figure 25 shows the pinout for FWT27s used on Digital I/O Cards. Note that only 23 of the I/O points are available on the FWT27—the HVOUT7 digital output is not available on the FWT27.

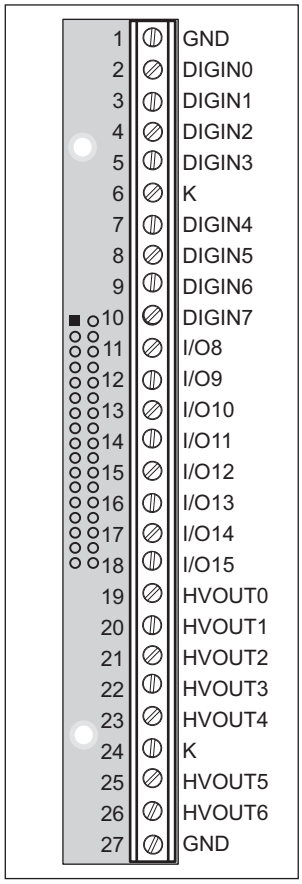



***Figure 25. FWT Pinout for Digital I/O Cards***

## 7.4 Digital Inputs and Outputs

The Digital I/O Card has 24 I/O points that are factory configured as either inputs or outputs in banks of eight, depending on the model.

Figure 26 shows the locations of the I/O banks.

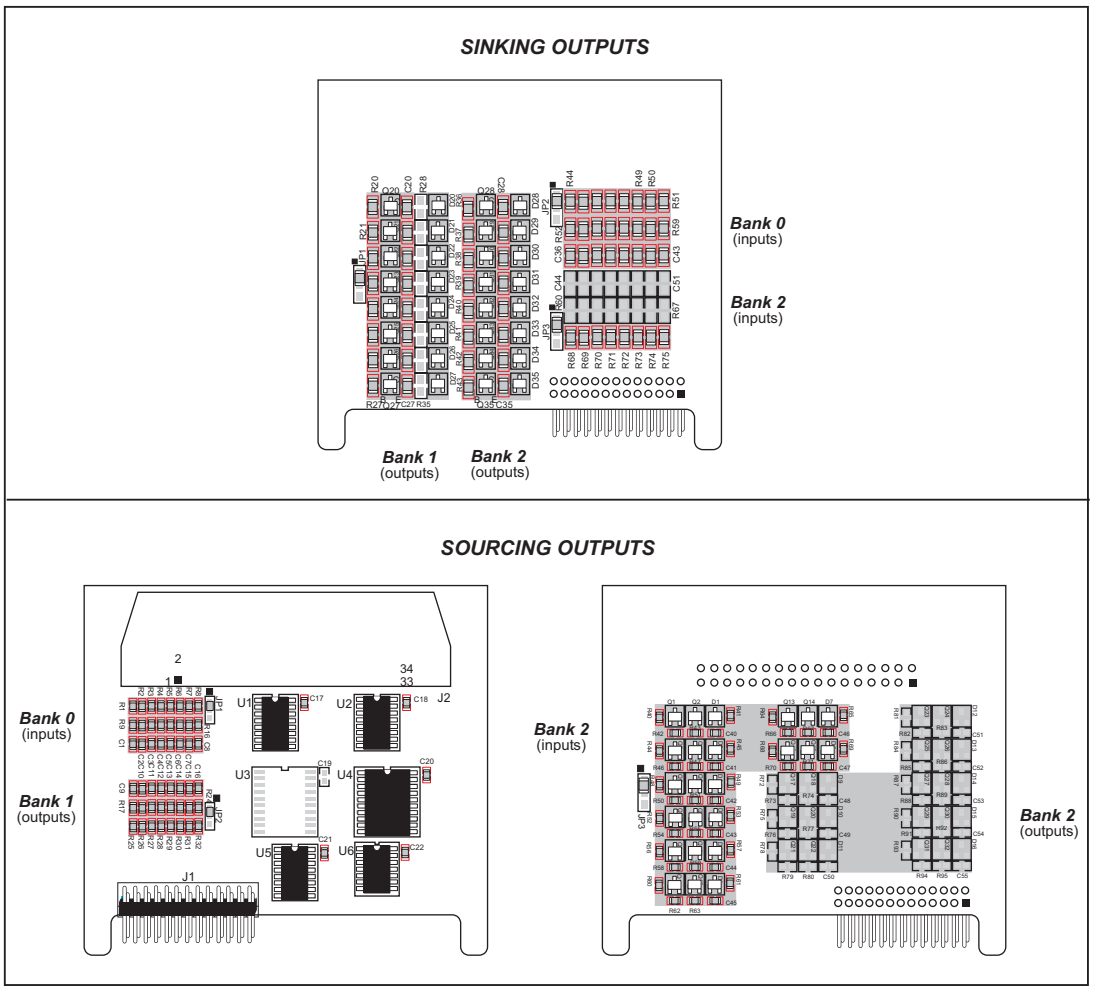



*Figure 26. Locations of Banks*

The I/O points on Bank 0 are always inputs, and the I/O points on Bank 1 are always outputs. The I/O points on Bank 2 were configured at the factory as either inputs *or* outputs, depending on the model of the Digital I/O Card. Table 10 lists the factory configurations.

***Table 10. Digital I/O Card Bank 2 Factory Configurations***

| Model | Bank 2 Configured As |
|---|---|
| SR9200 | Inputs |
| SR9210 | Sinking outputs |
| SR9220 | — |
| SR9205 | Inputs |
| SR9215 | Sourcing outputs |
| SR9225 | — |

The operation of Bank 2 is determined by the components on the Digital I/O Card. There is no jumper setting to select between inputs and outputs for Bank 2.

### 7.4.1 Digital Inputs

Table 11 provides the pinout configuration for the input points.

***Table 11. Digital Inputs Pinout***

| Pin | | Bank 0 | Pin | | Bank 2 |
|---|---|---|---|---|---|
| 2 | DIGIN0 | IN0 | 13 | I/O8 | IN8 |
| 3 | DIGIN1 | IN1 | 14 | I/O9 | IN9 |
| 4 | DIGIN2 | IN2 | 15 | I/O10 | IN10 |
| 5 | DIGIN3 | IN3 | 16 | I/O11 | IN11 |
| 8 | DIGIN4 | IN4 | 18 | I/O12 | IN12 |
| 9 | DIGIN5 | IN5 | 19 | I/O13 | IN13 |
| 10 | DIGIN6 | IN6 | 20 | I/O14 | IN14 |
| 11 | DIGIN7 | IN7 | 21 | I/O15 | IN15 |

The protected digital inputs, shown in Figure 27, are factory configured with 10 kΩ pull-up resistors. Digital I/O cards are also available in quantity with the protected digital inputs pulled down as shown in Figure 27.

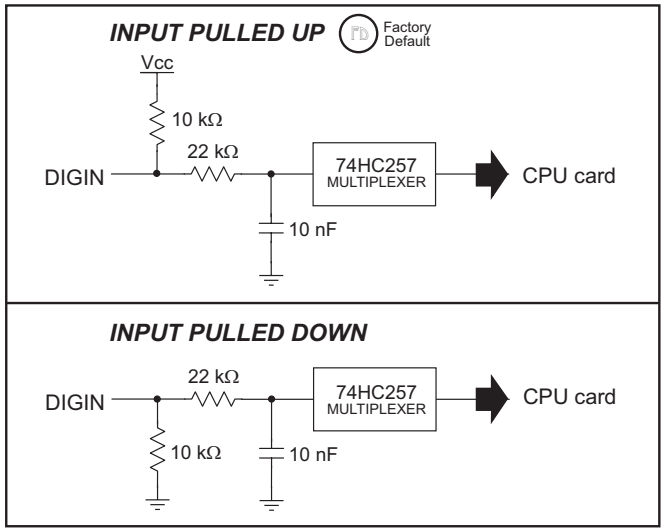



***Figure 27. Protected Digital Inputs***

A 0 Ω surface-mount resistor is used as a jumper to select whether the inputs are pulled up or down, as shown in Figure 28.

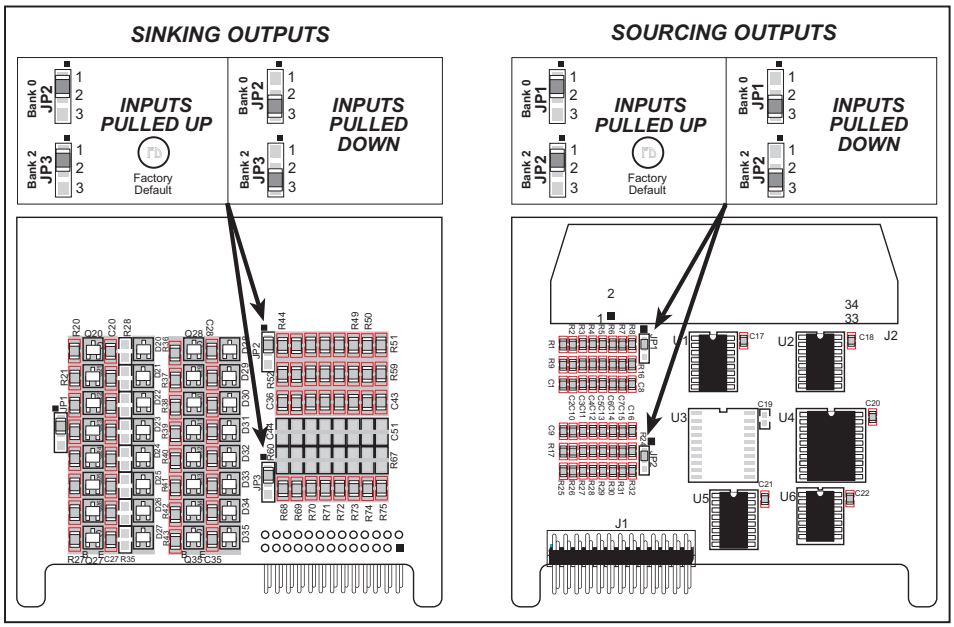



***Figure 28. Selecting Pulled Up or Pulled Down Digital Inputs***

The digital inputs are able to operate continuously from -30 V to +30 V, and have a logic threshold of 2.5 V. They are protected against spikes up to ±48 V.

### 7.4.2 Digital Outputs

The high-current digital outputs are either sinking or sourcing, depending on the model of the Digital I/O Card. Table 12 provides the pinout configuration for the output points.

***Table 12. Digital Outputs Pinout***

| Pin | Bank 2 | Pin | Bank 1 |
|---|---|---|---|
| 13 I/O8 | OUT8 | 22 HVOUT0 | OUT0 |
| 14 I/O9 | OUT9 | 23 HVOUT1 | OUT1 |
| 15 I/O10 | OUT10 | 24 HVOUT2 | OUT2 |
| 16 I/O11 | OUT11 | 26 HVOUT3 | OUT3 |
| 18 I/O12 | OUT12 | 27 HVOUT4 | OUT4 |
| 19 I/O13 | OUT13 | 29 HVOUT5 | OUT5 |
| 20 I/O14 | OUT14 | 30 HVOUT6 | OUT6 |
| 21 I/O15 | OUT15 | 31 HVOUT7 | OUT7 |

Figure 29 shows the power distribution on the Digital I/O Card.

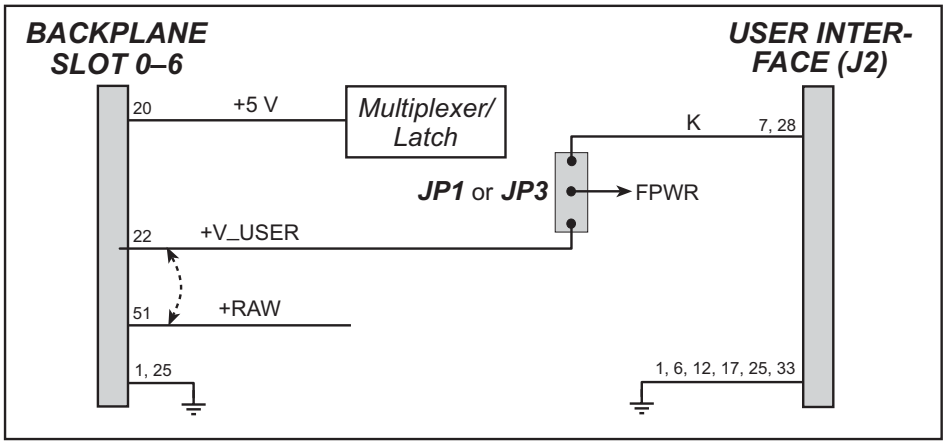



***Figure 29. Digital I/O Card Power Distribution***

When designing your interface with the Smart Star system, you need to establish whether you will use the **+V_USER/+RAW** supply on the backplane or your own independent **K** supply to drive the high-current outputs. The selection of this **FPWR** power supply is implemented via a 0 Ω surface-mount resistor on header JP1 (sinking outputs) or header JP3 (sourcing outputs) as shown in Figure 30. The factory default is to use **+V_USER/+RAW**, but Digital I/O Cards are available in quantity with the **FPWR** power supply jumpered to your own independent **K** supply.

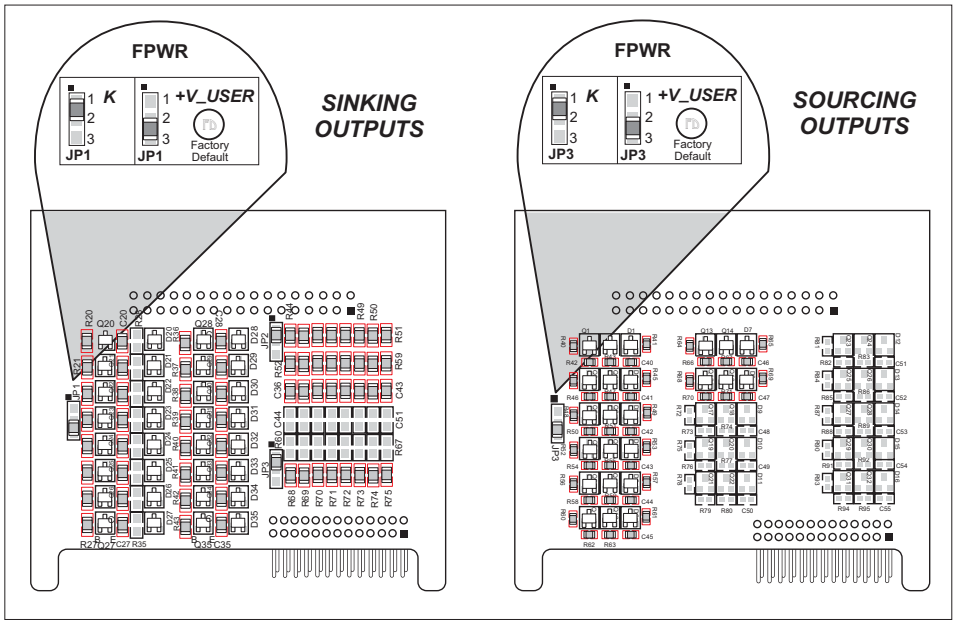



***Figure 30. Selecting Power Supply for High-Current Sinking or Sourcing Outputs***

Figure 31 shows how to connect a load to the high-current outputs based on whether your Digital I/O Card model has sinking or sourcing outputs.

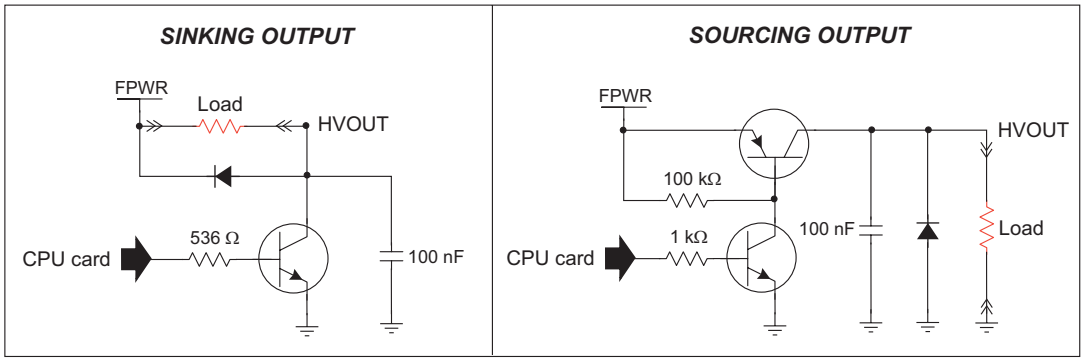



***Figure 31. Connecting a Load to the High-Current Outputs***

Each high-current output is able to sink or source up to 200 mA continuously, with a load limit of 40 V. Each high-current output may be switched independently, or a whole bank may be switched at once. The total current draw should be kept below 2.0 A when all high-current outputs on one Digital I/O Card are operating simultaneously, and the total current draw from your **+V_USER/+RAW** supply for all the I/O cards should be kept below 7.0 A.

> **NOTE:** Note that the power supply provided in the Smart Star Tool Kit has a maximum output of 1.1 A.

## 7.5 Software

### 7.5.1 Sample Programs

- **`SSTARIO.C`**—Demonstrates digital I/O using individual channels and whole banks. The sample program is set up for 8 inputs and 16 outputs. If necessary, you may change the macros in the sample program to match your Digital I/O Card.

#### 7.5.1.1 Running Sample Programs

To run a sample program, open it with the **File** menu (if it is not still open), compile it using the **Compile** menu, and then run it by selecting **Run** in the **Run** menu. The CPU Card must be connected to a PC using the programming cable as described in Section 2.3, "Programming Cable Connections."

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

### 7.5.2 Dynamic C Libraries

The **`SMRTSTAR`** directory contains libraries required to operate the Smart Star control system.

- **`SMRTSTAR.LIB`**—This library supports all the functions needed by the Smart Star systems including Digital I/O Cards, Relay Cards, D/A Converter and A/D Converter Cards, and serial communication.

Other functions applicable to all devices based on the Rabbit 2000 microprocessor are described in the ***Dynamic C Function Reference Manual***.

### 7.5.3 Smart Star Digital I/O Card Function Calls

## `int digIn(int channel);`

Reads the state of a digital input channel (IN0–IN15, IN8–IN15 is not available on all versions of the Digital I/O Card).

**PARAMETER**

**`channel`** is the digital input channel to read. **`channel`** should be passed as

```
channel = (slotnumber * 128) + (channelnumber)
```

or

```
channel = ChanAddr(slotnumber, channelnumber)
```

where **`slotnumber`** is 0–6, and **`channelnumber`** is 0–15.

**RETURN VALUE**

The state of the digital input channel, 0 or 1.

**SEE ALSO**

**`digBankIn, digOut, digBankOut`**

## `int digBankIn(int bank);`

Reads the state of Bank 0 or Bank 2 (if installed) digital input channels—Bank 0 consists of IN0–IN7 and Bank 2 consists of IN8–IN15.

**PARAMETER**

**`bank`** is the bank of digital input channels to read. **`bank`** should be passed as

```
bank = (slotnumber * 16) + (banknumber)
```

or

```
bank = BankAddr(slotnumber, banknumber)
```

where **`slotnumber`** is 0–6, and **`banknumber`** is 0 or 2.

**RETURN VALUE**

An input value in the lower byte, where each bit corresponds to one channel.

**SEE ALSO**

**`digIn, digOut, digBankOut`**

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

Writes a value to an output channel (OUT0–OUT15, OUT8–IN15 not available on all versions of the Digital I/O Card).

### PARAMETERS

**channel** is the digital output channel to write. **channel** should be passed as

```
channel = (slotnumber * 128) + (channelnumber)
```

or

```
channel = ChanAddr(slotnumber, channelnumber)
```

where **slotnumber** is 0–6, and **channelnumber** is 0–15.

**value** is the output value, 0 or 1.

### RETURN VALUE

None.

### SEE ALSO

**digBankOut, digIn,** digBankIn

## int digBankOut(int bank, int value);

Writes a byte value to Bank 1 or Bank 2 (if installed) digital output channels—Bank 1 consists of OUT0–OUT7 and Bank 2 consists of OUT8–OUT15.

