



# ***PK2400***

**C-Programmable Controller**

**User's Manual**

Revision F



---

# PK2400 User's Manual

Part Number 019-0052 • Revision F

Last revised on June 24, 2000 • Printed in U.S.A.

---

## Copyright

© 1998 Z-World, Inc. • All rights reserved.

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

---

## Trademarks

- Dynamic C<sup>®</sup> is a trademark of Z-World, Inc.
  - Windows<sup>®</sup> is a trademark of Microsoft Corporation
  - PLCBus<sup>™</sup> is a trademark of Z-World, Inc.
  - Hayes Smart Modem is a registered trademark of Hayes Microcomputer Products, Inc.
- 

## Notice to Users

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

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

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

---

## Company Address

### Z-World, Inc.

2900 Spafford Street  
Davis, California 95616-6800  
USA



Telephone: (530) 757-3737  
Facsimile: (530) 753-5141  
Web Site: <http://www.zworld.com>  
E-Mail: [zworld@zworld.com](mailto:zworld@zworld.com)

---

# TABLE OF CONTENTS

---

<b>About This Manual</b>	<b>vii</b>
<b>Chapter 1: Overview</b>	<b>11</b>
Introduction .....	12
Processor Module .....	12
PK2400 Features .....	13
PK2410 Features .....	13
Options and Upgrades .....	13
Development Kit .....	13
System Development Programming Tools .....	14
Factory Upgrades .....	14
CE Compliance .....	15
<b>Chapter 2: Getting Started</b>	<b>17</b>
Development Kit Packing List .....	18
Connecting the PK2400 to a Host PC .....	18
Establishing Communication .....	20
Steps to Run a Sample Program .....	20
<b>Chapter 3: Subsystems</b>	<b>21</b>
Processor Module .....	23
Interface Overview .....	23
Digital Input/Output .....	23
Digital Inputs .....	23
Digital Outputs .....	25
Sourcing Driver .....	26
Sinking Drivers (Optional) .....	26
Analog-to-Digital Converter .....	27
Extra Conversion .....	28
Voltage Reference .....	28
Data Conversion .....	29
Limitations on Output Range .....	29
Low-Pass Filter .....	29
Internal Test Voltages .....	29
Drift .....	30

Absolute Mode .....	30
Bipolar or Unipolar Conditioned Inputs .....	31
Factory Gain and Bias Resistors .....	31
Initial Setup .....	31
Representative Analog-to-Digital Setups .....	32
Setting Up Conditioned Input .....	32
Determine Bias Resistor To Center Span .....	34
Unipolar Variation .....	34
Choose Best Standard Resistor Values .....	34
Bracketing Input Range .....	34
Pick Proper Tolerance .....	36
Confirm Performance .....	36
Op-Amp Test Points .....	37
Calibrating the Analog-to-Digital Converter .....	38
Using Unconditioned Converter Channels .....	39
Serial Communication Ports .....	41
RS-485 .....	41
RS-232 and Programming Ports .....	42
Modem Communication .....	42
Power-Supervisor Integrated Circuit .....	43
Real-Time Clock (RTC) .....	44
Relay Outputs .....	44
Keypad and Liquid Crystal Display .....	46
Using the Keypad and Display .....	46
PK2400 Keypad .....	46
PK2400 Liquid Crystal Display .....	48
Bitmapped Graphics .....	48

## **Chapter 4: System Development 51**

Beginning Development .....	52
Operating Modes .....	53
Program Mode .....	53
Run Mode .....	53
Developing with the RS-232 Port .....	53
Developing with Serial Interface Board .....	55
Running a Program Standalone .....	56
Returning To Programming Mode .....	56
Developing an RS-485 Network .....	57

<b>Chapter 5: Software Reference</b>	<b>59</b>
Software Development Options .....	60
Dynamic C Development Software .....	60
Dynamic C Manuals .....	60
Supplied Software .....	60
Memory .....	61
Flash EPROM Functions .....	61
Nonvolatile Storage .....	62
Digital Input and Output .....	63
Digital Input/Output Functions .....	63
Advanced Input/Output Programming .....	66
U2 (PIO No. 1) .....	66
U9 (PIO No. 2) .....	66
Serial Communication Functions .....	67
RS-485 Functions .....	67
Analog-to-Digital Converter Drivers .....	68
Using the Liquid Crystal Display and Keypad .....	69
Real-Time Clock Integrated Circuit .....	76
Global Time and Date Structure .....	76
Sample Programs .....	78
Additional Software .....	78
<b>Appendix A: Troubleshooting</b>	<b>79</b>
Out of the Box .....	80
Dynamic C Will Not Start .....	81
Dynamic C Loses Serial Link .....	81
PK2400 Repeatedly Resets .....	81
Common Programming Errors .....	82
<b>Appendix B: Specifications</b>	<b>83</b>
Specifications .....	84
Hardware Dimensions .....	86
Suggested Operating Temperature .....	89
Power .....	89
Serial Interface Board .....	89
Jumper and Header Specifications .....	90
J12, RS-232, and RS-485 Ports .....	90
Processor Module Jumpers .....	91
Processor Module Gain and Bias Resistors .....	91
RS-232 Programming .....	91

<b>Appendix C: Power Management</b>	<b>93</b>
Direct Current Input .....	94
Power Regulator .....	94
Power Failure .....	94
Power Failure Sequence of Events .....	95
Multiple Power-Line Fluctuations .....	95
Holdup Time .....	96
Recommended Power-Failure Routine .....	97
<b>Appendix D: Input / Output Map and Interrupt Vectors</b>	<b>99</b>
Memory Map .....	100
Input / Output Map .....	100
Interrupt Vectors .....	101
Interrupt Priorities .....	102
<b>Appendix E: Serial Interface Board</b>	<b>103</b>
Features .....	104
External Dimensions .....	105
<b>Appendix F: Battery</b>	<b>107</b>
Storage Conditions and Shelf Life .....	108
Instructions for Replacing the Lithium Battery .....	108
Battery Cautions .....	109
<b>Index</b>	<b>111</b>
<b>Keypad Template</b>	

# ABOUT THIS MANUAL

---

This manual provides instructions for installing, testing, configuring, and interconnecting the Z-World PK2400 controller.

Instructions are also provided for using Dynamic C functions. Complete C and Dynamic C references and programming resources are referenced when necessary.

## Assumptions

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

- Ability to design and engineer the target system that interfaces with the PK2400.
- Understanding of the basics of operating a software program and editing files under Windows on a PC.
- Knowledge of the basics of C programming.



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

***The C Programming Language*** by Kernighan and Ritchie  
***C: A Reference Manual*** by Harbison and Steel

- Knowledge of basic Z80 assembly language and architecture.