### PARAMETER

**bank** is the bank of digital output channels to write. **bank** should be passed as

```
bank = (slotnumber * 16) + (banknumber)
```

or

```
bank = BankAddr(slotnumber, banknumber)
```

where **slotnumber** is 0–6, and **banknumber** is 1 or 2.

**value** is the output value, where each bit corresponds to one channel.

### RETURN VALUE

An input value in the lower byte, where each bit corresponds to one channel.

### SEE ALSO

digOut, digIn, digBankIn

## 7.6 Electrical and Mechanical Specifications

Figure 32 shows the mechanical dimensions for the Digital I/O Card.

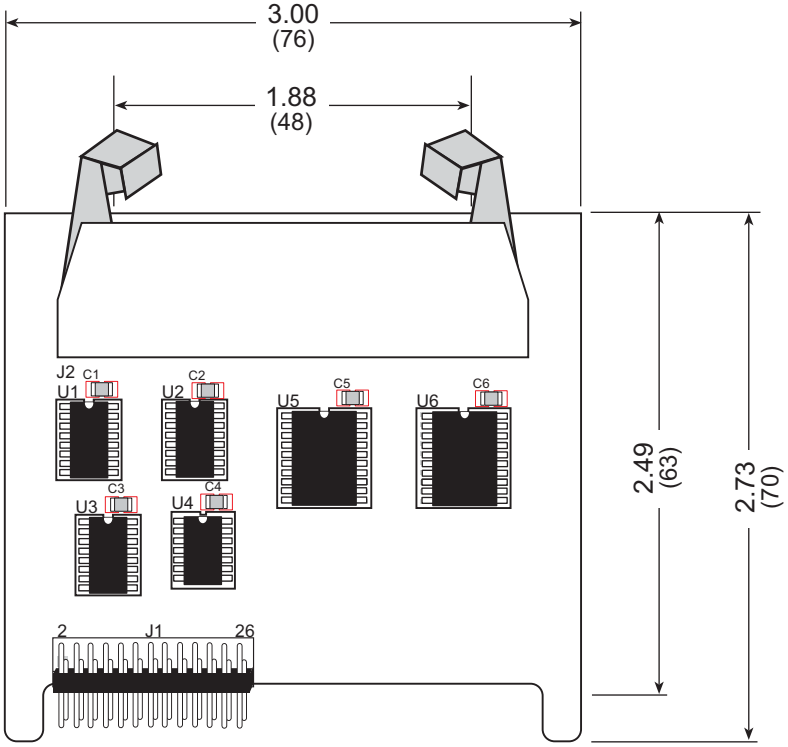



***Figure 32. Digital I/O Card Dimensions***

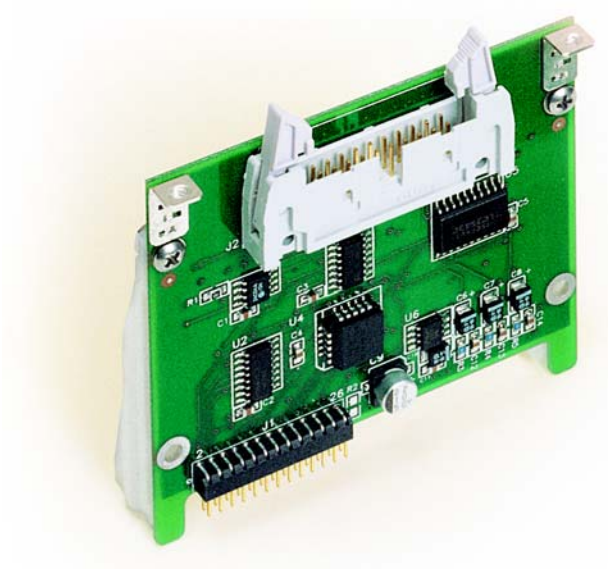
**NOTE:** All diagram and graphic measurements are in inches followed by millimeters enclosed in parentheses.

Table 13 lists the electrical, mechanical, and environmental specifications for the Digital I/O Card.

***Table 13. Digital I/O Card Specifications***

| Parameter | Specification |
|---|---|
| Board Size | 2.73" × 3.00" × 0.44"<br>(70 mm × 76 mm × 11 mm) |
| Connectors | one 2 × 17 latch/eject ribbon connector, 0.1 inch pitch |
| Operating Temperature | –40°C to +70°C |
| Humidity | 5% to 95%, noncondensing |
| Power Requirements | 5 V DC at 65 mA from backplane (+5 V supply)<br>9 V to 30 V DC for **+RAW/+V_USER** from backplane or 9 V to 30 V DC for K on user interface header J2<br>Maximum draw 2.0 A from **+RAW/+V_USER** on backplane |
| Digital Inputs | Continuous operation from -30 V to +30 V, logic threshold at 2.5 V, protected against spikes ±48 V, 10 kΩ pull-up/pull-down resistors |
| Digital Outputs | Each output can sink (source) up to 200 mA continuously with load limit of 40 V, each output may be switched independently or bank of eight may be switched all at once, load current supplied from **+RAW/+V_USER** on backplane or user-supplied **K** on user interface header J2 |

# PART III. A/D CONVERTER CARDS



# 8. A/D CONVERTER CARDS

Chapter 8 describes the features of the A/D Converter Card, one of the I/O cards designed for the Smart Star embedded control system.

The Smart Star is a modular and expandable embedded control system whose configuration of I/O, A/D Converter, D/A Converter, and Relay Cards can be tailored to a large variety of demanding real-time control and data acquisition applications.

The typical Smart Star system consists of a rugged backplane with a power supply, a CPU card, and one or more I/O cards. The CPU card plugs into a designated slot on the backplane chassis, which has seven additional slots available for I/O cards to be used in any combination. A high-performance Rabbit 2000 microprocessor on the CPU card provides fast data processing.

## 8.1 A/D Converter Card Features

Three models of A/D Converter Cards are available, as shown in Table 14.

***Table 14. Smart Star A/D Converter Cards***

| I/O Card | Model | Features |
|---|---|---|
| A/D Converter | SR9300 | 12-bit A/D converter, 11 channels, 0 V – 10 V |
| | SR9310 | 12-bit A/D converter, 11 channels, -10 V – +10 V |
| | SR9320 | 12-bit A/D converter, 11 channels, 4 mA – 20 mA |

## 8.2 User Interface

Figure 33 shows the circuit used to condition the analog signal before it goes to the A/D converter chip. Depending on the model of A/D Converter Card you have, it is designed to handle analog inputs of 0 V to 10 V, -10 V to +10 V, or 4–20 mA. The two different voltage ranges are handled with different gain resistors, $R_g$: 23.7 kΩ for the SR9300 and 12.1 kΩ for the SR9310. The input shown in Figure 33 is configured differently for the SR9320, which handles analog inputs of 4–20 mA.

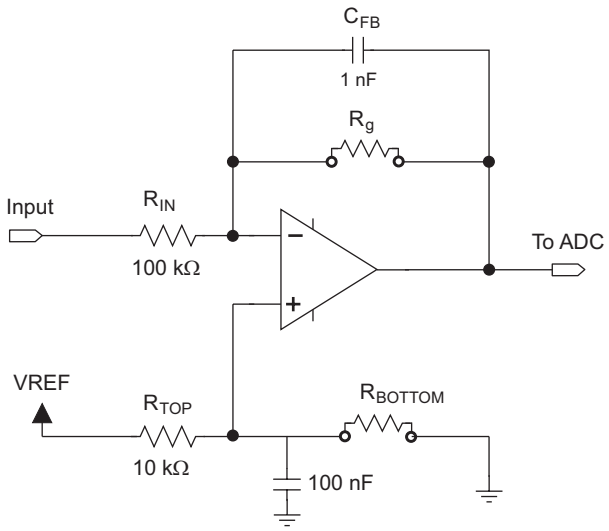



***Figure 33. Analog Input Amplifier Circuit***

The TLC2543 A/D converter chip on the A/D Converter Card uses synchronous Serial Port B and Timer A5 on the Rabbit 2000 to do the A/D conversions.

Figure 34 shows the complete pinout for the user interface on header J2. Note that pin 1 is indicated by a small arrow on the ribbon cable connector.

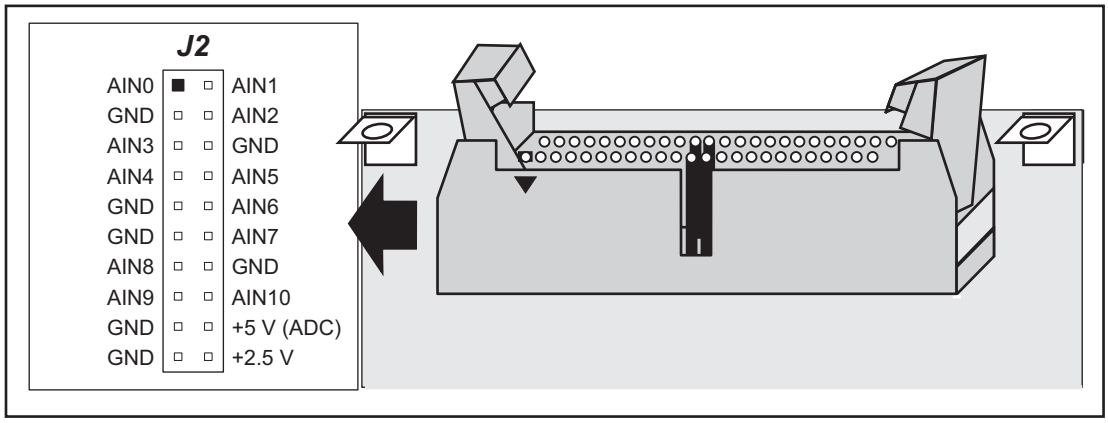
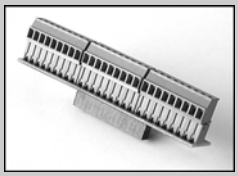
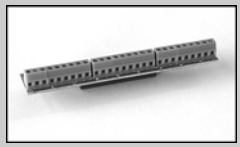



***Figure 34. A/D Converter Card User Interface Pinout***

## 8.3 User FWT Connections

Connections to the A/D Converter Cards are made via a ribbon cable connector or optional field wiring terminals that are either pluggable or have screw terminals. Table 15 lists the Rabbit part numbers for the FWTs.

***Table 15. Guide to FWT Selection***

| FWT Description | I/O Cards | Rabbit Part Number | |
|---|---|---|---|
| | | Pluggable Terminals | Screw Terminals |
| | |  |  |
| FWT18 | A/D Converter | 101-0421 | 101-0515 |

### 8.3.1 Pinouts

Figure 35 shows the pinout for the FWTs used on the A/D Converter Cards.

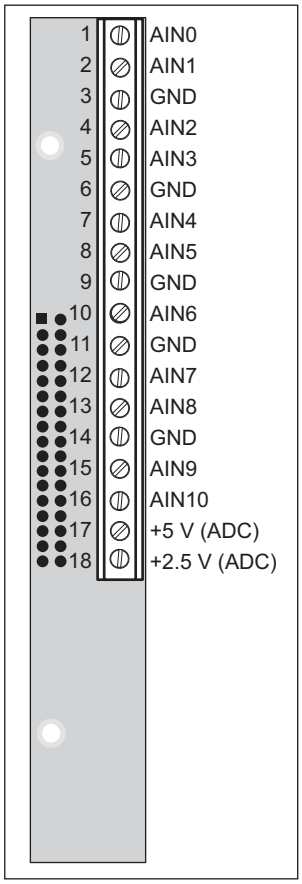



***Figure 35. FWT Pinout for A/D Converter Cards***

## 8.4 Power Distribution

Figure 36 shows the power distribution on the A/D Converter Card.

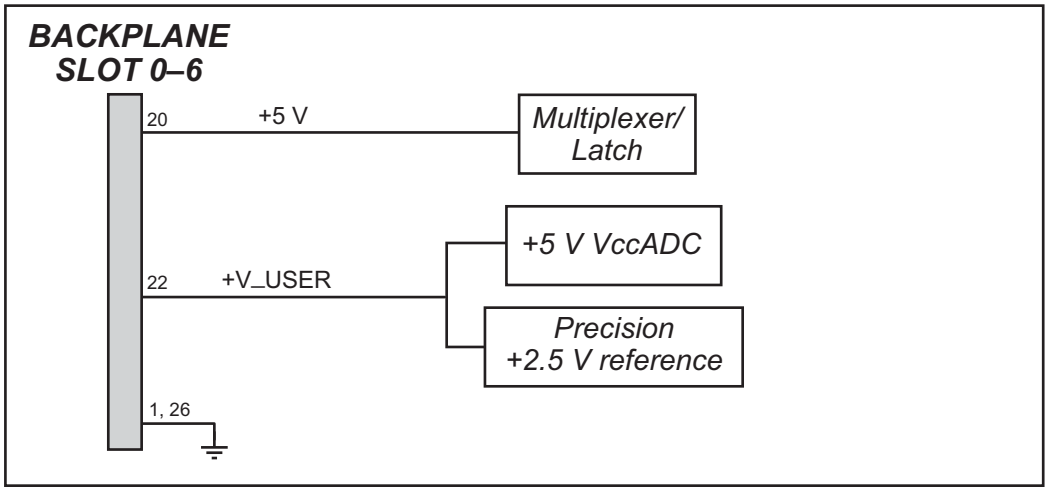



***Figure 36. A/D Converter Card Power Distribution***

## 8.5 Software

### 8.5.1 Sample Programs

- **`SSTARAD1.C`**—Demonstrates how to calibrate an A/D converter channel using two known voltages, and defines the two coefficients, gain and offset. These coefficients are then read back to compute the equivalent voltage.
- **`SSTARAD2.C`**—Reads and displays voltage and equivalent values of each A/D converter channel. Calibrations must have been previously stored into flash memory before running this program. See sample program **`SSTARAD3.C`**.
- **`SSTARAD3.C`**—Demonstrates how to calibrate all A/D converter channels using two known voltages and defines the two coefficients, gain and offset. These coefficients are then read back to compute the equivalent voltage and are saved to flash memory.

#### 8.5.1.1 Running Sample Programs

To run a sample program, open it with the **File** menu (if it is not still open), compile it using the **Compile** menu, and then run it by selecting **Run** in the **Run** menu. The CPU Card must be connected to a PC using the programming cable as described in Section 2.3, "Programming Cable Connections."

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

### 8.5.2 Dynamic C Libraries

The **`SMRTSTAR`** directory contains libraries required to operate the Smart Star control system.

- **`SMRTSTAR.LIB`**—This library supports all the functions needed by the Smart Star systems including Digital I/O Cards, Relay Cards, D/A Converter and A/D Converter Cards, and serial communication.

Other functions applicable to all devices based on the Rabbit 2000 microprocessor are described in the ***Dynamic C Function Reference Manual***.

### 8.5.3 Smart Star A/D Converter Card Function Calls

## int anaInEERd(int channel);

The A/D Converter Card calibration constants, gain, and offset are stored in the factory in the upper half of the EEPROM on the A/D Converter Card. Use this function to read the A/D Converter Card calibration constants, gain, and offset from the upper half of the EEPROM on the A/D Converter Card.

**PARAMETERS**

**channel** is the analog input channel. **channel** should be passed as

```
channel = (slotnumber * 128) + (channelnumber)
```

where **slotnumber** is 0–6, and **channelnumber** is 0–10

or

```
channel = ChanAddr(slotnumber, channelnumber)
```

where **slotnumber** is 0–6, and **channelnumber** is 0–10.

**RETURN VALUE**

0 if successful.
-1—control command unacceptable.
-2—EEPROM address unacceptable.

**SEE ALSO**

**anaInEEWr**

## int anaSaveCalib();

The calibration constants may also be saved in the flash memory on the Smart Star CPU Card. Doing so will speed up A/D conversions since a memory access from flash memory will be faster than from EEPROM. Use **anaSaveCalib** to save the current set of calibration constants for the analog input and output channels in the Smart Star flash memory. The calibration constants stored in flash memory can then be accessed at any time with the **anaLoadCalib** function.

If the factory-set calibration are not used, customer-measured calibration constants should first be established using the **anaInCalib** function.

**RETURN VALUE**

None.

**SEE ALSO**

**anaLoadCalib, anaInCalib**

## void anaLoadCalib();

Reads a complete set of calibration constants for the analog input and output channels from the Smart Star flash memory on the CPU Card. These should be set using the **anaInCalib** or **anaInEERd** function, then saved to flash memory using the **anaSaveCalib** function.

**RETURN VALUE**

None.

**SEE ALSO**

**anaSaveCalib, anaInCalib**

## int anaInCalib(int channel, int value1, float volt1, int value2, float volt2);

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

### PARAMETERS

**channel** is the A/D converter input channel (0–10). **channel** should be passed as

```
channel = (slotnumber * 128) + (channelnumber)
```

where **slotnumber** is 0–6, and **channelnumber** is 0–10

or

```
channel = ChanAddr(slotnumber, channelnumber)
```

where **slotnumber** is 0–6, and **channelnumber** is 0–10.

**value1** is the first A/D converter value.

**volt1** is the voltage/current corresponding to the first A/D converter value. Current values entered as milliamps will produce milliamp values, and amp values entered will produce amp values.

**value2** is the second A/D converter value.

**volt2** is the voltage/current corresponding to the second A/D converter value. Current values entered as milliamps will produce milliamp values, and amp values entered will produce amp values.

### RETURN VALUE

0 if successful,
-1, if not able to make calibration constants.

### SEE ALSO

**anaIn, anaInVolts**

## int anaInEEWr(int channel);

Writes the calibration constants, gain, and offset to the upper half of the EEPROM on the A/D Converter Card.

### PARAMETERS

**channel** is the analog input channel. **channel** should be passed as

```
channel = (slotnumber * 128) + (channelnumber)
```

where **slotnumber** is 0–6, and **channelnumber** is 0–10

or

```
channel = ChanAddr(slotnumber, channelnumber)
```

where **slotnumber** is 0–6, and **channelnumber** is 0–10.

### RETURN VALUE

0 if successful.
-1—control command unacceptable.
-2—EEPROM address unacceptable.
-3—data value unacceptable.

### SEE ALSO

**anaInEERd, _anaInEEWr**

## unsigned int anaIn(unsigned int channel);

Reads the state of an analog input channel and converts it to a digital value. A timeout occurs, causing the function to exit, if the end of the conversion is not detected within 13 µs.

### PARAMETERS

**channel** is the analog input channel to read. **channel** should be passed as

```
channel = (slotnumber * 128) + (channelnumber)
```

where **slotnumber** is 0–6, and **channelnumber** is 0–10

or

```
channel = ChanAddr(slotnumber, channelnumber)
```

where **slotnumber** is 0–6, and **channelnumber** is 0–10.

### RETURN VALUE

A value corresponding to the voltage on the analog input channel, 0–4095. A value outside this range indicates a failure

### SEE ALSO

**anaInCalib, anaInVolts**

## int anaInVolts(int channel);

Reads the state of an analog input channel and uses the previously set calibration constants to convert the state to volts.

### PARAMETERS

**channel** is the analog input channel. **channel** should be passed as

```
channel = (slotnumber * 128) + (channelnumber)
```

where **slotnumber** is 0–6, and **channelnumber** is 0–10

or

```
channel = ChanAddr(slotnumber, channelnumber)
```

where **slotnumber** is 0–6, and **channelnumber** is 0–10.

### RETURN VALUE

A voltage value corresponding to the voltage on the analog input channel (0–+10 V on the SR9300 or -10–+10 V on the SR9310).

### SEE ALSO

**anaIn, anaInCalib, anaInmAmps**

## `float anaInmAmps(unsigned int channel);`

Reads the state of an analog input channel and uses the previously set calibration constants to convert the state to current.

**NOTE:** The factory-set calibration constants are for current measurements in amperes.

### PARAMETERS

**`channel`** is the analog input channel. **`channel`** should be passed as

```
channel = (slotnumber * 128) + (channelnumber)
```

where **`slotnumber`** is 0–6, and **`channelnumber`** is 0–10

or

```
channel = ChanAddr(slotnumber, channelnumber)
```

where **`slotnumber`** is 0–6, and **`channelnumber`** is 0–10.

### RETURN VALUE

A current value corresponding to the 4–20 mA (0.004–0.020 A) current on the analog input channel.