For documentation from Zilog, refer to the following texts.

***Z180 MPU User's Manual***  
***Z180 Serial Communication Controllers***  
***Z80 Microprocessor Family User's Manual***

# Acronyms

Table 1 lists and defines the acronyms that may be used in this manual.







**Table 1. Acronyms**

Acronym	Meaning
EPROM	Erasable Programmable Read-Only Memory
EEPROM	Electrically Erasable Programmable Read-Only Memory
LCD	Liquid Crystal Display
LED	Light-Emitting Diode
NMI	Nonmaskable Interrupt
PIO	Parallel Input/Output Circuit (Individually Programmable Input/Output)
PRT	Programmable Reload Timer
RAM	Random Access Memory
RTC	Real-Time Clock
SIB	Serial Interface Board
SRAM	Static Random Access Memory
UART	Universal Asynchronous Receiver Transmitter

# Icons

Table 2 displays and defines icons that may be used in this manual.

**Table 2. Icons**

Icon	Meaning	Icon	Meaning
	Refer to or see		Note
	Please contact	<b>Tip</b>	Tip
	Caution		High Voltage
	Factory Default		



# Conventions

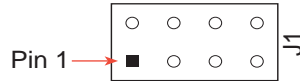
Table 3 lists and defines the typographic conventions that may be used in this manual.

**Table 3. Typographic Conventions**

Example	Description
<b>while</b>	Courier font (bold) indicates a program, a fragment of a program, or a Dynamic C keyword or phrase.
// IN-01...	Program comments are written in Courier font, plain face.
<i>Italics</i>	Indicates that something should be typed instead of the italicized words (e.g., in place of <i>filename</i> , type a file's name).
<b>Edit</b>	Sans serif font (bold) signifies a menu or menu selection.
...	An ellipsis indicates that (1) irrelevant program text is omitted for brevity or that (2) preceding program text may be repeated indefinitely.
[ ]	Brackets in a C function's definition or program segment indicate that the enclosed directive is optional.
< >	Angle brackets occasionally enclose classes of terms.
a   b   c	A vertical bar indicates that a choice should be made from among the items listed.

## Pin Number 1

A black square indicates pin 1 of all headers.



## Measurements

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

This page is blank intentionally.



## *CHAPTER 1: OVERVIEW*

---

## Introduction

The PK2400 consists of a processor module and a graphic liquid crystal display (LCD) integrated with a front panel and an interface board. The entire unit measures only 7.75" × 4.0" × 1.125". This slim package is ideal for an application that requires a powerful, compact controller with built-in operator interface. The front panel has an overlay containing a display window and a keypad for operator entry. The interface board contains the input and output connectors and the circuitry to interface the controller with the keypad and the display.

The keypad hosts snap-dome membrane switches that provide the user with tactile feedback. All keys are programmable, allowing the user to build a custom interface. The keypad legend and function key array can be easily changed. The three soft function keys (softkeys) allow for input to prompts on the LCD and are software programmable.

The PK2400's steel enclosure is a mylar-coated lexan front panel with heat releasing top and bottom vents. The optional bezel gasket allows for waterproof mounting; the panel mount is 1/16" to 1/8" thick.

## Processor Module

The PK2400 processor module is a small, powerful board that is easily programmed using Dynamic C. Moreover, the processor module offers impressive processing power for a wide variety of control applications.

Real-time programs can be developed on any of the PK2400 series units in the target system without the necessity of expensive in-circuit emulators or logic analyzers.

All PK2400s protect data in static RAM and the real-time clock's contents (PK2400 only) with a backup battery. The PK2400 has eight high-current outputs, two of which are used by the nonlatching relays.

## PK2400 Features

- Nine protected digital inputs, pull-up and/or pull-down (-20 to +24 V)
- Two conditioned analog inputs
- Six high-current outputs
- Two nonlatching relays (2 A at 30 V DC or 0.5 A at 120 V AC)
- Built-in temperature sensor for sensing temperature near board
- 128 x 64 graphic LCD with software contrast control
- Display backlight with software on/off
- Keypad with 20 keys and 3 softkeys
- RS-232 serial channel
- RS-485 serial channel
- Switching power supply
- Buzzer
- 128K static RAM (SRAM)
- 128K flash EPROM
- Real-time clock

## PK2410 Features

- All features of the PK2400 except relays, thermistor, switching power supply, analog inputs, and real-time clock
- Two additional high-current outputs

## Options and Upgrades

- Serial Interface Board (SIB) with programming cable
- Aluminum bezel, includes gasket and mounting clips (8.5" × 4.75" × 0.125")
- 256K flash EPROM
- Wall mount cold-rolled steel enclosure (7.85" × 4.1" × 1.5")

## Development Kit

- User's technical reference manual with schematics
- Power supply (AC adapter not in international kits)
- Programming cable

## System Development Programming Tools

Application programs for the PK2400 are developed using Dynamic C, Z-World's real-time C language development tool. Dynamic C is a version of the industry standard C programming language that has been optimized for development of real-time, multitasking control systems. Dynamic C runs under Windows on an IBM PC or compatible. As an integrated software-development system, Dynamic C provides an easy-to-use editor, compiler, and interactive debugger, which eliminate the need for expensive test equipment like third-party debuggers or in-circuit emulators. Dynamic C furnishes libraries of pre-written functions and software drivers that make software development fast and convenient.

When a program compiles, the host PC downloads the executable code, via the processor module's RS-232 port, directly to the onboard flash EPROM. This feature allows fast in-target development and debugging.

Another method for downloading programs from a host PC to a PK2400 is via a Z-World Serial Interface Board (SIB). By using the optional SIB, the RS-232 port is left free for other applications.

The PK2400 supports up to 256K of electronically reprogrammable flash EPROM. Flash EPROM allows programs to be downloaded into nonvolatile memory without using an EPROM burner.



Z-World's Dynamic C reference manuals provide complete software function descriptions and programming instructions.

## Factory Upgrades

In large quantities, the PK2400 can be factory customized with one or two latching relays. A latching relay replaces a nonlatching relay and uses an additional high-current output. In other words, a latching relay uses two high-current outputs, and a nonlatching relay uses only one high-current output.

Contact a Z-World Sales Representative for pricing information.



For ordering or customizing information, call a Z-World Sales Representative at (530) 757-3737.

## CE Compliance

The PK2400 has been tested by an approved competent body, and was found to be in conformity with applicable EN and equivalent standards. Note the following requirements for incorporating the PK2400 in your application to comply with CE requirements.