### SEE ALSO

**`anaIn, anaInCalib, anaInVolts`**

## 8.6 Electrical and Mechanical Specifications

Figure 37 shows the mechanical dimensions for the A/D Converter Card.

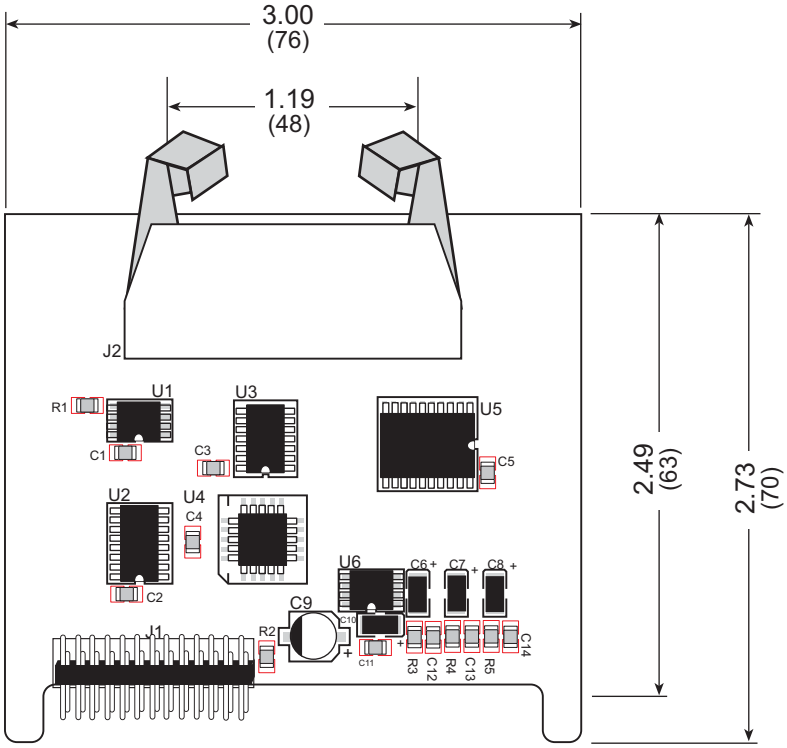



***Figure 37. Relay Card Dimensions***

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

Table 16 lists the electrical, mechanical, and environmental specifications for the A/D Converter Card.

***Table 16. A/D Converter Card Specifications***

| Parameter | Specification |
|---|---|
| Board Size | 2.73" × 3.00" × 0.44"<br>(70 mm × 76 mm × 11 mm) |
| Connectors | one 2 × 10 latch/eject ribbon connector, 0.1 inch pitch |
| Operating Temperature | –40°C to +70°C |
| Humidity | 5% to 95%, noncondensing |
| Power Requirements | 5 V DC at 40 mA from backplane (+5 V supply)<br>9 V to 30 V DC, 35 mA at 24 V DC, **+RAW/+V_USER** from backplane |
| Number of Inputs | 11 conditioned channels |
| Analog Input Ranges* | 0 V to +10 V (max. ±22 V DC)<br>–10 V to +10 V (max. ±40 V DC)<br>4 mA to 20 mA (max. 30 mA) |
| Resolution | 12 bits (0–4095) |
| Conversion Time (including Dynamic C) | 0.13 ms/channel (includes 0.08 ms/channel for raw count) |
| Repeatability | Typical ±½ count, maximum ±1 count @ –20°C to +70°C<br>Typical ±1 count, maximum ±2 counts @ –40°C to –20°C |
| Accuracy | Typical ±1 count, maximum ±2 counts @ 25°C<br>±4 counts @ –40°C and +70°C† |
| Input Impedance | SR9300 (0 V to +10 V): 100 kΩ min.<br>SR9310 (–10 V to +10 V): 100 kΩ min.<br>SR9320 (4 mA to 20 mA): 249 Ω ± 1% |
| Linearity Error (end to end) | ±1 count |

* The A/D Converter Card is protected against transients that might exceed the maximum ratings.

† Accuracy at temperature extremes can be improved by recalibrating the A/D Converter Card at the temperature it will be used at.

# PART IV.  D/A CONVERTER CARDS

# 9. D/A CONVERTER CARDS

Chapter 9 describes the features of the D/A Converter Card, one of the I/O cards designed for the Smart Star embedded control system.

The Smart Star is a modular and expandable embedded control system whose configuration of I/O, A/D Converter, D/A Converter, and Relay Cards can be tailored to a large variety of demanding real-time control and data acquisition applications.

The typical Smart Star system consists of a rugged backplane with a power supply, a CPU card, and one or more I/O cards. The CPU card plugs into a designated slot on the backplane chassis, which has seven additional slots available for I/O cards to be used in any combination. A high-performance Rabbit 2000 microprocessor on the CPU card provides fast data processing.

## 9.1 D/A Converter Card Features

Three models of D/A Converter Cards are available, as shown in Table 17.

***Table 17. Smart Star D/A Converter Cards***

| I/O Card | Model | Features |
|---|---|---|
| D/A Converter | SR9400 | 12-bit D/A converter, 8 channels, 0 V – 10 V |
| | SR9410 | 12-bit D/A converter, 8 channels, -10 V – +10 V |
| | SR9420 | 12-bit D/A converter, 8 channels, 4 mA – 20 mA |

## 9.2 User Interface

Figure 38 shows the D/A converter circuit. A buffer, U6, buffers the data signals D0–D7 from the Smart Star backplane, and sends them to the D/A converter, U2–U5. Signals D2–D5 are used to switch the chip select line to identify which D/A converter will perform the conversion. The model of D/A Converter Card determines the analog output ranges (0 V to 10 V, -10 V to +10 V, or 4–20 mA). The different voltage or current ranges are handled with different feedback resistors, as shown in Figure 38. A switching regulator provides a regulated power supply for the op-amps.

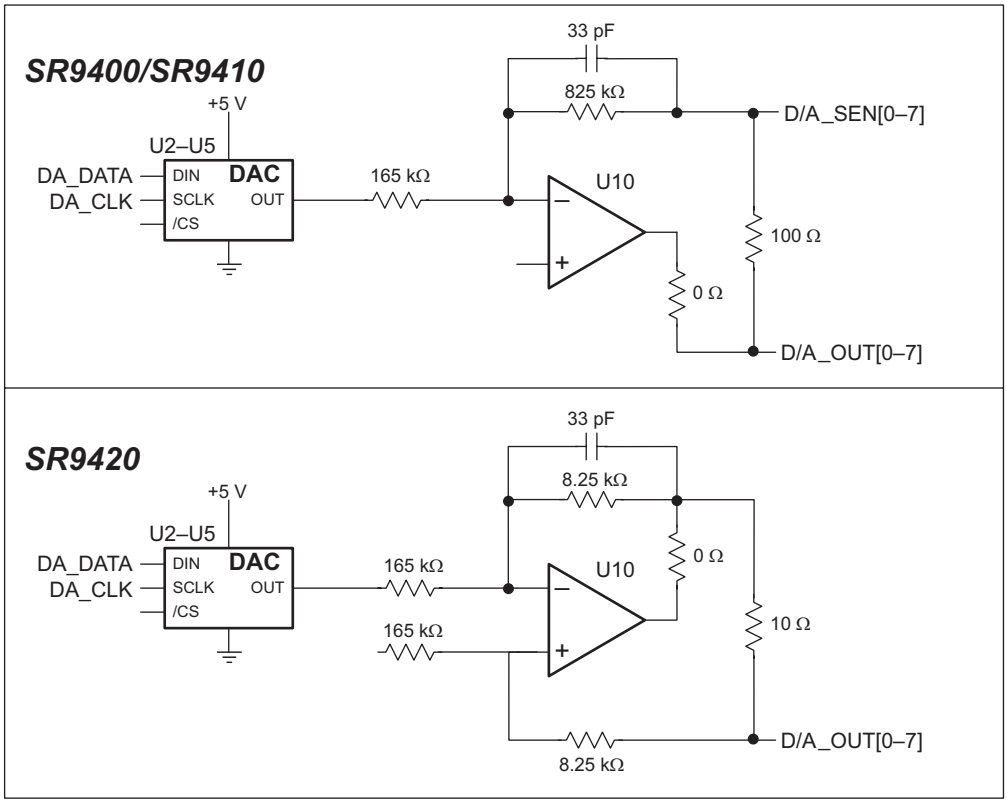



***Figure 38. D/A Converter Card Circuit***

**NOTE:** The **D/A_SEN[0–7]** sensing inputs are not used when using the current source version (model SR9420) of the D/A Converter Card.

Figure 39 shows the complete pinout for the user interface on header J1. Note that pin 1 is indicated by a small arrow on the ribbon cable connector.

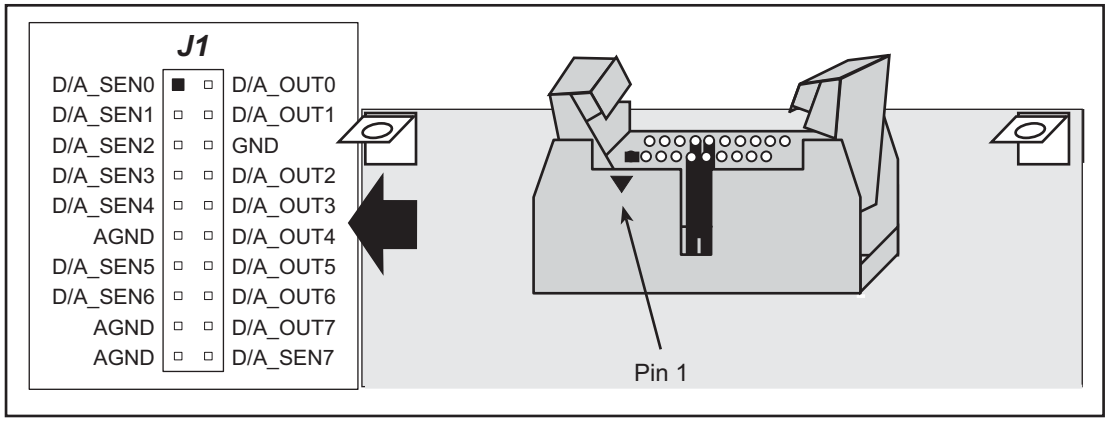



***Figure 39. D/A Converter Card User Interface Pinout***

The D/A Converter Card has eight analog output channels, **D/A_OUT[0–7]**, and is also equipped with a remote sensing capability through sensing inputs **D/A_SEN[0–7]** for the voltage-amplifier versions of the D/A Converter Card (models SR9400 and SR9410). These sensing inputs compensate for the voltage drop across the wire leads of low-impedance loads to provide a more precise output across the load.

Let's look at Figure 40 to see how this happens. Assume the load is 500 Ω. If the impedance of the wire used to connect the load to the output terminal on the D/A Converter Card is 5 Ω, there will be a voltage drop of about 5 Ω/500 Ω = 1% across the wire. The voltage across the load will then be 1% less, which is about 40 counts for the SR9400. By connecting **D/A_SEN** as shown in Figure 40, the output driver will be able to sense the voltage drop across the wire and provide a more accurate voltage output across the load. If the load impedance is much greater than the impedance of the wire leads, simply leave the **D/A_SEN** sensing inputs open.

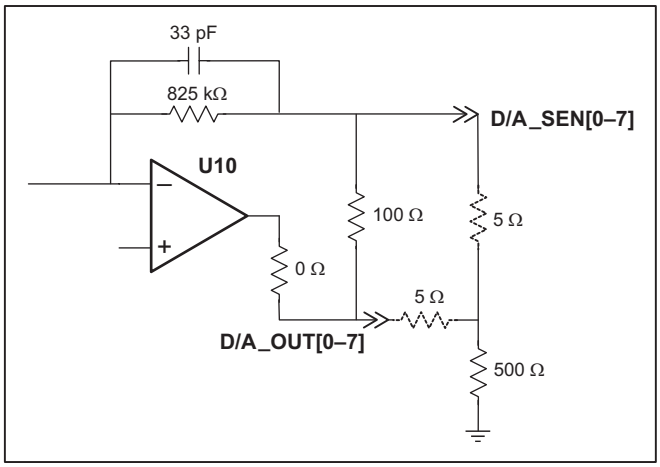
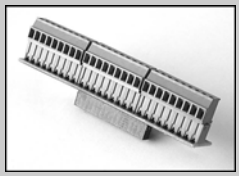
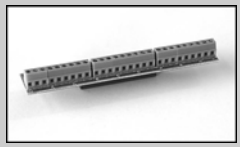



***Figure 40. D/A Converter Output for Low-Impedance Loads***

## 9.3 User FWT Connections

Connections to the D/A Converter Cards are made via a ribbon cable connector or optional field wiring terminals that are either pluggable or have screw terminals. Table 18 lists the Rabbit part numbers for the FWTs.

***Table 18. Guide to FWT Selection***

| FWT Description | I/O Cards | Rabbit Part Number | |
|---|---|---|---|
| | | Pluggable Terminals | Screw Terminals |
| | |  |  |
| FWT18 | D/A Converter | 101-0421 | 101-0515 |

### 9.3.1 Pinouts

Figure 41 shows the pinout for the FWTs used on the D/A Converter Cards.

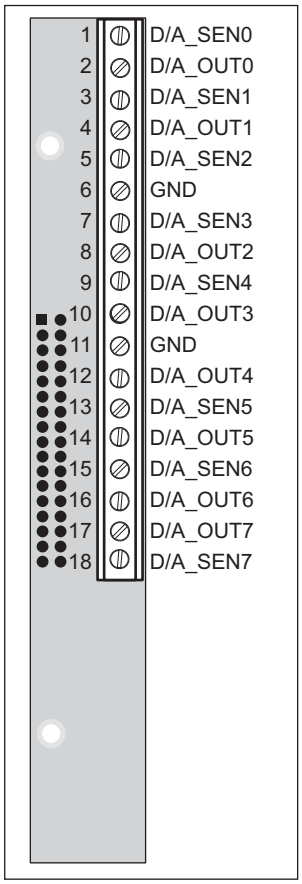



***Figure 41. FWT Pinout for D/A Converter Cards***

## 9.4 Power Distribution

Figure 42 shows the power distribution on the D/A Converter Card.

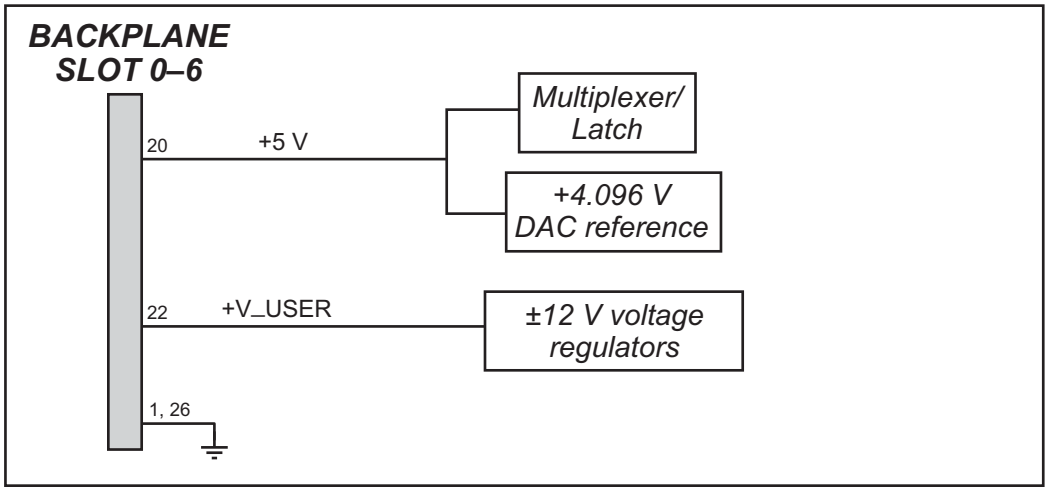



***Figure 42. D/A Converter Card Power Distribution***

Figure 43 shows the power supply for the op-amps used as voltage amplifiers/current sources.

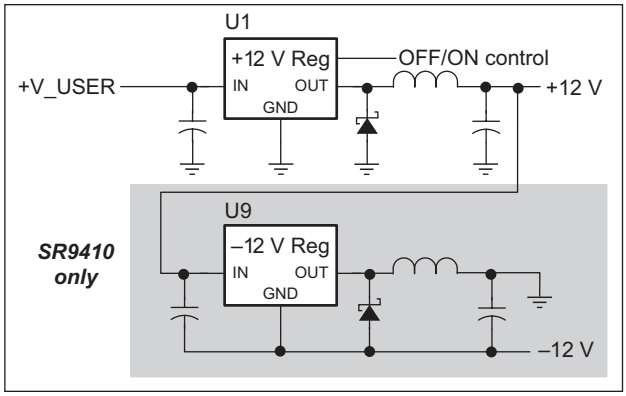



***Figure 43. Op-Amp Power Supplies***

There is provision in software using the **`anaOutDisable`** or the **`anaOutEnable`** function calls to turn the regulated ±12 V power supply off or on since pin 5 on U1 is connected to PE7 on the Rabbit 2000 microprocessor on the backplane. This type of disabling/enabling allows the analog output channels to float in a high-impedance state.

The voltage regulator on/off is disabled by default when there is a reset or when the D/A Converter Card is first used. All output channels must be configured to the required voltage or current outputs before calling the **`anaOutEnable`** function since unconfigured channels are automatically set to the maximum output.

The –12 V supply is provided only for the SR9410, which provides analog outputs up to ±10 V.

## 9.5 Software

### 9.5.1 Sample Programs

- **`ANAVOUT.C`**—Demonstrates how to set the D/A channel for the desired output.
- **`SSDAC1.C`**—Demonstrates how to recalibrate a D/A converter channel using two known voltages, and shows how to define the two coefficients, gain and offset, that will be rewritten into the D/A Converter Card's EEPROM.
- **`SSDAC2.C`**—Demonstrates how to recalibrate a D/A converter channel using an A/D Converter Card andtwo known voltages. Shows how to define the two coefficients, gain and offset, that will be rewritten into the D/A Converter Card's EEPROM.
- **`SSDAC3.C`**—Demonstrates how to recalibrate a D/A converter channel using two known currents, and shows how to define the two coefficients, gain and offset, that will be rewritten into the D/A Converter Card's EEPROM.
- **`SSDAC4.C`**—Demonstrates how to recalibrate a D/A converter channel using an A/D Converter Card,two known currents. Shows how to define the two coefficients, gain and offset, that will be rewritten into the D/A Converter Card's EEPROM.

#### 9.5.1.1 Running Sample Programs

To run a sample program, open it with the **File** menu (if it is not still open), compile it using the **Compile** menu, and then run it by selecting **Run** in the **Run** menu. The CPU Card must be connected to a PC using the programming cable as described in Section 2.3, “Programming Cable Connections.”

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

### 9.5.2 Dynamic C Libraries

The **`SMRTSTAR`** directory contains libraries required to operate the Smart Star control system.

- **`SMRTSTAR.LIB`**—This library supports all the functions needed by the Smart Star systems including Digital I/O Cards, Relay Cards, A/D Converter and D/A Converter Cards, and serial communication.

Other functions applicable to all devices based on the Rabbit 2000 microprocessor are described in the ***Dynamic C Function Reference Manual***.

### 9.5.3 Smart Star D/A Converter Card Function Calls

## `void anaOutDisable(void);`

Turns off (disables) voltage regulator for output-channel op-amps on ***all*** D/A Converter Cards, leaving all output channels in a high-impedance state.

**RETURN VALUE**

None.

**See Also**

**`anaOutEnable, anaOut, anaOutVolts, anaOutmAmps`**

## `void anaOutEnable(void);`

Turns on (enables) voltage regulator for output-channel op-amps on ***all*** D/A Converter Cards.

> **NOTE:** The voltage regulator on/off is disabled (off) at power-up or reset. All output channels must be configured to the required voltage or current outputs before calling the **`anaOutEnable`** function since unconfigured channels will be set automatically to the maximum output.

**RETURN VALUE**

None.