- The power supply provided with the Development Kit is for development purposes only. It is the customer's responsibility to provide a clean DC supply to the controller for all applications in end-products.
- Fast transients/burst tests were not performed on this controller. Signal and process control lines longer than 3 m should be routed in a separate shielded conduit.
- The PK2400 has been tested to Light Industrial Immunity standards. Additional shielding or filtering may be required for an industrial environment.



Visit the “Technical Reference” pages of the Z-World Web site at <http://www.zworld.com> for more information on shielding and filtering.

This page is blank intentionally.





## *CHAPTER 2: **GETTING STARTED***

---

Chapter 2 provides instructions for connecting the PK2400 to a host PC and running a sample program.

## Development Kit Packing List

The Development Kit includes items necessary for PK2400 software and hardware development. The kit includes the following items:

- Wall power supply (12 V DC) (not included in kits sold outside North America)
- Programming serial cable
- A reference manual with schematics

## Connecting the PK2400 to a Host PC

The PK2400 can be connected to a host PC via the J12 screw terminal strip or via Serial Interface Board (SIB). Although the ideal development method is with a SIB2, screws 9 through 13 on screw terminal strip J12 comprise the PK2400's development serial terminal. When a SIB is used, J12 is available for user applications.

PK2400 connectors are individual screw terminals that are not polarized or keyed. Carefully observe terminal identification and wire alignment before making a connection and before applying power.

Using the programming cable provided in the development kit, connect the PK2400 to a host PC with the following steps.

1. Disconnect power source to the PK2400.
2. Make sure a jumper is installed on pins 1 and 2 of header J1 on the processor module. The PK2400 is shipped with the required jumper factory-installed on pins 7 and 8 of header J1.
3. Connect the programming serial cable wires to screw terminal strip J12 as described in Table 2-1 and shown in Figure 2-1. Use caution when inserting wires into J12 and/or J13 in order to prevent short-circuiting the board.
4. Connect the serial cable's DE-9 connector to the host PC COM port.
5. Reconnect power source. The PK2400 is now in RS-232 programming mode.



**Caution:** Hazardous voltage is present in the inverter circuit.

Table 2-1 lists connection points between the PK2400 and the host PC.

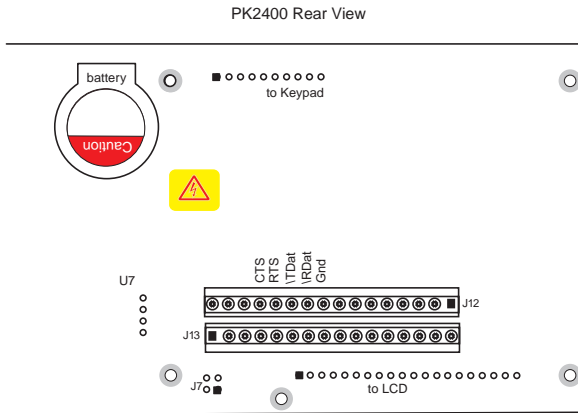
**Table 2-1. RS-232 Programming Cable Connections**

PC		Signal Direction	Serial Cable Wire Number	PK2400		
RS-232 Signal	DE-9 Pin Number			Serial Cable Signal	J12 Connection	Signal Name
DCD	1	←	1	None	nc	—
DSR	6	←	2	None	nc	—
RD	2	←	3	TXD	11	\TDAT
RTS	7	→	4	CTS	13	CTS
SD	3	→	5	RXD	10	\RDATA
CTS	8	←	6	RTS	12	RTS
DTR	4	→	7	None	nc	—
RI	9	←	8	None	nc	—
SG	5	—	9	None	9	GND

**Nomenclature**

CTS	Clear to Send	RI	Ring Indicator
DCD	Data Carrier Detect	RTS	Request to Send
DSR	Data Set Ready	SD	Send Data
DTR	Data Terminal Ready	SG	Signal Ground
RD	Receive Data	nc	no connection

Figure 2-1 illustrates the location of screw terminal strip J12.



**Figure 2-1. PK2400 Screw Terminal Strips**



Because of the proximity of solder pins, use caution when inserting wires into J12 and/or J13 in order to prevent a short circuit on the board.

## Establishing Communication

After hardware is connected, communication can be established between a host PC and the PK2400. Establish communication by double-clicking the Dynamic C icon to start the software.



Each time Dynamic C starts up, communication with the PK2400 is attempted.

If no error messages are displayed, communication has been successfully established.



If an error message such as **Target Not Responding** or **Communication Error** is shown, see Appendix A, “Troubleshooting.”

After making the necessary changes to establish communication between the host PC and the PK2400, use the Dynamic C shortcut **<Ctrl Y>** to reset the controller and initialize communication.

## Steps to Run a Sample Program

1. Open the sample program **DEMO\_RT.C** located in Dynamic C directory **Samples**.
2. Compile the program by pressing **F3** or by choosing **Compile** from the **Compile** menu. Dynamic C compiles and downloads the program into the PK2400’s flash memory.



During compilation, Dynamic C rapidly displays several messages in the compiling window. This condition is normal.

3. Run the program by pressing **F9** or by choosing **Run** from the **Run** menu.



If an error message such as **Target Not Responding** or **Communication Error** is shown, see Appendix A, “Troubleshooting.”

4. To halt the program execution, press **<Ctrl Z>**.
5. To restart program execution, press **F9**.



Programming with a SIB is described in Chapter 4, “System Development.” The SIB is not included in the standard Development Kit and must be purchased separately.



## CHAPTER 3: **SUBSYSTEMS**

---

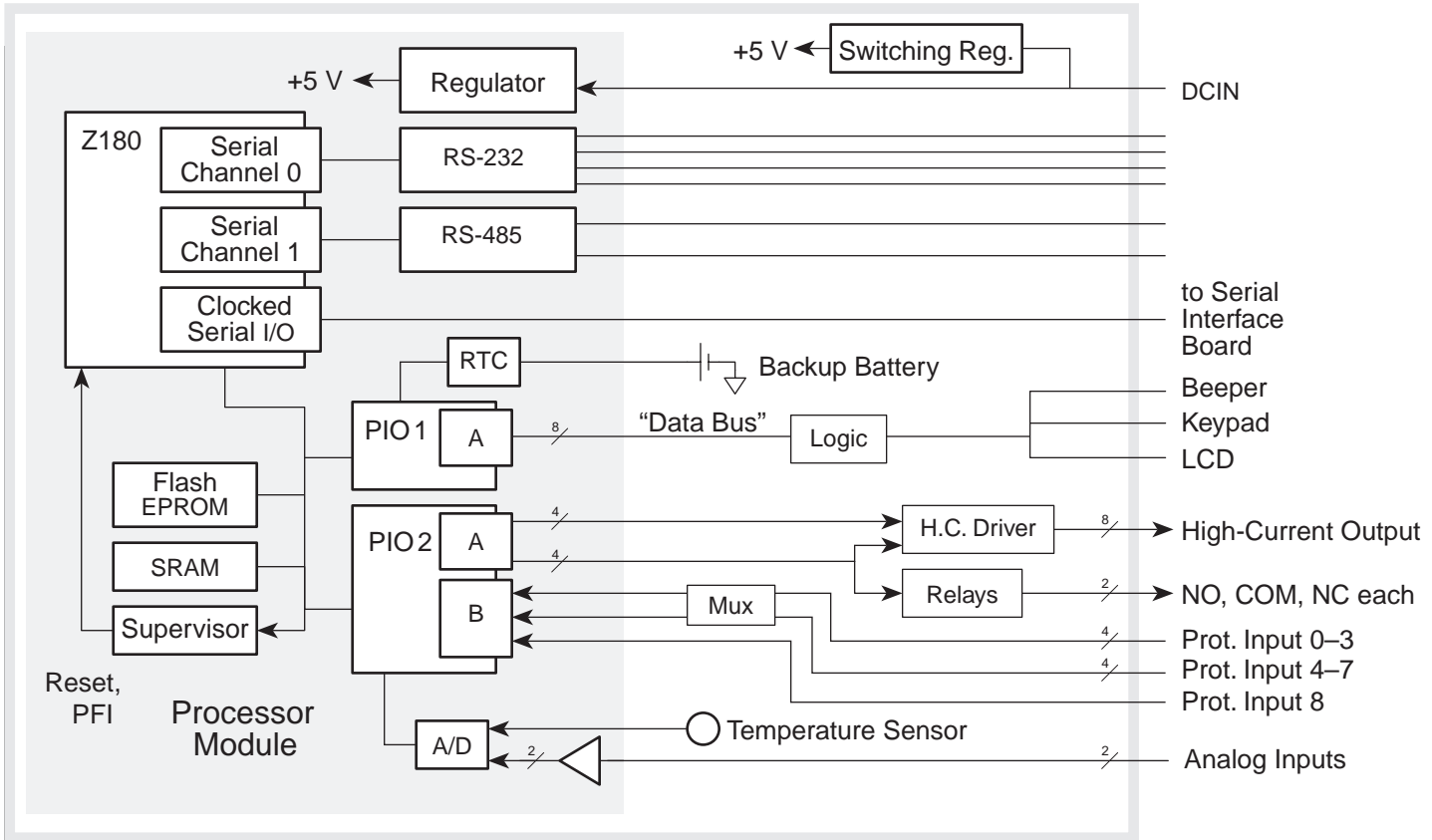


Figure 3-1. PK2400 Block Diagram

# Processor Module

## *Interface Overview*

The PK2400 uses a processor module as the “brains” of the system and as the source for inputs and outputs. This section describes the resources provided by the processor module and the interfaces between the PK2400 main circuit board and that module. Figure 3-1 is a block diagram of the PK2400 system.

## *Digital Input/Output*

The module uses two Zilog PIO integrated circuits to provide digital I/O signals. Some of the digital I/O signals are used in conjunction with the PK2400 main board to provide digital I/O to the user. Digital I/O software drivers relieve the user from having to understand the details of programming the PIOs. The digital I/O signals are brought to the PK2400 main board via headers. The digital I/O signals are connected to protection circuits and high-pow drivers to provide improved I/O capability for the PK2400.

## **Digital Inputs**

The PK2400’s nine protected digital inputs (Pin 00 through Pin 08) are flexible and robust.

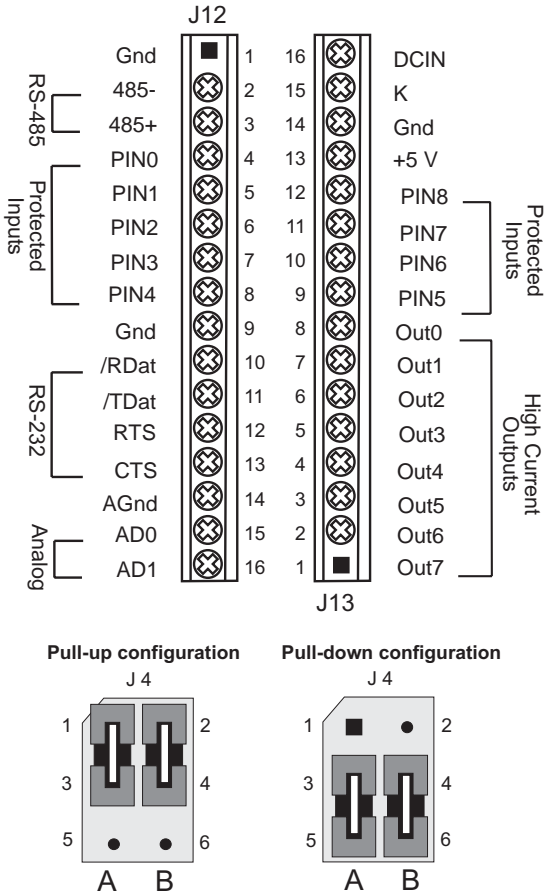
High-voltage protection circuits allow the inputs to detect switch contacts, relay contacts, outputs from open-collector transistor devices, logic-level outputs, and high-voltage outputs. The protected digital inputs have the following features:

- Nominal input voltage range of  $-20\text{ V}$  to  $+24\text{ V}$
- Protection against overloads over the range of  $-48\text{ V}$  to  $+48\text{ V}$
- Logic-level detection

The nominal voltage range for the protected digital inputs is  $-20\text{ V}$  to  $+24\text{ V}$ . The inputs are protected against overvoltages in the range  $-48\text{ V}$  to  $+48\text{ V}$ , but should not be regularly subjected to voltages outside the nominal voltage range.

Logic-level signals can be detected using any of the digital inputs. The logic threshold is nominally  $2.5\text{ V}$ . The maximum guaranteed low voltage is  $1.25\text{ V}$ . The minimum guaranteed high voltage is  $3.75\text{ V}$ .

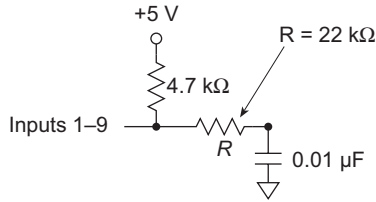
The digital inputs may be pulled up to 5 V or down to ground. The impedance of the digital input lines is 4.7 kΩ over 0 to 5 V. Outside this range, the input impedance is greater than 3.9 kΩ. Figure 3-2 illustrates header J4 pull-up and pull-down configurations. A jumper in position A affects protected inputs PIN0 through PIN3 on the J12 screw terminal strip. Jumpers in position B affect protected input PIN4 on J12 and PIN5 through PIN8 on the J13 screw terminal strip.