**SEE ALSO**

**`anaOutDisable, anaOut, anaOutVolts, anaOutmAmps`**

## `int anaOutEERd(int channel);`

The D/A Converter Card calibration constants, gain, and offset are stored in the factory in the upper half of the EEPROM on the D/A Converter Card. Use this function to read the D/A Converter Card calibration constants into the global table **`_dacCalib`**

**PARAMETERS**

**`channel`** is the D/A converter output channel. **`channel`** should be passed as

```
channel = (slotnumber * 128) + (channelnumber)
```

where **`slotnumber`** is 0–6, and **`channelnumber`** is 0–7

or

```
channel = ChanAddr(slotnumber, channelnumber)
```

where **`slotnumber`** is 0–6, and **`channelnumber`** is 0–7.

**RETURN VALUE**

0 if successful.
–1—control command unacceptable.
–2—EEPROM address unacceptable.

**SEE ALSO**

**`anaOutEEWr`**

```
int anaOutCalib(int channel, int value1,
   float voltamp1, int value2, float voltamp2);
```

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

**PARAMETERS**

**channel** is the D/A converter output channel. **channel** should be passed as

```
channel = (slotnumber * 128) + (channelnumber)
```

where **slotnumber** is 0–6, and **channelnumber** is 0–7

or

```
channel = ChanAddr(slotnumber, channelnumber)
```

where **slotnumber** is 0–6, and **channelnumber** is 0–7.

**value1** is the first D/A conversion data point. Use a value near 4095 to produce a lower output measurement.

**voltamp1** is the voltage/current measurement corresponding to the first D/A converter value. Current values entered as milliamps will produce milliamp values, and amp values entered will produce amp values.

**value2** is the second D/A conversion data point. Use a value near 0 to produce a higher output measurement.

**voltamp2** is the voltage/current measurement corresponding to the second D/A converter value. Current values entered as milliamps will produce milliamp values, and amp values entered will produce amp values.

| rawcount | Approximate Output Equivalent | | |
|---|---|---|---|
| | SR9400 | SR9410 | SR9420 |
| 0 (0000H) | +10 V | +10 V | 20 mA |
| 2047 (07FFH) | +5 V | 0 V | 12 mA |
| 4095 (0FFFH) | 0 V | –10 V | 4 mA |

**RETURN VALUE**

0 if successful.
–1 if not able to make calibration constants.

**SEE ALSO**

**anaOut, anaOutVolts, anaOutmAmps**

## int anaSaveCalib(int boardtype);

The calibration constants may also be saved in the flash memory on the Smart Star CPU Card. Doing so will speed up D/A conversions since a memory access from flash memory will be faster than from EEPROM. Use **anaSaveCalib** to save the current set of calibration constants for the analog input or output channels in the Smart Star flash memory. The calibration constants stored in flash memory can then be accessed at any time with the **anaLoadCalib** function.

Calibration constants should first be established using **anaOutCalib** or obtained via **anaOutEERd**.

### PARAMETER

**boardtype** is the type of board, which is 0 for the D/A Converter Card, 1 for the A/D Converter Card.

### RETURN VALUE

0 if successful.
–1—attempt to write non-flash area, nothing written.
–2—**rootSrc** not in root.
–3—timeout while writing flash memory.
–4—attempt to write to ID block sector(s).

### SEE ALSO

**anaLoadCalib, anaOutCalib**

## int anaLoadCalib(int boardtype);

Reads a complete set of calibration constants for the analog output channels from the Smart Star flash memory on the CPU Card. These should have been loaded to the flash memory with the **anaSaveCalib** function.

### PARAMETER

**boardtype** is the type of board, which is 0 for the D/A Converter Card, 1 for the A/D Converter Card.

### RETURN VALUE

0 if successful.
–1—attempt to read from non-flash area.
–2—destination not all in root.

### SEE ALSO

**anaSaveCalib, anaOutCalib**

```
int anaOut(unsigned int channel,
   unsigned int rawcount);
```

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

- Program bits (D15…D12)
- New data (D11…D0)

| D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R1 | SPD | PWR | R0 | MSB | | | | 12 data bits MSB–LSB (0–4095) | | | | | | | LSB |

SPD—Speed control bit: 1 = fast mode (default), 0 = slow mode

PWR—Power control bit: 1 = power down, 0 = normal operation (default)

The following table lists all the possible combinations of the register-selects bits R1 (Register 1) and R0 (Register 0)

| R1 | R0 | Register |
|---|---|---|
| 0 | 0 | Write data to D/A converter channel B |
| 0 | 1 | Write data to buffer |
| 1 | 0 | Write data to D/A converter channel A |
| 1 | 1 | Reserved |

**PARAMETERS**

**`channel`** is the D/A converter output channel to write. **`channel`** should be passed as

```
channel = (slotnumber * 128) + (channelnumber)
```

where **`slotnumber`** is 0–6, and **`channelnumber`** is 0–7

or

```
channel = ChanAddr(slotnumber, channelnumber)
```

where **`slotnumber`** is 0–6, and **`channelnumber`** is 0–7.

**`rawcount`** is a value corresponding to the voltage on the analog output channel (0–4095). The following **`rawcount`** data correspond to the analog outputs indicated.

| rawcount | Approximate Output Equivalent | | |
|---|---|---|---|
| | SR9400 | SR9410 | SR9420 |
| 0 (0000H) | +10 V | +10 V | 20 mA |
| 2047 (07FFH) | +5 V | 0 V | 12 mA |
| 4095 (0FFFH) | 0 V | –10 V | 4 mA |

**RETURN VALUE**

0 if successful.
–1 if **`rawcount`** is greater than 4095.

**SEE ALSO**

**`anaOutVolts, anaOutCalib`**

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

Sets the voltage of an analog output channel by using the previously set calibration constants to calculate correct data values.

### PARAMETERS

**channel** is the D/A converter output channel. **channel** should be passed as

```
channel = (slotnumber * 128) + (channelnumber)
```

where **slotnumber** is 0–6, and **channelnumber** is 0–7

or

```
channel = ChanAddr(slotnumber, channelnumber)
```

where **slotnumber** is 0–6, and **channelnumber** is 0–7.

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

### RETURN VALUE

None.

### SEE ALSO

**anaOut, anaOutCalib, anaOutmAmps**

## void anaOutmAmps(unsigned int channel, float current);

Sets the current of an analog output channel by using the previously set calibration constants to calculate correct data values.

**NOTE:** The factory-set calibration constants are for current measurements in amperes.

### PARAMETERS

**channel** is the D/A converter output channel. **channel** should be passed as

```
channel = (slotnumber * 128) + (channelnumber)
```

where **slotnumber** is 0–6, and **channelnumber** is 0–7

or

```
channel = ChanAddr(slotnumber, channelnumber)
```

where **slotnumber** is 0–6, and **channelnumber** is 0–7.

**current** is the current range (4–20 mA or 0.004–0.020 A) desired on the output channel.

### RETURN VALUE

0 if successful.

–1 if not able to make calibration constants.

### SEE ALSO

**anaOut, anaOutVolts, anaOutCalib**

## int anaOutEEWr(int channel);

Writes the calibration constants, gain, and offset to the upper half of the EEPROM on the D/A Converter Card.

### PARAMETERS

**channel** is the D/A converter output channel. **channel** should be passed as

```
channel = (slotnumber * 128) + (channelnumber)
```

where **slotnumber** is 0–6, and **channelnumber** is 0–7

or

```
channel = ChanAddr(slotnumber, channelnumber)
```

where **slotnumber** is 0–6, and **channelnumber** is 0–7.

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

### RETURN VALUE

0 if successful.
-1—control command unacceptable.
-2—EEPROM address unacceptable.
-3—data value unacceptable.

### SEE ALSO

**anaOutEERd**

## 9.6 Electrical and Mechanical Specifications

Figure 44 shows the mechanical dimensions for the D/A Converter Card.

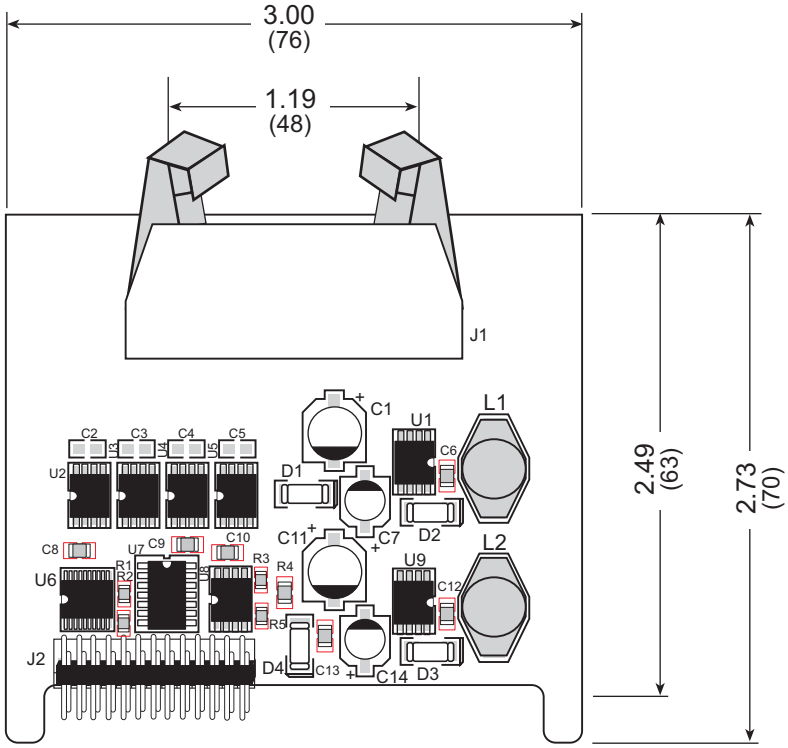



***Figure 44. D/A Converter Card Dimensions***

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

Table 19 lists the electrical, mechanical, and environmental specifications for the D/A Converter Card.

***Table 19. D/A Converter Card Specifications***

| Parameter | Specification |
|---|---|
| Board Size | 2.73" × 3.00" × 0.44"<br>(70 mm × 76 mm × 11 mm) |
| Connectors | one 2 × 10 latch/eject ribbon connector, 0.1 inch pitch |
| Operating Temperature | –40°C to +70°C |
| Humidity | 5% to 95%, noncondensing |
| Power Requirements | 5 V DC at 50 mA typical from backplane (+5 V supply)<br>15 V to 30 V DC, 30 mA at 24 V DC, **+RAW/+V_USER** from backplane |
| Number of Outputs | 8 channels |
| Analog Output Ranges | SR9400: 0 V to +10 V, 20 mA/channel (maximum)<br>SR9410: –10 V to +10 V, 20 mA/channel (maximum)<br>SR9420: 4 mA to 20 mA, 10 V (maximum) |
| Resolution | 12 bits (0–4095) |
| Conversion Time (including Dynamic C) | 0.2 ms/channel |
| Output Stability | ±½ count |
| Output Impedance | SR9400: < 1 Ω,<br>SR9410: < 1 Ω,<br>SR9420: > 100 kΩ |

# PART V. RELAY CARDS

# 10. RELAY CARDS

Chapter 10 describes the features of the Relay Card, one of the I/O cards designed for the Smart Star embedded control system.

The Smart Star is a modular and expandable embedded control system whose configuration of I/O, A/D Converter, D/A Converter, and Relay Cards can be tailored to a large variety of demanding real-time control and data acquisition applications.

The typical Smart Star system consists of a rugged backplane with a power supply, a CPU card, and one or more I/O cards. The CPU card plugs into a designated slot on the backplane chassis, which has seven additional slots available for I/O cards to be used in any combination. A high-performance Rabbit 2000 microprocessor on the CPU card provides fast data processing.

## 10.1 Relay Card Features

Two models of Relay Cards are available, as shown in Table 20.

***Table 20. Smart Star Relay Cards***

| I/O Card | Model | Features |
|---|---|---|
| Relay | SR9500 | 5 SPST relays and 1 SPDT relay, each protected with onboard snubbers |
| | SR9510 | 8 SPDT relays (no snubbers) |

The SR9500 Relay Cards are suitable for switching all kinds of loads up to 30 V DC at 1 A or 48 V AC at 0.5 A. The SR9510 handles similar loads, but is restricted to noninductive loads unless you add snubbers to the system that is interfacing with the Smart Star.

## 10.2 User Interface

Depending on the model of Relay Card (see Table 20), the relays on the Relay Card will be configured as SPDT or SPST with or without snubbers. Figure 45 shows these relay configurations.

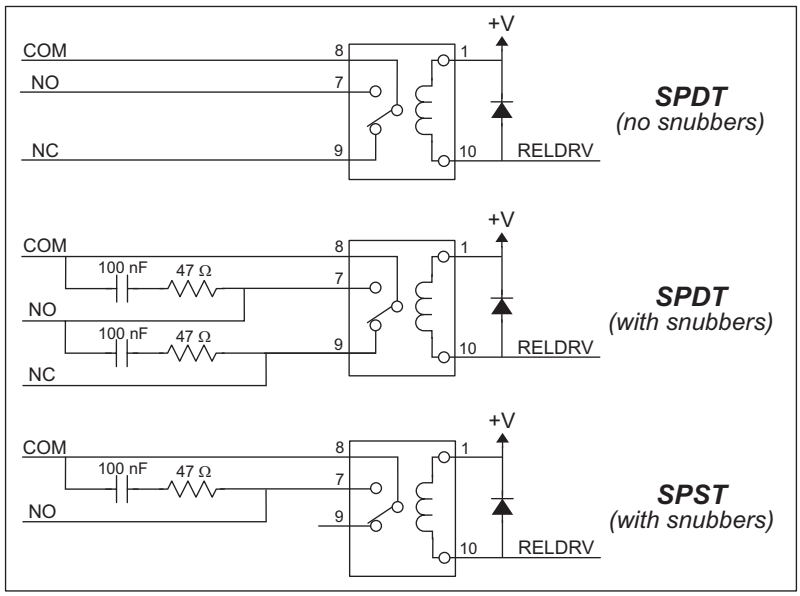



***Figure 45. Relay Configurations***

The diode protects the coil power supply (and the Smart Star backplane) from inductive spikes caused by energizing/de-energizing the coil, and the resistor-capacitor snubbers protect the relay contacts against voltage spikes induced by inductive loads.

Figure 46 shows the complete pinout for the user interface on header J1. Note that pin 1 is indicated by a small arrow on the ribbon cable connector.

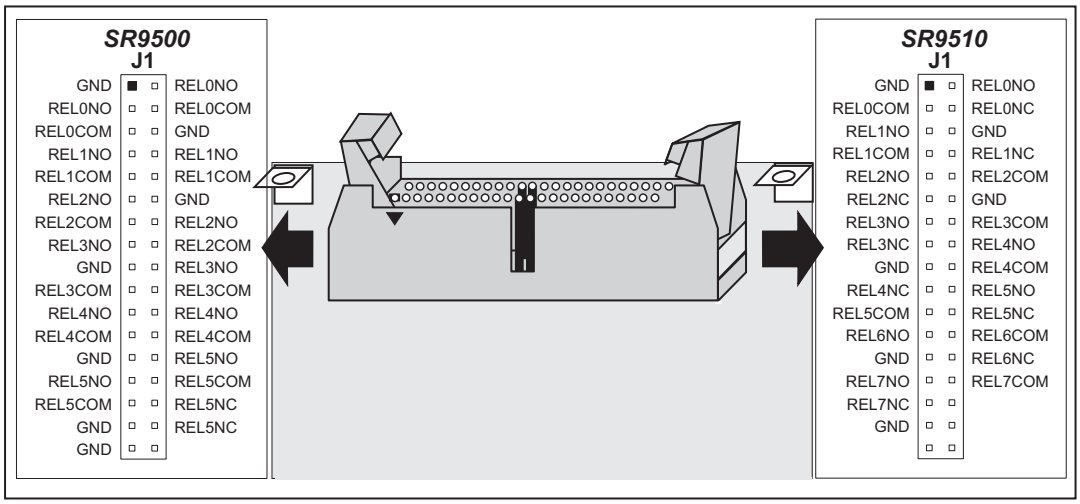
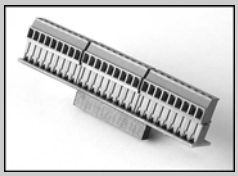
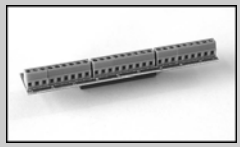



***Figure 46. Relay Card User Interface Pinout***

## 10.3 User FWT Connections

Connections to the Relay Cards are made via a ribbon cable connector or optional field wiring terminals that are either pluggable or have screw terminals. Table 21 lists the Rabbit part numbers for the FWTs.

***Table 21. Guide to FWT Selection***

| FWT Description | I/O Cards | Rabbit Part Number | |
|---|---|---|---|
| | | Pluggable Terminals | Screw Terminals |
| | |  |  |
| FWT18R | Relay (SR9500) | 101-0422 | 101-0516 |
| FWT27 | Relay (SR9510) | 101-0420 | 101-0514 |

### 10.3.1 Pinouts

Figure 47 shows the pinout for the FWTs used on the Relay Cards.

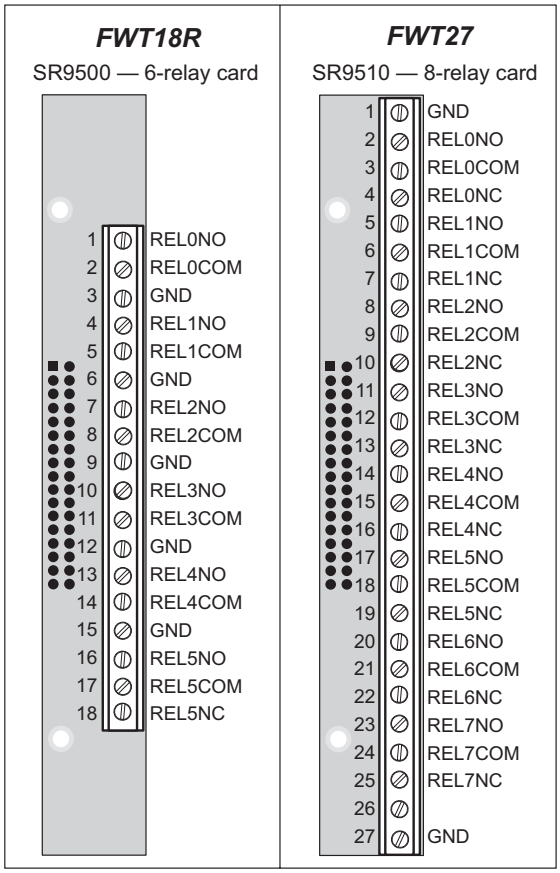



***Figure 47. FWT Pinouts for Relay Cards***

## 10.4 Power Distribution

Figure 48 shows the power distribution on the Relay Card.

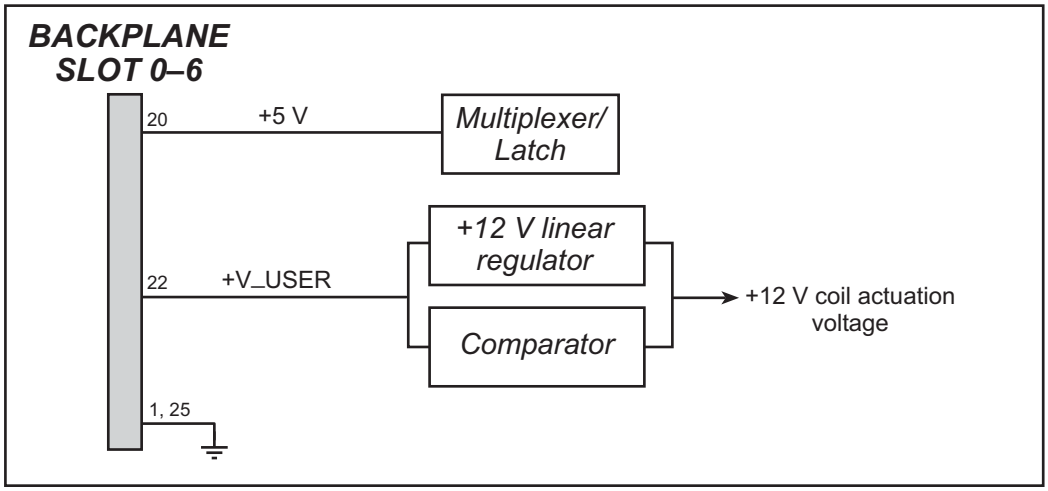



***Figure 48. Relay Card Power Distribution***

The relay coil actuation voltage is 12 V, and so **+V_USER** should be 12 V to 30 V DC. The **+V_USER** supply passes through a linear regulator and comparator, which are in parallel. The comparator is set for approximately +13.9 V, and as long as **+V_USER** is more than +13.9 V, the +12 V from the linear regulator will provide the coil actuation voltage. Should **+V_USER** be less than +13.9 V, the comparator will supply **+V_USER** directly to provide the coil actuation voltage.