**Figure 3-2. Digital Input Jumper Settings**



Figure 3-3 illustrates a typical digital input line.



**Figure 3-3. Typical Digital Input**

### Digital Outputs

The PK2400's eight digital outputs (Out0 through Out7) provide high-voltage, high-current digital outputs. Two outputs repeat the condition of the nonlatching relays, and therefore are not independently usable. See the "Relays" section in this chapter for more information. Sourcing and optional sinking drivers will drive a variety of loads, including inductive loads such as relays, small solenoids, or stepping motors.

Figure 3-2 illustrates the location of the high-current digital output screw terminals on screw terminal strip J13.

Note the following points regarding the digital outputs:

- Each output is individually addressable.
- Each output includes a protective diode that returns inductive spikes to the power supply.
- Sourcing drivers are standard. Sinking drivers are optional.

The total number of outputs that can be on simultaneously is subject to chip power limits and ambient temperature. There are power limitations on each channel as well as on the entire driver chip. Eight channels (Out0 through Out7) are driven by one driver chip.

## Sourcing Driver

Figure 3-4 illustrates the connection for the UDN2985A sourcing driver.

Note the following points regarding the UDN2985A sourcing driver:

- Outputs pull high (source current) when turned on.
- The chip's rating is 30 V and 250 mA maximum per channel, subject to the chip's thermal limits and ambient temperature.
- With all channels on, each channel can source up to 170 mA continuously (100 percent duty cycle) as long as the chip temperature is less or equal to 50°C. At 70°C the current must be reduced to 140 mA or less.

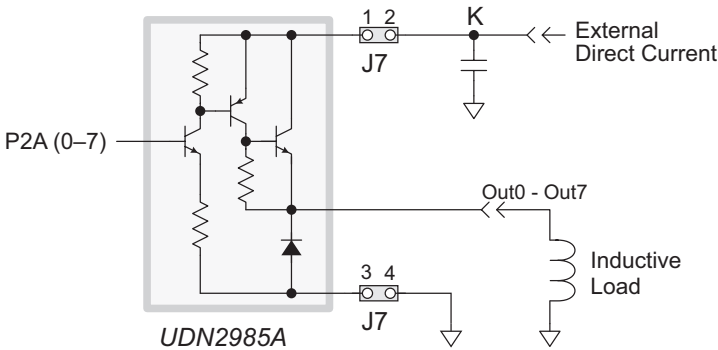


Figure 3-4. Sourcing Driver Configurations

## Sinking Drivers (Optional)

Figure 3-5 illustrates the configuration for the ULN2803 sinking driver:

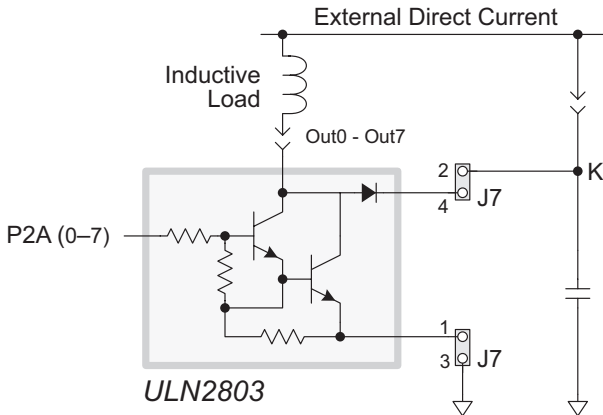
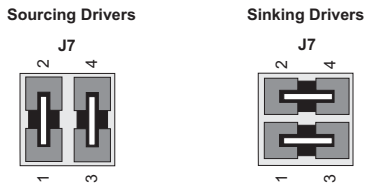


Figure 3-5. Sinking Driver Configuration

Note the following points regarding the ULN2803 sinking driver chip:

- Outputs pull low (sink current) when turned on.
- The chip's rating is 48 V and 500 mA maximum per channel, subject to the chip's thermal limits and ambient temperature.
- With all channels on, each channel can sink up to 170 mA continuously (100 percent duty cycle) as long as the chip temperature is less or equal to 50°C. At 70°C the current must be reduced to 140 mA or less.

Jumper J7 configures the outputs for either sourcing or sinking drivers. Figure 3-6 illustrates the J7 settings for sinking and sourcing drivers.



**Figure 3-6. J7 Digital Output Jumper Settings**

**Tip** Use a series resistor to limit the in-rush current when driving incandescent lamps.



See specifications section for more detailed information on the sinking and sourcing drivers.

## Analog-to-Digital Converter

The optional A/D converter is a 12-bit, serial-I/O, switched-capacitor, successive-approximation converter that can monitor temperature, measure position, or sense other types of analog signals. The A/D converter is onboard the processor module. The A/D converter's internal multiplexer samples and converts one input channel at a time.

The A/D converter provides four channels of 12-bit analog-to-digital conversion. Two of the channels have op-amps for signal conditioning and two are unconditioned. The conditioned channels are brought out to the PK2400 as analog inputs. One of the unconditioned channels is used for an onboard thermistor. The second unconditioned channel is connected to an onboard reference network.

Based on the A/D converter's minimum conversion period, the maximum data conversion rate is approximately 5,000 conversions per second. The actual level of performance will be less because application programming has a major effect on conversion rate.

### Extra Conversion

The A/D converter sends converted data out serially, and receives commands serially. During each serial-clock cycle, the chip shifts in one command bit and shifts out one data bit. This combined shifting accounts for the device's behavior of returning a previous data reading automatically each time it accepts a conversion or configuration command.

The A/D converter communicates with the Z180 via data-input line ADDIN, the data-output line ADDOUT, the clock line ADCLK, the end-of-conversion line EOC, and the chip-select line /ADCS.

EOC goes to a logic "0" level during conversion, returning to "1" when the conversion is complete. Following the conversion period, the A/D converter shifts the resulting data one bit at a time over ADDOUT. Also, during each shift-clock period, the converter shifts in one bit of a command word into the converter over ADDIN. This command word specifies the converter's next operation. The PIO's ARDY line drives /ADCS. This line goes from a logic "1" to "0" during power-on initialization as a result of putting Port A into Mode 3 operation. This transition is necessary for the chip to function properly. Thereafter, the line is left low.

EOC comes in over PIO Port B data line 3 (P2B3), which also senses /NMI through the logic gate U7. (/NMI and EOC are not at a logic 0 level at the same time and the driver software can distinguish between them in context.)

Library routines take care of all the low-level details of communication with the A/D converter automatically.



Because the protocol for controlling the serial A/D converter is complex, Z-World strongly recommends using the library functions to control the converter instead of writing custom functions.

### Voltage Reference

The A/D converter's two reference inputs, REF<sup>-</sup> and REF<sup>+</sup>, establish the voltage limits for analog inputs that produce the maximum and minimum conversion values. Inputs higher than REF<sup>+</sup> return the maximum conversion value, and inputs less than REF<sup>-</sup> return the minimum conversion value. The A/D converter has *no* out-of-range signal. Software *will not* be able to distinguish between an input that is exactly at either limit of the voltage range and one that exceeds the limits.

## Data Conversion

The two conditioned inputs measure input signals over either a bipolar or unipolar voltage range. In either case, the operational amplifier's gain and bias resistors scale the signal ranges to conform to the 0 to 2.5 V input range of the A/D converter. Because of the inverting configuration of the op-amps, the maximum input voltage results in a minimum input voltage at the converter.

The A/D converter determines a 12-bit digital value representing the converted value of the input voltage. An input voltage at the A/D integrated circuit (equal to 0 V) converts to all zeros, and an input at 2.5 V converts as all ones. Dynamic C functions return 16-bit sign-extended values. The functions return a reading appropriate to the unipolar or bipolar signals being measured, based on the arguments supplied to the function.

## Limitations on Output Range

In actual practice, the op-amp outputs can only approach ground (0 V) but cannot actually reach it. The output low-voltage limit is about 10 mV to 20 mV. The practical effect of this limitation is that approximately 0.4 through 0.8 percent of the upper end of the input-signal range is unusable. For example, if the input signal range is 0 to 10 V, the maximum useful input voltage is 0.02 to 9.96 V.

## Low-Pass Filter

The 0.01  $\mu\text{F}$  feedback capacitors in the op-amp's circuits transform the op-amps into low-pass filters to attenuate any high-frequency noise that may be present in the signal. The filters' characteristics depend on the resistors selected. Equation 3-1 illustrates how to calculate the 3-dB corner frequency:

$$f_{3\text{dB}} = \frac{1}{2\pi \times R_g \times 0.01 \mu\text{F}} \quad (3-1)$$

The 3-dB corner frequency, then, is 6715 kHz for this case with a gain of 0.25 and a 1 percent feedback resistor of 2370  $\Omega$ .

## Internal Test Voltages

By addressing "virtual" channels in the A/D converter, Dynamic C routines can obtain internal test voltages from the A/D converter. However, these readings measure VREF+ and VREF- with respect to VREF and GND so that the resulting conversions are all zeros or all ones.

## Drift

The AD680JT voltage reference exhibits a voltage drift of 10 ppm/°C (typical) to 30 ppm/°C (maximum). This drift corresponds to 25 to 75 mV/°C or 1.75 to 5.25 mV over the temperature range of 0°C to 70°C.

The LMC662C operational amplifier exhibits an offset-voltage drift of 1.3 mV/°C (typical), or 910 mV over temperature. A greater contribution to overall drift arises from differences in the temperature coefficients of the user-installed gain and bias resistors and the fixed 10 kΩ resistors R21 through R24. Resistors R21 through R24 have temperature coefficients of ±200 ppm/°C. Since the surface-mount resistors are close to each other, they are always at essentially the same temperature and their temperature deviations track closely.

## Absolute Mode

The A/D converter operates in absolute mode. In absolute mode, the A/D converter compares the input signal against an accurate, stable, onboard reference. The onboard voltage-reference integrated circuit is an AD680JT, which has a drift specification of 10 ppm per °C typical.

The A/D converter is configured to operate in absolute-conversion mode using an onboard voltage reference U4, as shown in Figure 3-7. REF– is hard-wired to ground. REF+ connects to one of two sources of 2.5 V. The A/D is factory-configured for absolute-conversion mode with REF+ connected to a precision voltage reference, usually an AD680, U4 (but alternatively an LM385, U5).

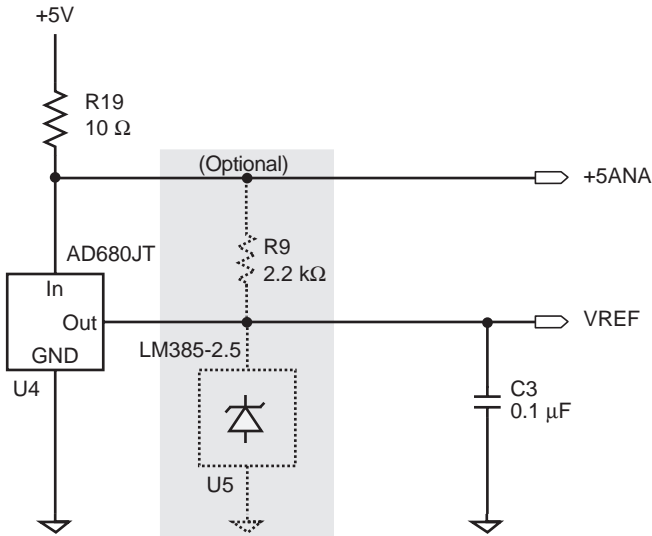


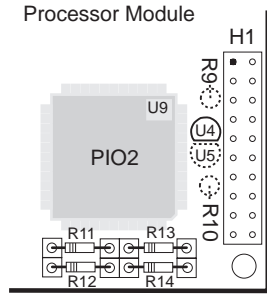
Figure 3-7. Analog Reference in Absolute Conversion Mode

## Bipolar or Unipolar Conditioned Inputs

Bipolar or unipolar signals can be measured using the two conditioned inputs.

Signals from sensors connected to AD0 and AD1 go to the inverting input of one of two op-amps. The op-amps operate in the inverting configuration. User-selectable resistors R11 through R14 on the processor module set the gain and bias voltages of the amplifiers. The 10 k $\Omega$  input and bias resistors R21 through R24 are fixed. Feedback capacitors C20 and C21 roll off the high-frequency response of the amplifiers to attenuate noise.

Figure 3-8 illustrates the location of the processor module conditioned analog input gain and bias resistors R11 through R14.



**Figure 3-8. Gain and Bias Resistors**

## Factory Gain and Bias Resistors

The A/D converter is configured with gain and bias resistors installed in the conditioned A/D channels (configured for 0 V to 10 V direct current inputs) so that the controller works right out of the box. If a different input voltage range is required, change the gain and bias resistors. See Chapter 4 for information on accessing the processor module.