## 10.5 Relay Cards Software

### 10.5.1 Sample Programs

- **`SSTARRLY.C`**—Demonstrates turning a relay on the Relay Card on and off.

### 10.5.2 Running Sample Programs

To run a sample program, open it with the **File** menu (if it is not still open), compile it using the **Compile** menu, and then run it by selecting **Run** in the **Run** menu. The CPU Card must be connected to a PC using the programming cable as described in Section 2.3, "Programming Cable Connections."

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

### 10.5.3 Dynamic C Libraries

The **`SMRTSTAR`** directory contains libraries required to operate the Smart Star control system.

- **`SMRTSTAR.LIB`**—This library supports all the functions needed by the Smart Star systems including Digital I/O Cards, Relay Cards, D/A Converter and A/D Converter Cards, and serial communication.

### 10.5.4 Smart Star Relay Card Function Calls

```
void relayOut(int relay, int value);
```

Sets the state of a relay.

**PARAMETER**

**`relay`** is the relay to set. **`relay`** should be passed as

```
relay = (slotnumber * 128) + (relaynumber)
```

or

```
relay = ChanAddr(slotnumber, relaynumber)
```

where **`slotnumber`** is 0–6, and **`relaynumber`** is 0–5 (SR9500) or 0–7 (SR9510), depending on the model of Relay Card.

**`value`** is the value to set the relay to, 0 or 1 (off or on).

## 10.6 Electrical and Mechanical Specifications

Figure 49 shows the mechanical dimensions for the Relay Card.

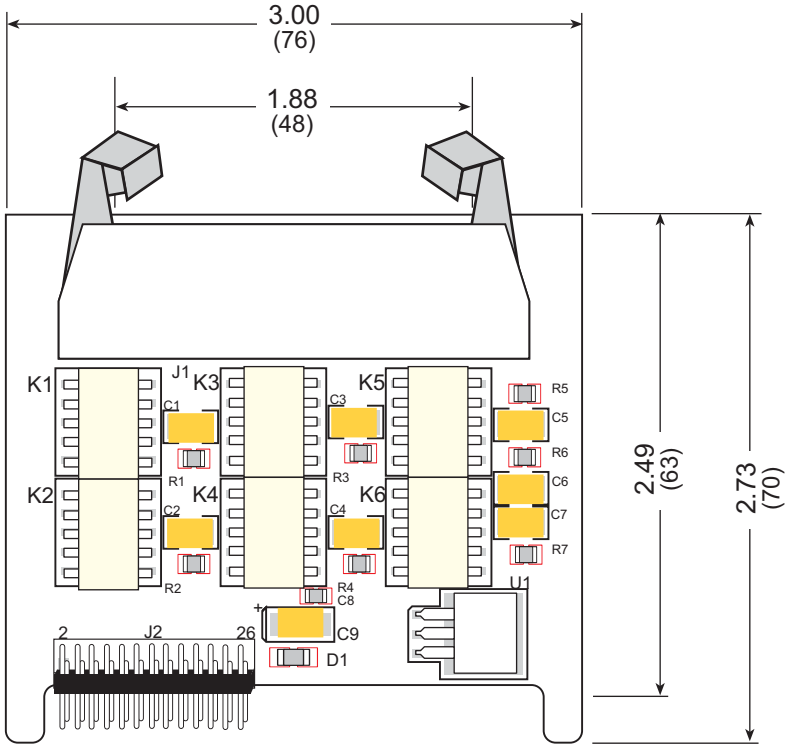



***Figure 49. Relay Card Dimensions***

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

Table 22 lists the electrical, mechanical, and environmental specifications for the Relay Card.

***Table 22. Relay Card Specifications***

| Parameter | Specification |
|---|---|
| Board Size | 2.73" × 3.00" × 0.44"<br>(70 mm × 76 mm × 11 mm) |
| Connectors | one 2 × 17 latch/eject ribbon connector, 0.1 inch pitch |
| Operating Temperature | –40°C to +70°C |
| Humidity | 5% to 95%, noncondensing |
| Power Requirements | 5 V DC at 10 mA from backplane (+5 V supply)<br>12 V to 30 V DC, 10 mA at 24 V DC, **+RAW/+V_USER** from backplane |
| Relay Switching Contacts | 30 V DC at 1 A or 48 V AC at 0.5 A |
| Relays | SR9500: 1 SPDT, 5 SPST (N.O., COM) with snubbers<br>SR9510: 8 SPDT (N.O., N.C., COM), no snubbers |

# PART VI. APPENDICES

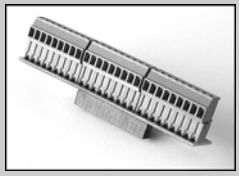
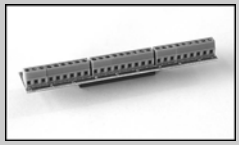
# APPENDIX A.
# FIELD WIRING TERMINALS

Appendix A explains how to prepare the connector on an I/O card to accept a field wiring terminal, and how to secure the field wiring terminal to the I/O card. The dimensions for the field wiring terminals are included.

## A.1 Selecting and Installing a Field Wiring Terminal

Connections to the I/O cards are made via a ribbon cable connector or optional field wiring terminals that are either pluggable or have screw terminals. Three different Field Wiring Terminals (FWTs) are available. Table A-1 lists the I/O cards and the Rabbit part numbers for the corresponding FWTs.

***Table A-1. Guide to FWT Selection***

| FWT Description | I/O Cards | Rabbit Part Number | |
|---|---|---|---|
| | | Pluggable Terminals  | Screw Terminals  |
| FWT27 | Digital I/O (SR9200 series)<br>Relay (SR9510) | 101-0420 | 101-0514 |
| FWT18 | A/D Converter (SR9300 series)<br>D/A Converter (SR9400 series) | 101-0421 | 101-0515 |
| FWT18R | Relay (SR9500) | 101-0422 | 101-0516 |

Before you can install the FWT you selected for your I/O card, you must remove the tabs from the connector on the I/O card. To do so, move the tab inwards as far as possible, as shown in Figure A-1. Then insert a screwdriver into the space below the tab on the side of the connector and gently nudge the tab up and out. If you are careful, the tab will remain intact to be saved and snapped back in place should you need to use a ribbon cable connector in the future.

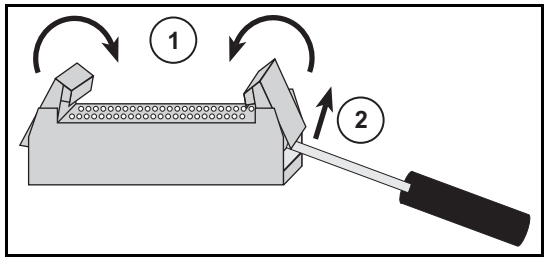



***Figure A-1. Remove Tabs from Connector on I/O Card***

Plug the FWT connector into the connector on the I/O card. Be sure to position the pluggable or screw connectors so that the header pins on the printed circuit board are towards you, as shown in Figure A-2. Position the mylar insulator above the FWT as shown in Figure A-2 to protect the header pins on the printed circuit board, and secure the FWT using the two 4-40 × ¼ screws supplied. them. Note that the mylar insulator will be bowed slightly once the screws are in place.

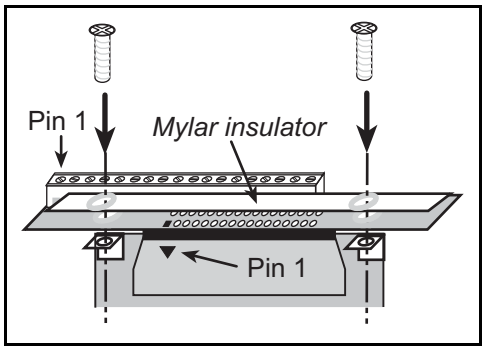



***Figure A-2. Secure FWT to I/O Card***

## A.2 Dimensions

Figure A-3 shows the overall FWT dimensions.

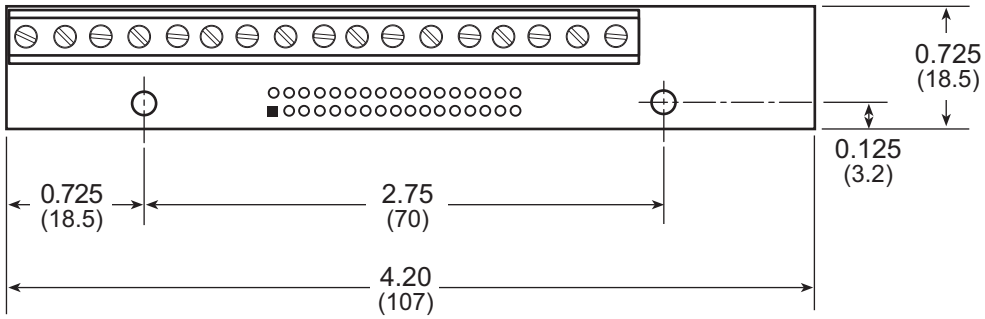



***Figure A-3. FWT Dimensions***

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

The actual appearance of the terminals may vary, depending on the number and type of terminals. The pinouts for the FWTs applicable to a particular I/O card are shown with the pinouts for the connectors on the individual I/O cards.

# APPENDIX B. LCD/KEYPAD MODULE

An optional LCD/keypad module with a NEMA 4 water-resistant bezel is available for the Smart Star. Appendix B describes the LCD/keypad module and provides the software function calls to make full use of the LCD/keypad module.

## B.1 Specifications

The LCD/keypad module comes with or without a panel-mounted NEMA 4 water-resistant bezel as shown in Figure B-1.

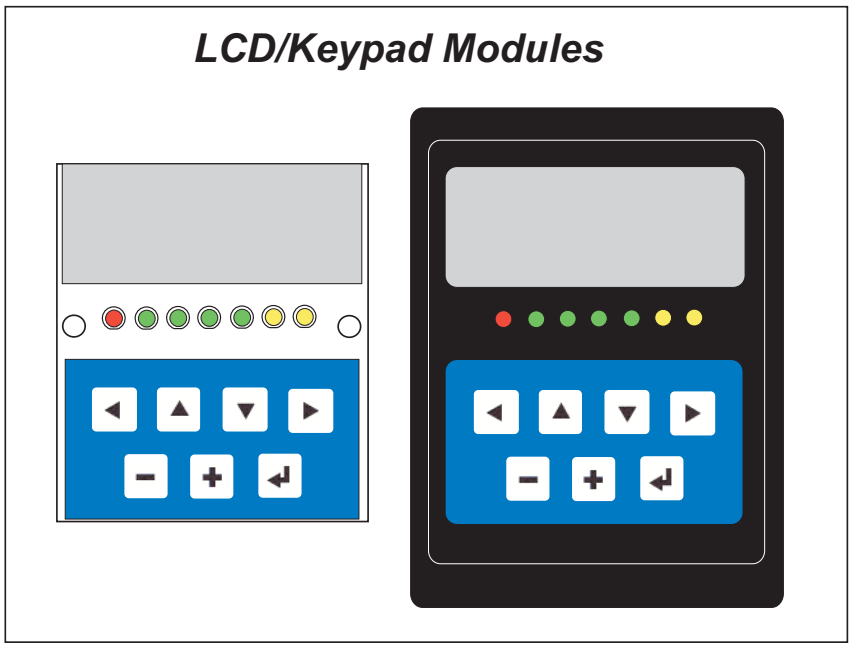



***Figure B-1. LCD/Keypad Module Versions***

Either version can be connected to the Smart Star backplane, and can be installed at a remote location up to 60 cm (24") away. Contact your Rabbit sales representative or your authorized distributor for further assistance in purchasing an LCD/keypad module.

Mounting hardware and a 60 cm (24") extension cable are also available for the LCD/keypad module through your sales representative or authorized distributor.

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

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

| Parameter | Specification |
|---|---|
| Board Size | 2.60" × 3.00" × 0.75"<br>(66 mm × 76 mm × 19 mm) |
| Bezel Size | 4.50" × 3.60" × 0.30"<br>(114 mm × 91 mm × 7.6 mm) |
| Temperature | Operating Range: 0°C to +50°C<br>Storage Range: –40°C to +85°C |
| Humidity | 5% to 95%, noncondensing |
| Power Consumption | 1.5 W maximum* |
| Connections | Connects to high-rise header sockets on Smart Star |
| LCD Panel Size | 122 × 32 graphic display |
| Keypad | 7-key keypad |
| LEDs | Seven user-programmable LEDs |

* The backlight adds approximately 650 mW to the power consumption.

The LCD/keypad module has 0.1" IDC headers at J1, J2, and J3 for physical connection to other boards or ribbon cables. Figure B-2 shows the LCD/keypad module footprint. These values are relative to one of the mounting holes.

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

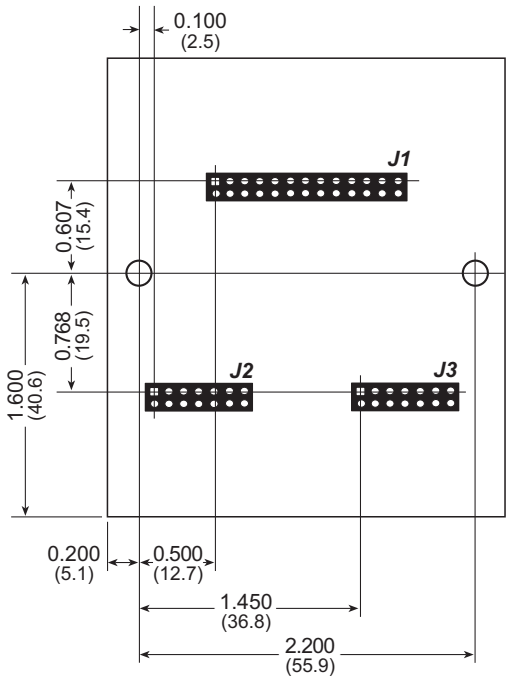



***Figure B-2. User Board Footprint for LCD/Keypad Module***

## B.2 Contrast Adjustments for All Boards

Starting in 2005, LCD/keypad modules were factory-configured to optimize their contrast based on the voltage of the system they would be used in. Be sure to select a KDU5V LCD/keypad module for use with the Smart Star — these modules operate at 5 V. You may adjust the contrast using the potentiometer at R2 as shown in Figure B-3. LCD/keypad modules configured for 3.3 V should not be used with the 5 V Smart Star because the higher voltage will reduce the backlight service life dramatically.

***LCD/Keypad Module Jumper Configurations***

| Header | Description | Pins Connected | Factory Default |
|---|---|---|---|
| J5 | 2.8 V | 1–2 | × |
| | 3.3 V | 3–4 | |
| | 5 V | n.c. | |

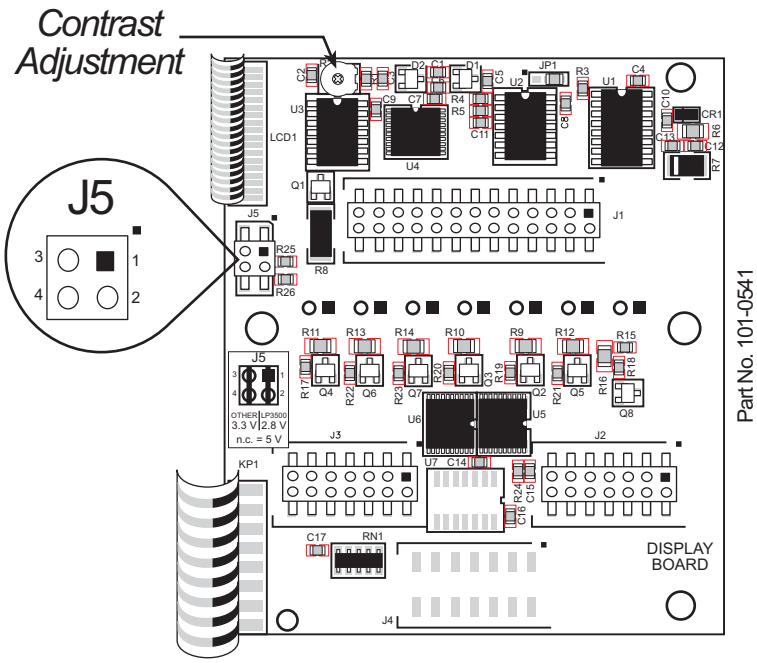



***Figure B-3. LCD/Keypad Module Voltage Settings and Contrast Adjustment***

You can set the contrast on the LCD display of pre-2005 LCD/keypad modules by adjusting the potentiometer at R2 or by setting the voltage for 5 V by removing the jumper that was installed at the factory across pins 1–2 on header J5 as shown in Figure B-3. Only one of these two options is available on these older LCD/keypad modules.

**NOTE:** Older LCD/keypad modules that do not have a header at J5 or a contrast adjustment potentiometer at R2 are limited to operate only at 5 V, and will work with the Smart Star. The older LCD/keypad modules are no longer being sold.

## B.3 Keypad Labeling

The keypad may be labeled according to your needs. A template is provided in Figure B-4 to allow you to design your own keypad label insert.

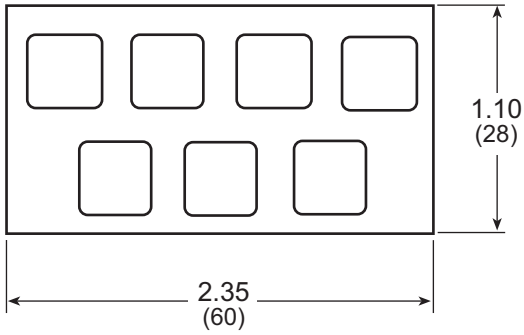



***Figure B-4. Keypad Template***

To replace the keypad legend, remove the old legend and insert your new legend prepared according to the template in Figure B-4. The keypad legend is located under the blue keypad matte, and is accessible from the left only as shown in Figure B-5.

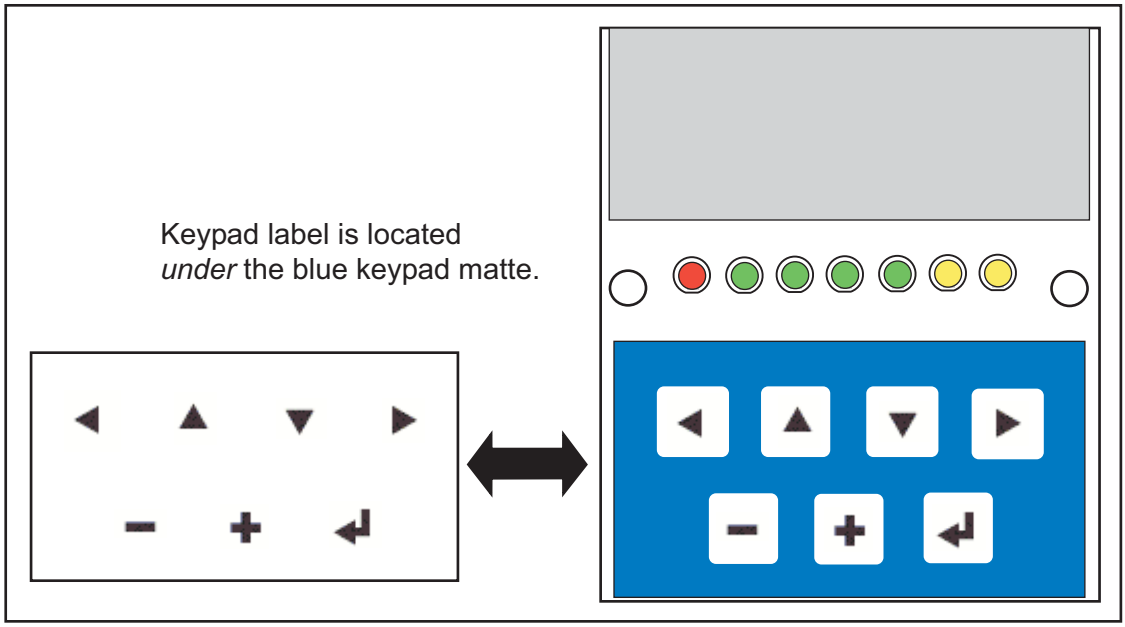



***Figure B-5. Removing and Inserting Keypad Label***

The sample program **`KEYBASIC.C`** in the **`122x32_1x7`** folder in **`SAMPLES\LCD_KEYPAD`** shows how to reconfigure the keypad for different applications.

## B.4 Header Pinouts

Figure B-6 shows the pinouts for the LCD/keypad module.

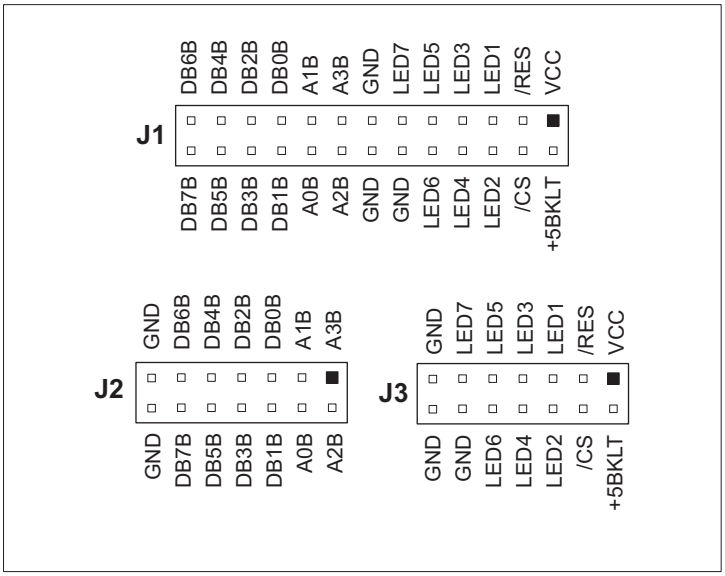



***Figure B-6. LCD/Keypad Module Pinouts***

### B.4.1 I/O Address Assignments

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

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

| Address | Function |
|---|---|
| 61C0Exx0–61C0Exx7 | LCD control |
| 61C0Exx8 | LED enable |
| 61C0Exx9 | Not used |
| 61C0ExxA | 7-key keypad |
| 61C0ExxB (bits 0–6) | 7-LED driver |
| 61C0ExxB (bit 7) | LCD backlight on/off |
| 61C0ExxC–61C0ExxF | Not used |

## B.5 Mounting LCD/Keypad Module

### B.5.1 Installation Guidelines

When possible, following these guidelines when mounting the LCD/keypad module.

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

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

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

## B.5.2 Mounting Instructions

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

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

### B.5.2.1 Bezel-Mount Installation

This section describes and illustrates how to bezel-mount the LCD/keypad module. Follow these steps for bezel-mount installation.

1. Cut mounting holes in the mounting panel in accordance with the recommended dimensions in Figure B-7, then use the bezel faceplate to mount the LCD/keypad module onto the panel.

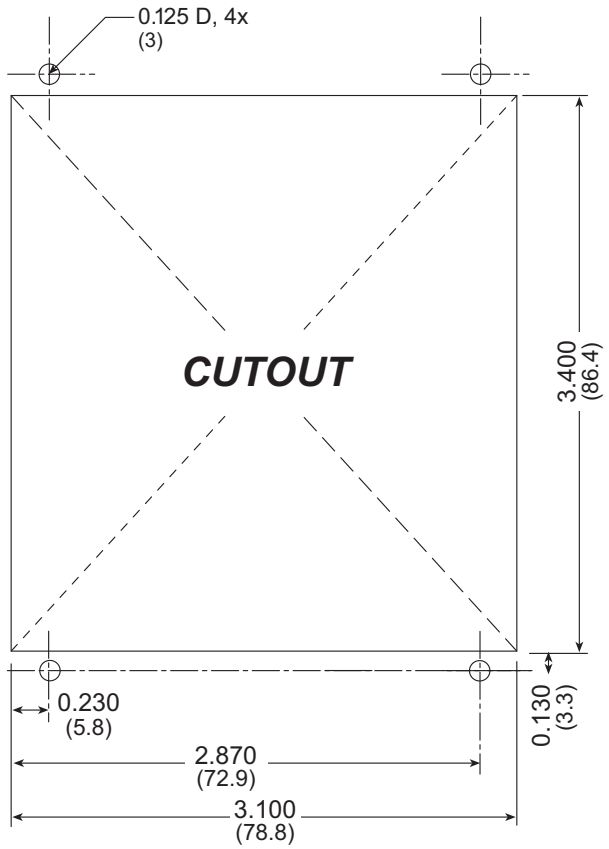



***Figure B-7. Recommended Cutout Dimensions***

2. Carefully “drop in” the LCD/keypad module with the bezel and gasket attached.

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

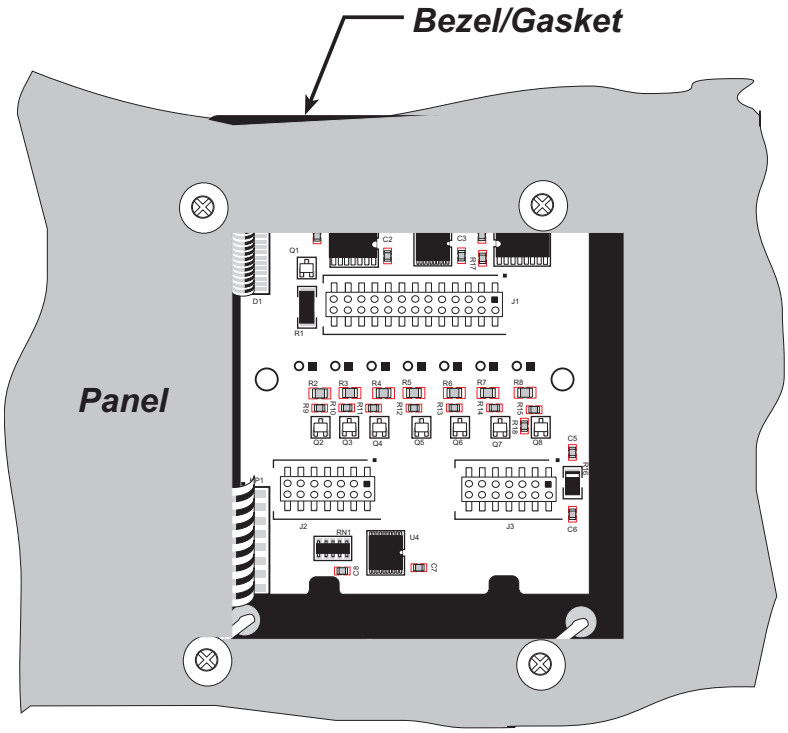



***Figure B-8. LCD/Keypad Module Mounted in Panel (rear view)***

Carefully tighten the screws until the gasket is compressed and the plastic bezel face-plate is touching the panel.

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

## B.6 Connecting LCD/Keypad Module to Smart Star Backplane

The LCD/keypad module can be located as far as 2 ft. (60 cm) away from the Smart Star backplane, and is connected via a ribbon cable as shown in Figure B-9.

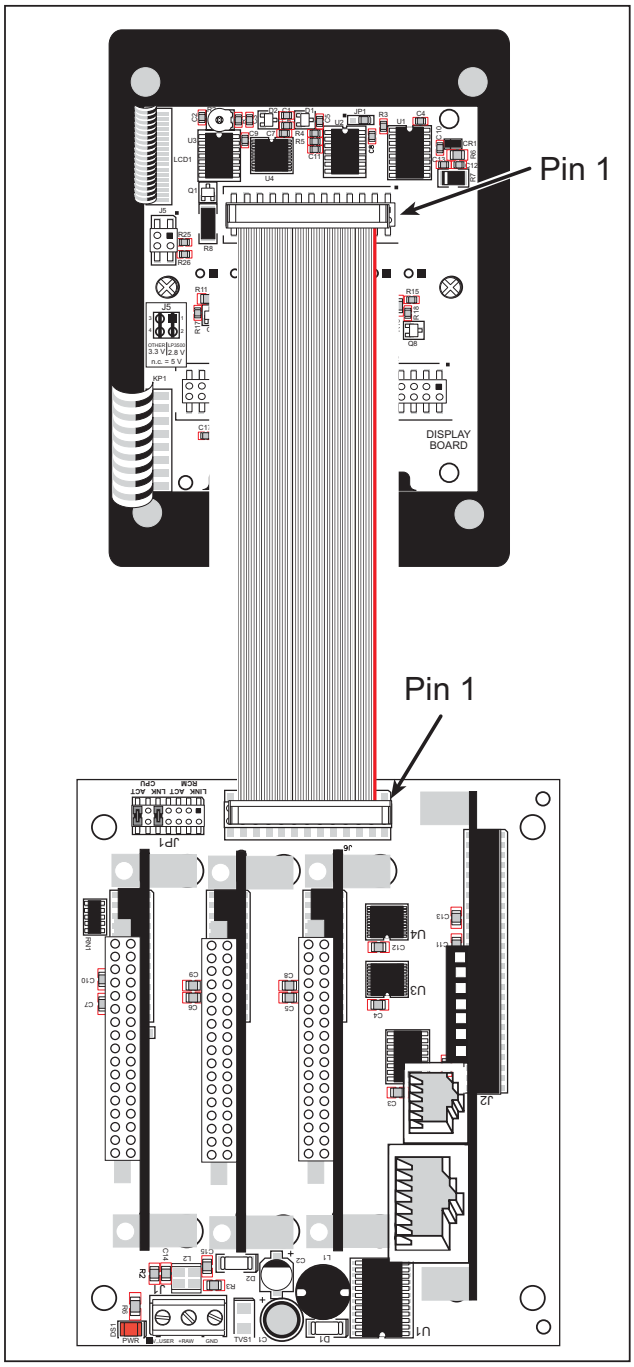



***Figure B-9. Connecting LCD/Keypad Module to Backplane***

Note the locations and connections for pin 1 on both the backplane and the LCD/keypad module.

The SR9050 backplane can also be panel-mounted behind the LCD/keypad module.

1. Prepare a cutout and install the LCD/keypad module in the cutout as explained in Section B.5.2.1.
2. Use brackets to secure the LCD/keypad module to the panel using the four 4-40 screws and washers included with the LCD/keypad module. The four screw positions are indicated with the number 1 in Figure B-10. If your panel is thick, use a 4-40 screw that is approximately 3/16" (5 mm) longer than the thickness of the panel.
3. Use a ribbon cable to connect header J6 on the backplane to header J1 on the LCD/keypad module. Note the pin 1 positions reflected by the red-colored line in the ribbon cable shown in Figure B-10.

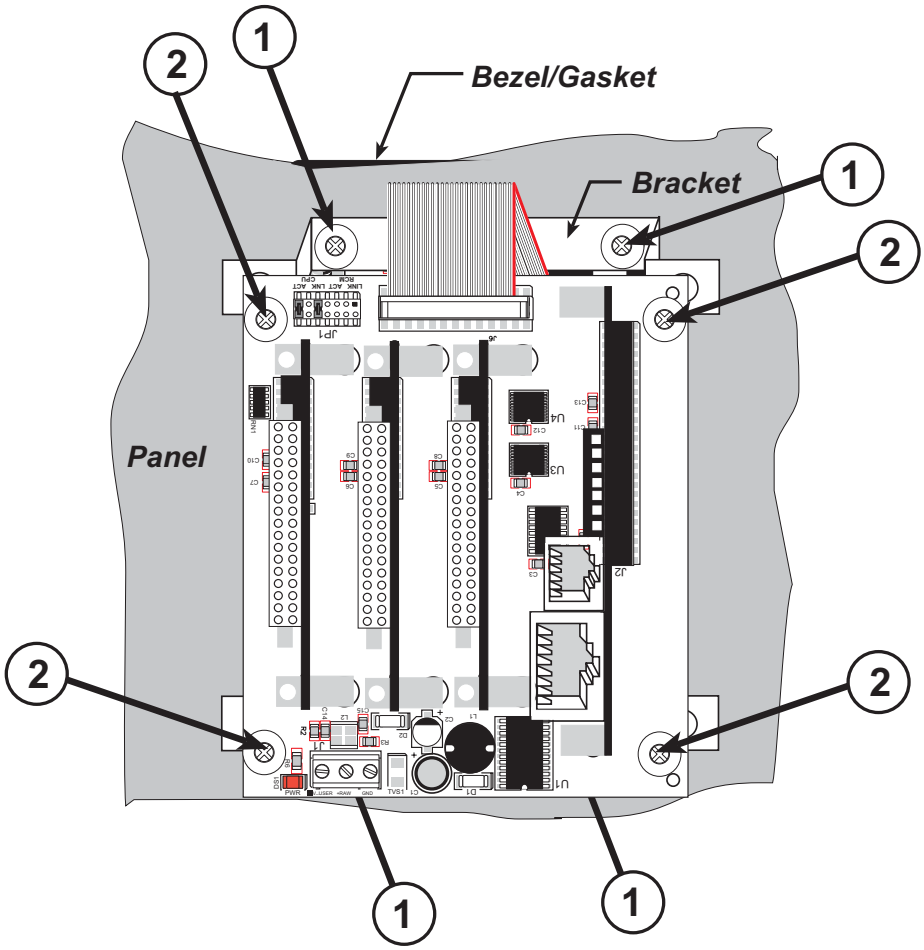



***Figure B-10. Install Smart Star Backplane Behind LCD/Keypad Module***

4. Secure the Smart Star backplane using four 4-40 × ½" or 6-32 × ½" screws at the screw positions indicated with the number 2 in Figure B-10.

Brackets and ribbon cables are sold separately. Note that only a Smart Star assembly using the SR9050 backplane can be panel-mounted.

## B.7 Sample Programs

The following sample programs are found in the **SAMPLES\LCD_Keypad\122x32_1x7** folder.

- **ALPHANUM.C**—Demonstrates how to create messages using the keypad and then displaying them on the LCD display.
- **COFTERMA.C**—Demonstrates cofunctions, the cofunction serial library, and using a serial ANSI terminal such as Hyperterminal from an available COM port connection.
- **DISPPONG.C**—Demonstrates output to LCD display.
- **DKADEMO1.C**—Demonstrates some of the LCD/keypad module font and bitmap manipulation features with horizontal and vertical scrolling, and using the **GRAPHIC.LIB** library.
- **FUN.C**—Demonstrates drawing primitive features (lines, circles, polygons) using the **GRAPHIC.LIB** library
- **KEYBASIC.C**—Demonstrates the following keypad functions in the **STDIO** display window:
  - default ASCII keypad return values.
  - custom ASCII keypad return values.
  - keypad repeat functionality.
- **KEYMENU.C**—Demonstrates how to implement a menu system using a highlight bar on a graphic LCD display. The menu options for this sample are as follows.
  1. Set Date/Time
  2. Display Date/Time
  3. Turn Backlight OFF
  4. Turn Backlight ON
  5. Toggle LEDs
  6. Increment LEDs
  7. Disable LEDs
- **LED.C**—Demonstrates how to toggle the LEDs on the LCD/keypad module.
- **SCROLLING.C**—Demonstrates scrolling features of the **GRAPHIC.LIB** library.
- **TEXT.C**—Demonstrates the text functions in the **GRAPHIC.LIB** library. Here is a list of what is demonstrated.
  1. Font initialization.
  2. Text window initialization.
  3. Text window, end-of-line wraparound, end-of-text window clipping, line feed, and carriage return.
  4. Creating 2 different TEXT windows for display.
  5. Displaying different FONT sizes.

The following sample programs, found in the **SAMPLES\LCD_Keypad\122x32_1x7\TCPIP** folder, are targeted at the Ethernet-enabled versions of the Smart Star and the BL2110. Remember to configure the IP address, netmask, and gateway as indicated in the sample programs.

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

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

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

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

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

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

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

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

## B.8 LCD/Keypad Module Function Calls

### B.8.1 LEDs

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

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

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

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

**PARAMETERS**

**`led`** is the LED to control.

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

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

0 = off
1 = on

**RETURN VALUE**

None.

**SEE ALSO**

**`brdInit`**

### B.8.2 LCD Display

The functions used to control the LCD display are contained in the Dynamic C **LIB\DISPLAYS\GRAPHIC\GRAPHIC.LIB** library. When *x* and *y* coordinates on the display screen are specified, *x* can range from 0 to 121, and *y* can range from 0 to 31. These numbers represent pixels from the top left corner of the display.

## void glInit(void);

Initializes the display devices, clears the screen.

**RETURN VALUE**

None.

**SEE ALSO**

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

## void glBackLight(int onOff);

Turns the display backlight on or off.

**PARAMETER**

**onOff** turns the backlight on or off

1—turn the backlight on
0—turn the backlight off

**RETURN VALUE**

None.

**SEE ALSO**

**glInit, glDispOnoff, glSetContrast**

## void glDispOnOff(int onOff);

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

**PARAMETER**

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

1—turn the LCD screen on
0—turn the LCD screen off

**RETURN VALUE**

None.

**SEE ALSO**

**glInit, glSetContrast, glBackLight**

## void glSetContrast(unsigned level);

Sets display contrast.

> **NOTE:** This function is not used with the LCD/keypad module since the support circuits are not available on the LCD/keypad module.

## void glFillScreen(char pattern);

Fills the LCD display screen with a pattern.

**PARAMETER**

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

**RETURN VALUE**

None.

**SEE ALSO**

**glBlock, glBlankScreen, glPlotPolygon, glPlotCircle**

## void glBlankScreen(void);

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

**RETURN VALUE**

None.

**SEE ALSO**

**glFillScreen, glBlock, glPlotPolygon, glPlotCircle**

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

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

**PARAMETERS**

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

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

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

**bmWidth** is the height of the block.

**RETURN VALUE**

None.

**SEE ALSO**

**glFillScreen, glBlankScreen, glPlotPolygon, glPlotCircle**

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

Plots the outline of a polygon in the LCD page buffer, and on the LCD if the buffer is unlocked. Any portion of the polygon that is outside the LCD display area will be clipped. If fewer than 3 vertices are specified, the function will return without doing anything.

### PARAMETERS

**n** is the number of vertices.

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

### RETURN VALUE

None.

### SEE ALSO

**glPlotPolygon, glFillPolygon, glFillVPolygon**

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

Plots the outline of a polygon in the LCD page buffer and on the LCD if the buffer is unlocked. Any portion of the polygon that is outside the LCD display area will be clipped. If fewer than 3 vertices are specified, the function will return without doing anything.

### PARAMETERS

**n** is the number of vertices.

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

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

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

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

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

### RETURN VALUE

None.

### SEE ALSO

**glPlotVPolygon, glFillPolygon, glFillVPolygon**

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

Fills a polygon in the LCD page buffer and on the LCD screen if the buffer is unlocked. Any portion of the polygon that is outside the LCD display area will be clipped. If fewer than 3 vertices are specified, the function will return without doing anything.

### PARAMETERS

**n** is the number of vertices.

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

### RETURN VALUE

None.

### SEE ALSO

**glFillPolygon, glPlotPolygon, glPlotVPolygon**

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

Fills a polygon in the LCD page buffer and on the LCD if the buffer is unlocked. Any portion of the polygon that is outside the LCD display area will be clipped. If fewer than 3 vertices are specified, the function will return without doing anything.

### PARAMETERS

**n** is the number of vertices.

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

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

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

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

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

### RETURN VALUE

None.

### SEE ALSO

**glFillVPolygon, glPlotPolygon, glPlotVPolygon**

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

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

### PARAMETERS

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

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

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

### RETURN VALUE

None.

### SEE ALSO

**glFillCircle, glPlotPolygon, glFillPolygon**

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

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

### PARAMETERS

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

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

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

### RETURN VALUE

None.