## Initial Setup

The op-amp's gain and bias resistors, R11 through R14, are installed in sockets provided on the processor module.

- $R_{\text{GAIN}}$ , R11 and R13 = 2370  $\Omega$
- $R_{\text{BIAS}}$ , R12 and R14 = 39.2 k $\Omega$



Refer to sections under “Setting Up Conditioned Input” in this chapter for a detailed method of determining the best values for gain and bias resistors.

Resistors R11 through R14 yield a nominal gain of 0.25 volts for a unipolar input-signal range of 0 to 10 volts. These values slightly differ from theoretical values. This discrepancy is derived from real-world resistor tolerances.

Strip sockets with 0.300" centers accommodate one-eighth watt resistors R11 through R14.

## Representative Analog-to-Digital Setups

Table 3-1 gives the values of gain and bias resistors for various common input-voltage ranges in the absolute-conversion mode using the onboard voltage reference. These values, which require standard one percent resistors, have been adjusted from theoretical values to account for tolerance variations. If none of these setups suits the application under development, proceed to the next section and follow the design method presented there to calculate the resistor values required.

**Table 3-1. Gain and Bias Resistor Input Voltage Ranges**

Input Range (V)	Gain	R <sub>GAIN</sub> (kΩ)	R <sub>BIAS</sub> (kΩ)
-10.0 to +10.0	0.125	1.18	8.06
-5.0 to +5.0	0.250	2.37	6.65
-2.5 to +2.5	0.500	4.75	4.99
-2.0 to +2.0	0.625	5.90	4.53
-1.0 to +1.0	1.250	11.80	2.87
-0.5 to +0.5	2.500	23.70	1.69
-0.25 to +0.25	5.000	47.50	0.931
-0.10 to +0.10	12.500	1180.00	0.392
0 to +10.0	0.250	2.37	39.20
0 to +5.0	0.500	4.75	20.00
0 to +2.5	1.000	9.53	10.00
0 to +1.0	2.500	23.20	4.02



In production quantities, the PK2400 can be ordered with surface-mount resistors installed on the processor module in place of the R11 through R14 resistors. Contact Z-World at (530) 757-3737 for details.

## Setting Up Conditioned Input

The gain and bias resistors (R11 through R14) determine the input signal's voltage relation to ground as well as its range. For example, assume a circuit must handle an input-signal range of 10 V spanning  $-5$  V to  $+5$  V.

Given this specification, select gain resistor R<sub>GAIN</sub> (R11 or R13) to suit the input-signal voltage range of 10 V.



The gain of the amplifier is the ratio of its maximum output-voltage swing to the application's maximum input-voltage swing. The fixed 2.5 V input range of the A/D converter limits the op-amp's output swings to 2.5 V.

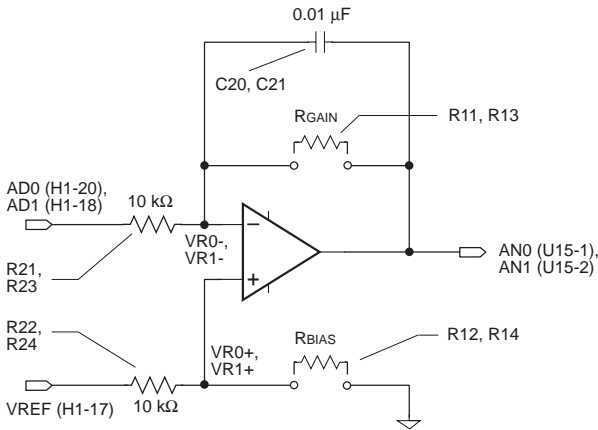
Equation 3-2 expresses an amplifier's gain in terms of its input-voltage range, where  $g$  is the gain,  $V_{IN\_MAX}$  is the maximum input voltage, and  $V_{IN\_MIN}$  is the minimum input voltage.

$$g = \frac{2.5 \text{ V}}{V_{IN\_MAX} - V_{IN\_MIN}} \quad (3-2)$$

The ratio of the user-specified gain resistor  $R_{GAIN}$  (R11 or R13) to its associated fixed-input resistor (R21 or R23) determines an amplifier's gain. The gain for the amplifier in Figure 3-9 with its input resistor fixed at 10 k $\Omega$  is

$$g = \frac{R_{GAIN}}{10,000}$$

**Figure 3-9. Conditioned Analog Input Diagram**



Given an input voltage range of 10 V, Equation (3-2) fixes the amplifier's gain at 0.25, and scales the input signal's range correctly to the op-amp's 2.5 V max output range.  $R_{GAIN}$  must therefore be 2500  $\Omega$ .

## Determine Bias Resistor To Center Span

If the op-amp is to servo its output properly around the desired center voltage, establish the appropriate bias voltage at the op-amp's non-inverting input. Select the bias, or offset, resistor  $R_{BIAS}$  ( $R_{12}$  or  $R_{14}$ ) to position the input-voltage range correctly with respect to ground (in this example,  $-5\text{ V}$  to  $+5\text{ V}$ ).

Because the value for  $R_{GAIN}$  has already been selected, the maximum input voltage ( $V_{IN_{MAX}}$ ) determines the maximum input voltage seen at the amplifier's summing junction (inverting input), circuit nodes  $VR_{0-}$  and  $VR_{1-}$ . Compute  $VR_{0-}$  or  $VR_{1-}$  using Equation (3-3).

$$VR_{0-} = V_{IN_{MAX}} \times \left( \frac{g}{1+g} \right) \quad (3-3)$$

The bias voltage,  $V_{BIAS}$ , must equal its corresponding  $VR_{n-}$  for each op-amp. A voltage divider consisting of a bias resistor,  $R_{BIAS}$  ( $R_{12}$  or  $R_{14}$ ), and a fixed  $10\text{ k}\Omega$  resistor ( $R_{22}$  or  $R_{24}$ ) derives this bias voltage ( $V_{BIAS} = VR_{0+}$  or  $VR_{1+}$ ) from  $V_{REF}$ , the  $2.5\text{ volt}$  reference voltage. Equation (3-4) gives  $R_{BIAS}$ .

$$R_{BIAS} = \frac{V_{BIAS} \times 10,000}{2.5 - V_{BIAS}} \quad (3-4)$$

The  $2.5\text{ V}$  term in the equation's denominator is the reference voltage. The low-impedance voltage reference supplies this voltage in the absolute conversion mode.

## Unipolar Variation

Suppose the input range is  $0$  to  $+10$  volts instead of  $-5\text{ V}$  to  $+5\text{ V}$ .  $V_{IN_{MAX}}$  is now  $+10\text{ V}$ ,  $V_{BIAS}$  becomes  $2.0\text{ V}$ , and  $R_{BIAS}$  is  $40\text{ k}\Omega$ .