### SEE ALSO

**glPlotCircle, glPlotPolygon, glFillPolygon**

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

Initializes the font descriptor structure, where the font is stored in **xmem**.

**PARAMETERS**

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

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

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

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

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

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

**RETURN VALUE**

None.

**SEE ALSO**

**glPrinf**

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

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

**PARAMETERS**

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

**letter** is an ASCII character.

**RETURN VALUE**

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

**SEE ALSO**

**glPutFont, glPrintf**

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

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

### PARAMETERS

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

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

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

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

### RETURN VALUE

None.

### SEE ALSO

**glFontCharAddr, glPrintf**

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

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

### PARAMETERS

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

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

### RETURN VALUE

None.

### SEE ALSO

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

## int glGetPfStep(void);

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

### RETURN VALUE

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

### SEE ALSO

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

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

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

### PARAMETERS

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

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

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

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

### RETURN VALUE

None.

### SEE ALSO

**glPrintf, glPutFont, doprnt**

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

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

### PARAMETERS

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

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

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

***fmt** is a formatted string.

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

### EXAMPLE

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

### RETURN VALUE

None.

### SEE ALSO

**glXFontInit**

## void glBuffLock(void);

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

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

**RETURN VALUE**

None.

**SEE ALSO**

**glBuffUnlock, glSwap**

## void glBuffUnlock(void);

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

**RETURN VALUE**

None.

**SEE ALSO**

**glBuffLock, glSwap**

## void glSwap(void);

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

**RETURN VALUE**

None.

**SEE ALSO**

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

## void glSetBrushType(int type);

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

**PARAMETER**

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

**PIXBLACK** draws black pixels (turns pixel on).
**PIXWHITE** draws white pixels (turns pixel off).
**PIXXOR** draws old pixel XOR'ed with the new pixel.

**RETURN VALUE**

None.

**SEE ALSO**

**glGetBrushType**

## int glGetBrushType(void);

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

**RETURN VALUE**

The current brush type.

**SEE ALSO**

`glSetBrushType`

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

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

**PARAMETERS**

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

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

**RETURN VALUE**

None.

**SEE ALSO**

`glPlotline, glPlotPolygon, glPlotCircle`

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

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

**PARAMETERS**

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

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

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

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

**RETURN VALUE**

None.

**SEE ALSO**

`glPlotDot, glPlotPolygon, glPlotCircle`

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

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

**PARAMETERS**

**left** is the top left corner of bitmap, must be evenly divisible by 8, otherwise truncates.

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

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

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

**RETURN VALUE**

None.

**SEE ALSO**

**glHScroll, glRight1**

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

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

**PARAMETERS**

**left** is the top left corner of bitmap, must be evenly divisible by 8, otherwise truncates.

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

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

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

**RETURN VALUE**

None.

**SEE ALSO**

**glHScroll, glLeft1**

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

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

**PARAMETERS**

**left** is the top left corner of bitmap, must be evenly divisible by 8, otherwise truncates.

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

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

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

**RETURN VALUE**

None.

**SEE ALSO**

**glVScroll, glDown1**

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

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

### PARAMETERS

**left** is the top left corner of bitmap, must be evenly divisible by 8, otherwise truncates.

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

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

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

### RETURN VALUE

None.

### SEE ALSO

**glVScroll, glUp1**

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

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

Parameters will be verified for the following:

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

### PARAMETERS

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

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

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

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

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

### RETURN VALUE

None.

### SEE ALSO

**glVScroll**

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

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

Parameters will be verified for the following:

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

### PARAMETERS

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

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

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

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

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

### RETURN VALUE

None.

### SEE ALSO

**glHScroll**

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

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

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

### PARAMETERS

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

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

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

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

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

### RETURN VALUE

None.

### SEE ALSO

**glXPutFastmap, glPrintf**

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

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

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

**PARAMETERS**

**left** is the top left corner of the bitmap, must be evenly divisible by 8, otherwise truncates.

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

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

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

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

**RETURN VALUE**

None.

**SEE ALSO**

**glXPutBitmap, glPrintf**

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

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

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

**PARAMETERS**

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

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

**x** is the *x* coordinate of the top left corner of the text window frame.

**y** is the *y* coordinate of the top left corner of the text window frame.

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

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

**RETURN VALUE**

0—window frame was successfully created.
-1—*x* coordinate + width has exceeded the display boundary.
-2—*y* coordinate + height has exceeded the display boundary.

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

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

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

**PARAMETERS**

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

**`col`** is a character column location.

**`row`** is a character row location.

**RETURN VALUE**

None.

**SEE ALSO**

**`TextPutChar, TextPrintf, TextWindowFrame`**

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

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

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

**PARAMETERS**

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

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

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

**RETURN VALUE**

Lower word = Cursor Row location
Upper word = Cursor Column location

**SEE ALSO**

**`TextGotoXY, TextPrintf, TextWindowFrame, TextCursorLocation`**

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

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

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

### PARAMETERS

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

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

### RETURN VALUE

None.

### SEE ALSO

**TextGotoXY, TextPrintf, TextWindowFrame, TextCursorLocation**

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

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

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

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

### PARAMETERS

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

***fmt** is a formatted string.

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

### EXAMPLE

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

### RETURN VALUE

None.

### SEE ALSO

**TextGotoXY, TextPutChar, TextWindowFrame, TextCursorLocation**

### B.8.3 Keypad

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

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

Initializes keypad process

**RETURN VALUE**

None.

**SEE ALSO**

**brdInit**

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

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

**PARAMETERS**

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

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

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

#### User Keypad Interface

**cPress** is a key press code

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

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

**cRelease** is a key release code.

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

**cCntHold** is a hold tick, which is approximately one debounce period or 5 µs.

How long to hold before repeating.
0 = No Repeat.

**cSpdLo** is a low-speed repeat tick, which is approximately one debounce period or 5 µs.

How many times to repeat.
0 = None.

**cCntLo** is a low-speed hold tick, which is approximately one debounce period or 5 µs.

How long to hold before going to high-speed repeat.
0 = Slow Only.

**cSpdHi** is a high-speed repeat tick, which is approximately one debounce period or 5 µs.

How many times to repeat after low-speed repeat.
0 = None.

**RETURN VALUE**

None.

**SEE ALSO**

**keyProcess, keyGet, keypadDef**

## void keyProcess(void);

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

**NOTE:** This function is also able to process an 8 × 8 matrix keypad.

**RETURN VALUE**

None

**SEE ALSO**

**keyConfig, keyGet, keypadDef**

## char keyGet(void);

Get next keypress

**RETURN VALUE**

The next keypress, or 0 if none

**SEE ALSO**

**keyConfig, keyProcess, keypadDef**

## int keyUnget(char cKey);

Pushes the value of **cKey** to the top of the input queue, which is 16 bytes deep.

**PARAMETER**

**cKey**

**RETURN VALUE**

None.

**SEE ALSO**

**keyGet**

## void keypadDef();

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

Keypad physical mapping 1 × 7

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

where

'D' represents Down Scroll
'U' represents Up Scroll
'R' represents Right Scroll
'L' represents Left Scroll
'–' represents Page Down
'+' represents Page Up
'E' represents the ENTER key

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

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

Characters are returned upon keypress with no repeat.

### RETURN VALUE

None.

### SEE ALSO

`keyConfig, keyGet, keyProcess`

## void keyScan(char *pcKeys);

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

### PARAMETER

`*pcKeys` is a pointer to the address of the value read.

### RETURN VALUE

None.

### SEE ALSO

`keyConfig, keyGet, keypadDef, keyProcess`

## B.9 Font and Bitmap Converter

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

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

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

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

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

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

# APPENDIX C. POWER MANAGEMENT

Appendix C provides information on the current requirements of the Smart Star I/O cards, the use and installation of a backup battery, and some background on power management.

## C.1 Current Requirements

Remember to take the current draw of the various I/O cards into consideration when selecting the power supply for your Smart Star control system.

Table C-1 lists the typical current consumption for the CPU Card and the I/O cards.

***Table C-1. Current Consumption of I/O Cards Attached to Smart Star Backplane***

| I/O Cards | Current Consumption | |
|---|---|---|
| | +5 V Supply | +V_USER Supply |
| Digital I/O (SR9200 series) | 65 mA | up to 200 mA/output* |
| A/D Converter (SR9300 series)<br>D/A Converter (SR9400 series) | 40 mA | 35 mA |
| Relay (SR9500 series) | 10 mA | 75 mA |
| CPU Card | 190 mA | — |

* Maximum current 2.0 A per I/O card, 7.0 A for Smart Star system

## C.2 Batteries and External Battery Connections

An onboard 265 mA·h lithium coin cell on the CPU Card provides power to the real-time clock and SRAM when external power is removed from the Smart Star control system. This allows the CPU Card to continue to keep track of time and preserves the SRAM memory contents while the power is off.

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

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

The drain on the battery is typically less than 4 µA when external power ***is*** applied. The battery can last for

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

Since the shelf life of the battery is 10 years, the battery can last for most of its shelf life when external power is applied most of the time.

### C.2.1 Replacing the Backup Battery

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

> **NOTE:** The SRAM contents and the real-time clock settings will be lost if the battery is replaced with no power applied to the Smart Star. There is a provision for an external battery if you need to save the SRAM contents and the real-time clock settings since the CPU Card needs to be removed from the backplane in order to change the onboard battery.

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

### C.2.2 Battery-Backup Circuit

Figure C-1 shows the battery-backup circuit.

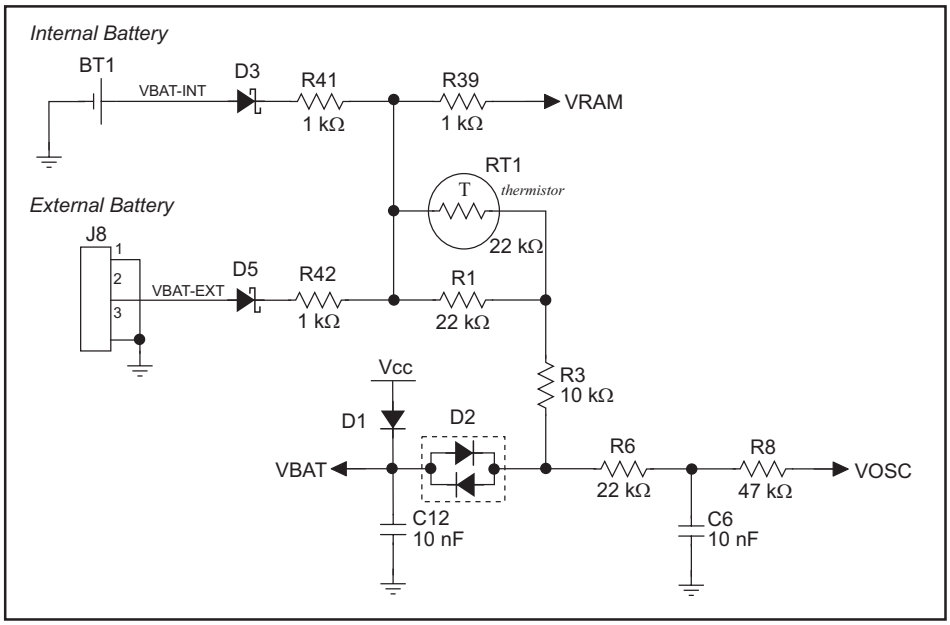



***Figure C-1. Smart Star CPU Card Backup Battery Circuit***

The battery-backup circuit serves three purposes:

- It reduces the battery voltage to the SRAM and to the real-time clock, thereby limiting the current consumed by the real-time clock and lengthening the battery life.
- It ensures that current can flow only ***out*** of the battery to prevent charging the battery.
- A voltage, VOSC, is supplied to U14, which keeps the 32.768 kHz oscillator working when the voltage begins to drop.

VRAM and Vcc are nearly equal (<100 mV, typically 10 mV) when power is supplied to the CPU Card.

### C.2.3 Power to VRAM Switch

The VRAM switch, shown in Figure C-2, allows the battery backup to provide power when the external power goes off. The switch provides an isolation between +5 V and the battery when +5 V goes low. This prevents the +5 V line from draining the battery.

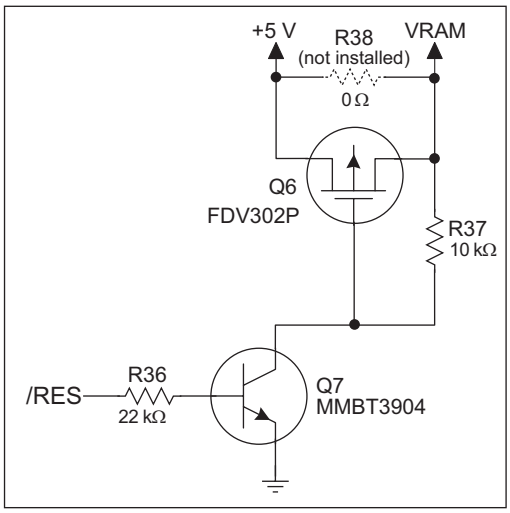



***Figure C-2. VRAM Switch***

Transistor Q6 is needed to provide a very small voltage drop between +5 V and VRAM (<100 mV, typically 10 mV) so that the processor lines powered by +5 V will not have a significantly different voltage than VRAM.

When the CPU Card is ***not*** resetting (pin 2 on U4 is high), the /RES line will be high. This turns on Q6, causing its collector to go low. This turns on Q7, allowing VRAM to nearly equal +5 V.

When the CPU Card ***is*** resetting, the /RES line will go low. This turns off Q6 and Q7, providing an isolation between +5 V and VRAM.

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

### C.2.4 Reset Generator

The CPU Card uses a reset generator, U4, to reset the Rabbit 2000 microprocessor when the voltage drops below the voltage necessary for reliable operation. The reset occurs between 4.50 V and 4.75 V, typically 4.63 V. The reset can be initiated either externally or by a watchdog timeout (**WDTOUT**) on the Rabbit 2000 microprocessor.

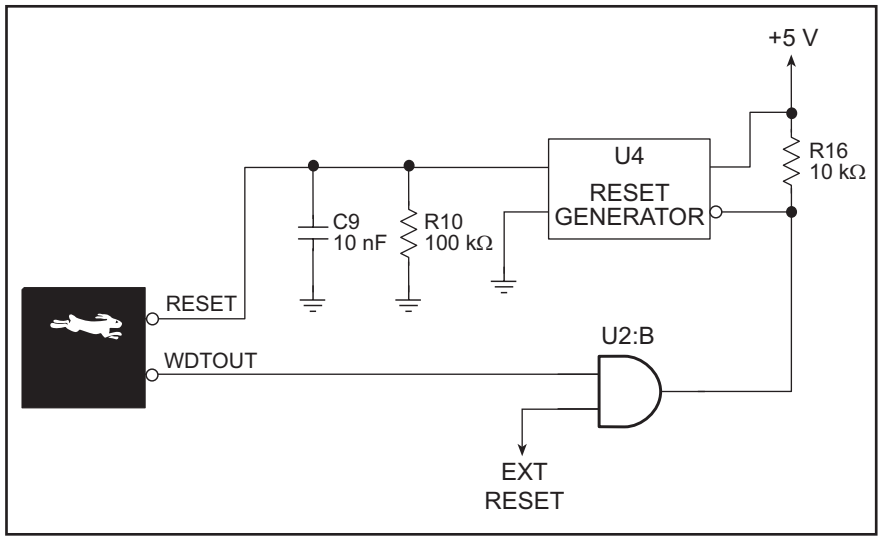



***Figure C-3. Reset Generator***

**NOTE:** The Dynamic C function **`chkWDTO`** is not able to detect whether a watchdog timeout has occurred on the SR9100 series of CPU cards. The GCSR status bits are read and stored by the BIOS, and the reset status bit would normally change once a reset has occurred. However, since **WDTOUT** is tied to the reset generator, a watchdog timeout forces a hardware reset, followed by the BIOS reading and storing the status bits corresponding to power-up or reset.

### C.2.5 External Battery

A connection for an external backup battery is provided at header J8, shown in Figure C-4. The header is wired to provide reverse polarity protection.

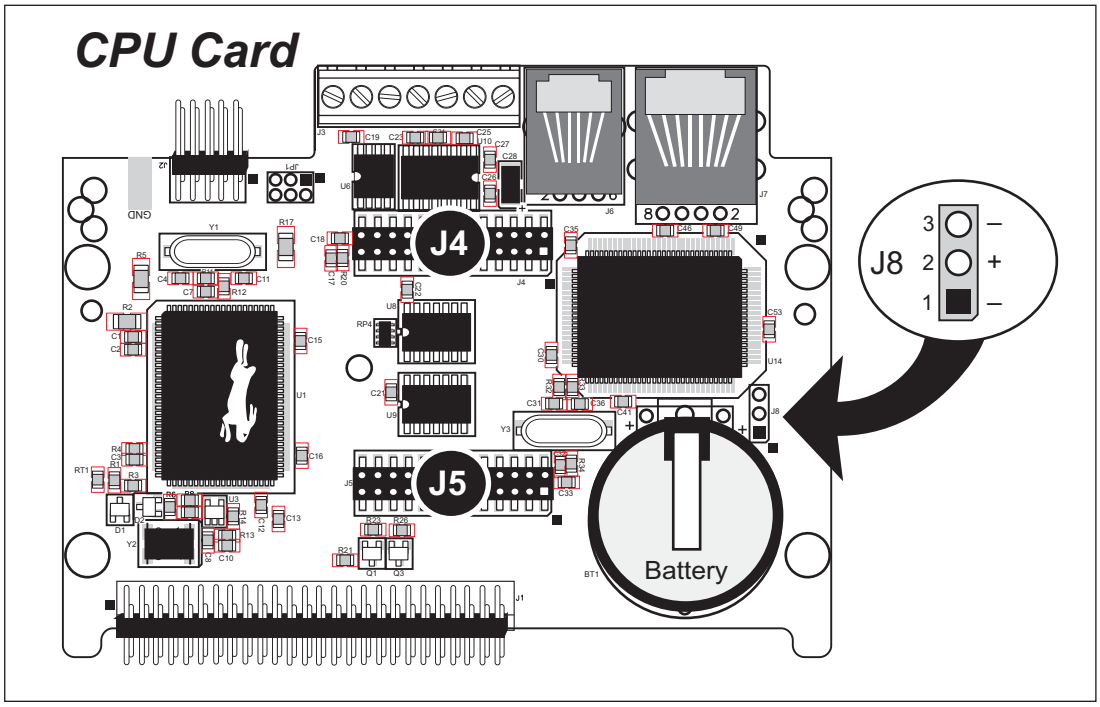



***Figure C-4. External Backup Battery Connection***

The external battery connection is useful if the SRAM and real-time clock data need to be preserved while the backup battery is being changed. This way power can continue to be applied to the CPU Card from the backplane (if the external backup battery is being replaced) or from the external battery (if the onboard backup battery needs to be changed since this requires removing the CPU Card from the backplane in order to access the onboard backup battery).

## C.3 Chip Select Circuit

Figure C-5 shows a schematic of the chip select circuit for the RAM.

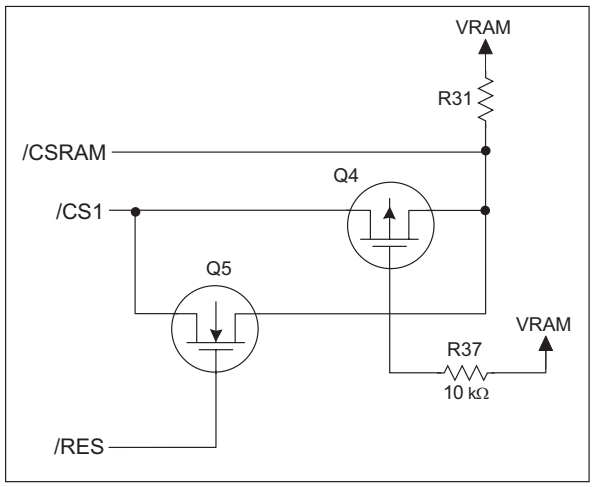



***Figure C-5. Chip Select Circuit***

The current drain on the battery in a battery-backed circuit must be kept to a minimum. When the CPU Card 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 +5 V 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). Q4 and Q5 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 (R31). 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 Q4 and Q5 are of opposite polarity so that a rail-to-rail voltage can be passed. When the /CS1 voltage is low, Q5 will conduct. When the /CS1 voltage is high, Q4 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 15 ns.

The signal that turns the transistors on is a high on the processor's reset line, /RES. When the CPU Card is not in reset, the reset line will be high, turning on n-channel Q5. When a reset occurs, the /RES line will go low.

# APPENDIX D.
# SMART STAR SLOT ADDRESS LAYOUT

Appendix D provides information about the register addresses for the various I/O card slots on the backplane. The information in this appendix will be of interest to more advanced users.

The slots on the Smart Star backplane are accessed as external registers via the Rabbit 2000's assembly **`IOE`** prefix or via standard Rabbit BIOS functions. More convenient functions specific to the Smart Star control system have been written to provide more flexibility; for example, there is now a provision for the automatic update of shadow registers for each slot and for each register.

The Smart Star design routes four address bits to each slot, providing 16 register addresses for each slot. These bits are passed through as bits 0–3 of the register address. The slot number itself is assigned to bits 6–8 of the address. In addition, the backplane design requires that bits 13 and 14 be high and that bit 9 be low. The simplest way to enforce this is to use a base address of 0x6000. Table D-1 provides the address layout for accessing the Smart Star slots, where S*n* is the binary representation of the slot number (0–6), R*n* is the binary representation of the register numbers (0–15), and *X* means the value does not matter.

***Table D-1. Smart Star External Register Address Bitmap***

| A15 | A14 | A13 | A12 | A11 | A10 | A9 | A8 | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | *X* | *X* | 0 | S2 | S1 | S0 | *X* | *X* | R3 | R2 | R1 | R0 |

This bit mapping of the external register address provides the register addresses for each slot as listed in Table D-2.

***Table D-2. Slot External Register Addresses***

| Slot Number | Address Range |
|---|---|
| 0 | 0x6000–0x600F |
| 1 | 0x6040–0x604F |
| 2 | 0x6080–0x608F |
| 3 | 0x60C0–0x60CF |
| 4 | 0x6100–0x610F |
| 5 | 0x6140–0x614F |
| 6 | 0x6180–0x618F |

## D.1 Digital I/O Card Channel Layout

The Digital I/O Card layout is complicated by the standard Rabbit method of minimizing chip layout while adding channel arrangement flexibility. In particular, the nibble-wise layout of digital input channels requires fewer chips if fewer channels are desired. This is a common feature on Rabbit products and should not surprise most users. The digital output channel layout is straightforward.

It is also possible to access the digital I/O channels in banks of eight channels. This method is significantly faster than reading eight channels one at a time, and so was included in the function call.

***Table D-3. Digital I/O Card Bank/Channel Mapping***

| Local Board Address | Input Bank | Output Bank | Input Channels | Output Channels |
|---|---|---|---|---|
| 0x00 | 0 | | 0–3/8–11 | |
| 0x01 | 2 | | 4–7/12–15 | |
| 0x02 | | 1 | | 0–7 |
| 0x03 | | 2 | | 8–15 |

## D.2 A/D Converter Card Channel Layout

The A/D Converter Card contains a single 11-input 12-bit A/D converter, TLC2543. The method of interfacing to this chip is a combination of single-bit writes via board registers and synchronous clocked serial access via the CPU Card's Serial Port B, which is extended across all eight slots. In addition, a serial EEPROM is installed on the A/D Converter Card to store the calibration constants.

***Table D-4. A/D Converter Card Control Registers***

| Address | Data Bits | Value | Description |
|---|---|---|---|
| 0x0 | Write D7–D0 | D7–D4 selects input channel,<br>D3–D0 selects conversion channel | Load A/D converter with data byte |
| 0x0 | Read D1 | 0 | A/D converter end of conversion signal |
| | | 1 | A/D converter busy |
| 0x1 | Write D0 | 0 | Enable A/D conversion |
| | | 1 | Disable A/D conversion |
| 0x2 | Write D0 | 0 | EEPROM clock line low |
| | | 1 | EEPROM clock line high |
| 0x3 | Write D0 | 0 | EEPROM data line low |
| | | 1 | EEPROM data line high |
| 0x0 | Read D2 | 0 | EEPROM acknowledge signal |
| | | 1 | EEPROM busy |

## D.3 D/A Converter Card Channel Layout

The D/A Converter Card contains four two-channel 12-bit D/A converters, TLV5618, to produce 8 analog output channels. Each channel is accessed by the slot, channel and device addressing scheme. The D/A Converter Card also has an EEPROM to store calibration constants.

***Table D-5. D/A Converter Card Control Registers***

| Address | Data Bits | Value | Description |
|---|---|---|---|
| 0x0 | D0 | 0 | D/A converter clock line low |
| | | 1 | D/A converter clock line high |
| | D1 | X | D/A converter data input line |
| | D2 | 0 | D/A converter chip select channels 0 and 1 |
| | D3 | 0 | D/A converter chip select channels 2 and 3 |
| | D4 | 0 | D/A converter chip select channels 4 and 5 |
| | D5 | 0 | D/A converter chip select channels 6 and 7 |
| | D6 | 0 | EEPROM clock line low |
| | | 1 | EEPROM clock line high |
| | D7 | X | EEPROM data line |

External reads and writes (/IORD and /IOWR) control the data direction.

## D.4 Relay Card Channel Layout

The Relay Card layout is complemented by the standard Rabbit method of minimizing chip layout while adding channel arrangement flexibility. In particular, the nibble-wise layout of the relay channels requires fewer chips if fewer channels are desired. This is a common feature on Rabbit products and should not surprise most users. The relay channel layout is straightforward.

***Table D-6. Relay Card Channel Mapping***

| Local Board Address | SR9500 Relay Channels | SR9510 Relay Channels |
|---|---|---|
| 0x00 | REL0 | REL0 |
| 0x01 | REL1 | REL1 |
| 0x02 | REL2 | REL2 |
| 0x03 | REL3 | REL3 |
| 0x04 | REL4 | REL4 |
| 0x05 | REL5 | REL5 |
| 0x06 | — | REL6 |
| 0x07 | — | REL7 |

# INDEX


## A

A/D Converter Card
- function calls ..... 92
  - anaIn ..... 94
  - anaInCalib ..... 93
  - anaInEERd ..... 92
  - anaInEEWr ..... 93
  - anaInmAmps ..... 95
  - anaInVolts ..... 94
  - anaLoadCalib ..... 92
  - anaSaveCalib ..... 92
- models ..... 87
- sample programs ..... 91

analog input conditioning circuit ..... 88

## B

backplane
- dimensions ..... 60

battery
- replacing the backup battery ..... 167

battery backup circuit ..... 167
battery connections ..... 166
battery life ..... 166

## C

CE compliance ..... 13, 14
- backplanes and cards not CE-complaint ..... 14
- CE-compliant backplanes and cards ..... 13
- design guidelines ..... 15
- LCD/keypad module ..... 13

chip select circuit ..... 171
clock doubler ..... 41
conformal coating ..... 64
connections
- Ethernet cable ..... 51
- power supply ..... 19
- programming cable ..... 20

CPU Card
- attaching to backplane ..... 18
- dimensions ..... 62

## D

D/A Converter Card
- analog outputs
  - enabling ..... 105
- circuit ..... 102
- function calls ..... 107
  - anaLoadCalib ..... 109
  - anaOut ..... 110
  - anaOutCalib ..... 108
  - anaOutDisable ..... 105, 107
  - anaOutEERd ..... 107
  - anaOutEEWr ..... 112
  - anaOutEnable ..... 105, 107
  - anaOutmAmps ..... 111
  - anaOutVolts ..... 111
  - anaSaveCalib ..... 109
- models ..... 101
- sample programs ..... 106

digital I/O
- SMODE0 ..... 35
- SMODE1 ..... 35

Digital I/O Card ..... 71
- banks
  - Bank 2 configurations ... 75
  - locations ..... 74
- digital outputs
  - digBankOut ..... 81
  - digOut ..... 81
  - function calls ..... 81
- function calls ..... 80
  - digBankIn ..... 80
  - digIn ..... 80
- locations of I/O banks ..... 74
- sample programs ..... 79

digital inputs ..... 76
- pulldown configuration ..... 76
- pullup configuration ..... 76
- pullup/pulldown jumper settings ..... 76

digital outputs
- connecting a load ..... 78
- sinking or sourcing jumper settings ..... 78

dimensions
- A/D Converter Card ..... 96
- backplane ..... 60
- CPU Card ..... 62
- D/A Converter Card ..... 113
- Digital I/O Card ..... 82
- field wiring terminals ..... 129
- LCD/keypad module ..... 131
- LCD/keypad template ..... 134
- Relay Cards ..... 123

Dynamic C ..... 12
- add-on modules ..... 45
- COM port ..... 23
- libraries . 47, 79, 91, 106, 121
- running sample programs ..... 91, 106, 121
- standard features ..... 44
  - debugging ..... 44
- starting ..... 23
- telephone-based technical support ..... 12, 45
- upgrades and patches ..... 45

## E

EMI
- spectrum spreader feature . 41

Ethernet cables ..... 51
Ethernet connections ..... 51
- 10Base-T Ethernet card .... 51
- Ethernet hub ..... 51
- steps ..... 51

Ethernet port
- handling EMI and noise .... 36
- pinout ..... 36

exclusion zone ..... 68

## F


features .....9
A/D Converter Card .....87
D/A Converter Card .....101
Digital I/O Card .....71
Relay Cards .....117
field wiring terminals ....11, 128
guide to FWT selection .....11, 128
A/D Converter Card .....89
D/A Converter Card ....104
Digital I/O Card .....73
Relay Cards .....119
installation .....128
positioning on I/O card ....128
flash memory
lifetime write cycles .....43
font and bitmap converter ...163
FWT. See field wiring terminals


## H


headers
JP1 .....34


## I


I/O address assignments
LCD/keypad module .....135
I/O cards
attaching to backplane .....25
installation
CPU Card .....18
field wiring terminals .....128
I/O cards .....25
IP addresses
how to set .....53
how to set PC IP address ...54


## J


jumper configurations .....65
JP1 (RS-485 bias and termination resistors) .....34, 65
JP2 (flash memory bank select) .....38
JP5 (flash memory bank select) .....65
jumper locations .....64
jumper settings
digital inputs
pullup/pulldown .....76
digital outputs
sinking or sourcing .....78


## K


keypad template .....134
removing and inserting label .....134


## L


LCD/keypad module .....11
contrast adjustment .....133
dimensions .....131
header pinout .....135
I/O address assignments ..135
keypad
function calls
keyConfig .....160
keyGet .....161
keyInit .....160
keypadDef .....162
keyProcess .....161
keyScan .....162
keyUnget .....161
keypad template .....134
LCD display
function calls
glBackLight .....144
glBlankScreen .....145
glBlock .....145
glBuffLock .....152
glBuffUnlock .....152
glDispOnOff .....144
glDown1 .....155
glFillCircle .....148
glFillPolygon .....147
glFillScreen .....145
glFillVPolygon .....147
glFontCharAddr .....149
glGetBrushType .....153
glGetPfStep .....150
glHScroll .....155
glInit .....144
glLeft1 .....154
glPlotCircle .....148
glPlotDot .....153
glPlotLine .....153
glPlotPolygon .....146
glPlotVPolygon .....146
glPrintf .....151
glPutChar .....151
glPutFont .....150
glRight1 .....154
glSetBrushType .....152
glSetContrast .....145
glSetPfStep .....150
glSwap .....152
glUp1 .....154
glVScroll .....156
glXFontInit .....149, 163
glXPutBitmap ..156, 163
glXPutFastmap .....157
TextCursorLocation .158
TextGotoXY .....158
TextPrintf .....159
TextPutChar .....159
TextWindowFrame ..157
LEDs
function calls .....143
ledOut .....143
mounting instructions .....136
removing and inserting keypad label .....134
sample programs .....141
versions .....131


## M


memory .....38
flash EPROM configuration for different sizes .....38
SRAM configuration for different sizes .....38
models
A/D Converter Card .....87
D/A Converter Card .....101
Digital I/O Card .....71
Relay Cards .....117
mounting instructions
LCD/keypad module .....136


## O


options
LCD/keypad module .....11


## P


pinout
A/D Converter Card user interface .....88
backplane SLOT 0–SLOT 6 .....31
CPU Card (serial communication) .....32
D/A Converter Card user interface .....103
Digital I/O Card .....72
digital inputs .....75
digital outputs .....77
user interface .....72
Ethernet port .....36

pinout (continued)
FWT
A/D Converter Card ..... 89
D/A Converter Card .... 104
Digital I/O Card ..... 73
Relay Cards ..... 119
LCD/keypad module ..... 135
Relay Cards ..... 118
power distribution
A/D Converter Card ..... 90
backplane ..... 29
CPU Card ..... 29
D/A Converter Card ..... 105
Digital I/O Card ..... 77
Relay Cards ..... 120
Smart Star system ..... 30
power management ..... 165
power supplies
backup-battery circuit ..... 167
battery backup ..... 166
battery backup circuit ..... 167
battery life ..... 166
chip select circuit ..... 171
VRAM switch ..... 168
power supply ..... 12
programming
flash vs. RAM ..... 43
programming cable ..... 12, 20
programming port ..... 35
programming cable ... 12, 20, 37
PROG connector ..... 20, 37
switching between Program Mode and Run Mode .... 37
programming port ..... 35

## R

Relay Cards
function calls ..... 122
relayOut ..... 122
sample programs ..... 121
relay circuit configurations . 118
diodes ..... 118
snubbers ..... 118
reset ..... 21
reset generator ..... 169
RS-232 ..... 32
RS-485 network ..... 33
termination and bias resistors ..... 34

## S

sample programs ..... 46
A/D Converter Card ..... 91
SSTARAD1.C ..... 91
SSTARAD2.C ..... 91
SSTARAD3.C ..... 91
D/A Converter Card ..... 106
ANAVOUT.C ..... 106
SSDAC1.C ..... 106
SSDAC2.C ..... 106
SSDAC3.C ..... 106
SSDAC4.C ..... 106
Digital I/O Card ..... 79
SSTARIO.C ..... 79
how to set IP address ..... 53
LCD/keypad module ..... 141
ALPHANUN.C ..... 141
COFTERMA.C ..... 141
DISPPONG.C ..... 141
DKADEMO1.C ..... 141
FUN.C ..... 141
KEYBASIC.C ..... 134, 141
KEYMENU.C ..... 141
LED.C ..... 141
SCROLLING.C ..... 141
TEXT.C ..... 141
LCD/keypad module (with TCP/IP)
MBOXDEMO.C ... 56, 142
TCP_RESPOND.C 56, 142
TCPSEND.C ..... 56, 142
PONG.C ..... 24
Relay Cards ..... 121
SSTARRLY.C ..... 121
serial communication
MASTER.C ..... 46
SLAVE.C ..... 46
SSTAR232.C ..... 46
SSTAR5W.C ..... 46
TCP/IP ..... 53
PINGME.C ..... 55
SMTP.C ..... 55
SSI.C ..... 55
SSI2.C ..... 55
serial communication
function calls ..... 49
serDRS485Rx ..... 50
serDRS485Tx ..... 49
serMode ..... 49
programming port ..... 35
RS-232 description ..... 32
RS-485 description ..... 33
RS-485 network ..... 33
RS-485 termination and bias resistors ..... 34
slot address layout ..... 173
A/D Converter Card ..... 176
D/A Converter Card ..... 177
Digital I/O Card ..... 175
Relay Cards ..... 178
Smart Star bus reset
function calls ..... 48
Smart Star initialization
function calls ..... 48
SMRTSTAR.LIB
function calls
brdInit ..... 48
brdResetBus ..... 48
software .... 47, 79, 91, 106, 121
libraries . 47, 79, 91, 106, 121
PACKET.LIB ..... 49
RS232.LIB ..... 49
SMRTSTAR.LIB ..... 47, 79, 91, 106, 121
specifications ..... 59
A/D Converter Card
dimensions ..... 96
electrical ..... 97
temperature ..... 97
backplane
dimensions ..... 60
electrical ..... 61
temperature ..... 61
CPU Card
dimensions ..... 62
electrical ..... 63
mechanical ..... 63
temperature ..... 63
D/A Converter Card
dimensions ..... 113
electrical ..... 114
temperature ..... 114
Digital I/O Card
dimensions ..... 82
electrical ..... 83
temperature ..... 83
exclusion zone ..... 68
field wiring terminals
dimensions ..... 129
LCD/keypad module
dimensions ..... 131
electrical ..... 132
header footprint ..... 132
mechanical ..... 132
relative pin 1 locations 132
temperature ..... 132

specifications (continued)
Relay Cards
dimensions .....123
electrical .....124
temperature .....124
spectrum spreader .....41
subsystems .....32

## T

TCP/IP connections
additional resources .....57
Tool Kit .....12
DC power supply .....12
Dynamic C software .....12
field wiring terminal .....12
programming cable .....12
software .....12
User's Manual .....12

## W

watchdog timeout
function calls
chkWDTO .....169

# SCHEMATICS

## 090-0129 CPU Card (SR9150) Schematic

www.rabbit.com/documentation/schemat/090-0129.pdf

## 090-0143 Backplane (SR9010) Schematic

www.rabbit.com/documentation/schemat/090-0143.pdf

## 090-0130 Backplane (SR9050) Schematic

www.rabbit.com/documentation/schemat/090-0130.pdf

## 090-0101 Digital I/O Card–Sinking (SR9200) Schematic

www.rabbit.com/documentation/schemat/090-0101.pdf

## 090-0118 Digital I/O Card–Sourcing (SR92x5) Schematic

www.rabbit.com/documentation/schemat/090-0118.pdf

## 090-0086 A/D Converter Card (SR9300) Schematic

www.rabbit.com/documentation/schemat/090-086.pdf

## 090-0121 D/A Converter Card (SR9400) Schematic

www.rabbit.com/documentation/schemat/090-0121.pdf

## 090-0098 6-Relay Card (SR9500) Schematic

www.rabbit.com/documentation/schemat/090-0098.pdf

## 090-0108 8-Relay Card (SR9510) Schematic

www.rabbit.com/documentation/schemat/090-0108.pdf

## 090-0102 FWT18 Schematic

www.rabbit.com/documentation/schemat/090-0102.pdf

## 090-0106 FWT18R Schematic

www.rabbit.com/documentation/schemat/090-0106.pdf

## 090-0103 FWT27 Schematic

www.rabbit.com/documentation/schemat/090-0103.pdf

## 090-0125 LCD/Keypad Module Schematic

www.rabbit.com/documentation/schemat/090-0125.pdf

## 090-0128 Programming Cable Schematic

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

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

Мы молодая и активно развивающаяся компания в области поставок электронных компонентов. Мы поставляем электронные компоненты отечественного и импортного производства напрямую от производителей и с крупнейших складов мира.

Благодаря сотрудничеству с мировыми поставщиками мы осуществляем комплексные и плановые поставки широчайшего спектра электронных компонентов.

Собственная эффективная логистика и склад в обеспечивает надежную поставку продукции в точно указанные сроки по всей России.

Мы осуществляем техническую поддержку нашим клиентам и предпродажную проверку качества продукции. На все поставляемые продукты мы предоставляем гарантию .

Осуществляем поставки продукции под контролем ВП МО РФ на предприятия военно-промышленного комплекса России , а также работаем в рамках 275 ФЗ с открытием отдельных счетов в уполномоченном банке. Система менеджмента качества компании соответствует требованиям ГОСТ ISO 9001.

Минимальные сроки поставки, гибкие цены, неограниченный ассортимент и индивидуальный подход к клиентам являются основой для выстраивания долгосрочного и эффективного сотрудничества с предприятиями радиоэлектронной промышленности, предприятиями ВПК и научно-исследовательскими институтами России.

С нами вы становитесь еще успешнее!

**Наши контакты:**

**Телефон:** +7 812 627 14 35

**Электронная почта:** sales@st-electron.ru

**Адрес:** 198099, Санкт-Петербург, Промышленная ул, дом № 19, литера Н, помещение 100-Н Офис 331