## Choose Best Standard Resistor Values

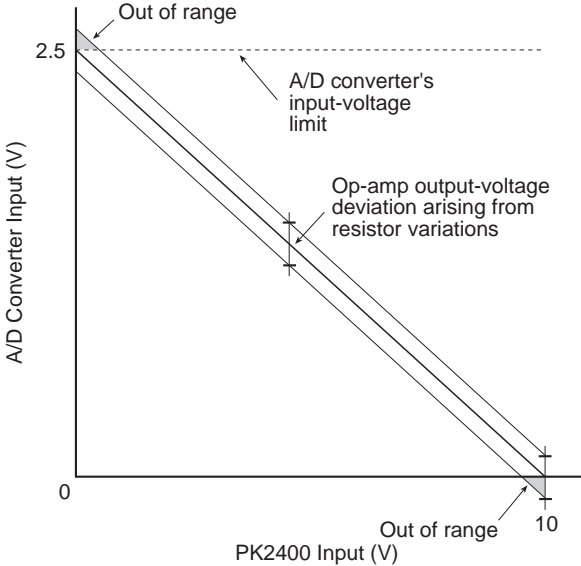
Calculated values will not always be available as standard resistor values. In these cases, use the nearest standard resistor value. For example, use  $6650\ \Omega$  rather than  $6667\ \Omega$  if using 1 percent resistors, or use  $6800\ \Omega$  if using 5 percent resistors.

## Bracketing Input Range

To be sure of accurately measuring signals at the extremes of an input range, be aware of the interaction between the  $10\text{ k}\Omega$  fixed resistors,  $R_{21}$  through  $R_{24}$ , and the other installed resistors,  $R_{11}$  through  $R_{14}$ . In the ideal case, the A/D converter's input would be at the maximum expected value of  $2.5\text{ volts}$  if a signal is measured at the minimum input level.

Resistor values vary within their rated tolerance bands. Thus, if the fixed input resistor has less resistance than its nominal value and the installed resistor has a resistance slightly higher than its nominal value, the actual input to the A/D converter would be more than 2.5 V. A loss of accuracy then results because the A/D input would reach its maximum input value before the true signal input reaches the minimum expected input level.

Figure 3-10 illustrates how variations in tolerance can cause the analog signal to exceed the limits of the analog converter.



**Figure 3-10. Out of Range A/D Converter Input**

Similarly, a deviation from nominal values in the bias network could skew the A/D converter's input voltage away from a theoretically computed value. For example, a small positive or negative deviation of the bias voltage arising from variances in the resistive divider would offset the A/D converter's input voltage. This offset would be positive or negative, tracking the deviation's sign, and equal to the bias deviation multiplied by the amplifier's gain plus one. Both of these effects could occur in the same circuit.

## Pick Proper Tolerance

Use care when compensating for any discovered discrepancies. For example, if standard 5 percent resistors are used for R11 through R14, recall that resistor values are spaced approximately 10 percent apart.



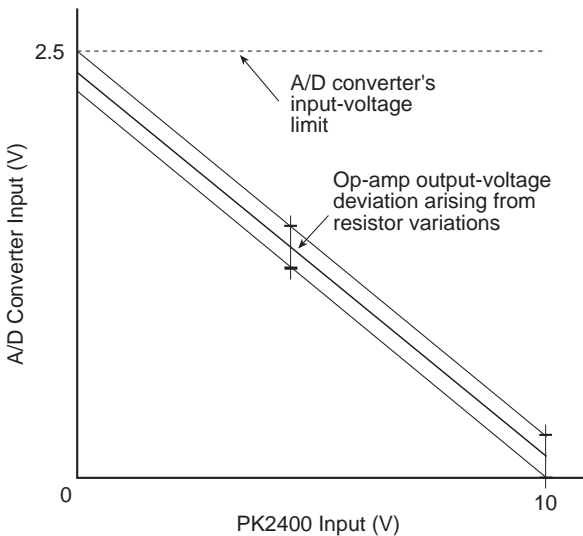
The tolerance is  $\pm 5$  percent; therefore, any value calculated will be within  $\pm 5$  percent of a standard value.

If a gain is too high by just a small amount, then going to the next smallest standard 5 percent value could result in a decrease in gain approaching 10 percent. The same caveat applies to the bias network. Use 1 percent resistors to get a more precise choice of values.



Use the mathematically derived values if the loss of signal range is acceptable.

Figure 3-11 illustrates the result of adjusting the resistor values so that input to the A/D converter stays within its specified 2.5 V range.



**Figure 3-11. Converter Input Within A/D Range**

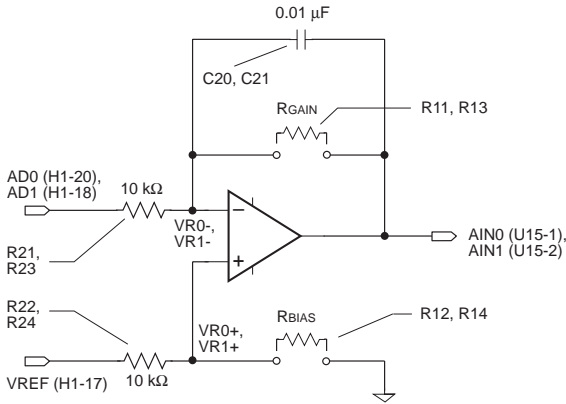
## Confirm Performance

If measurements are critical, check setups after installing resistors by measuring test signals at and near the input-voltage limits. See if voltages fall within the A/D converter's input range or if loss of accuracy occurs because of over-excursions at the A/D converter's input. Alternatively, measure the resistance of the factory-installed fixed resistors before selecting and measuring new resistors accordingly.

Fixed resistors can be measured indirectly after installing resistors by measuring the voltages at the amplifier’s inputs and outputs. Using Channel 0 as an example, ground the input AD0 at pin 20 of H1. Then measure the voltages at VR0– and the amplifier’s output. Because the currents through the input resistor and the feedback resistor are essentially identical, the ratio of the voltages across the resistors is equivalent to the ratio of the resistors. Therefore, the gain is as displayed in Equation (3-5):

$$gain = \frac{V_{OUT} - VR0-}{VR0-} \tag{3-5}$$

Again using Channel 0 as an example, measure the voltage of VREF and the voltage at VR0+ as displayed in Figure 3-12. Because the current into the op-amp input is negligible, the resistance ratio of the two resistors in the voltage divider alone determines VR0. Once both the value of the installed resistor and the value of VR0+ are known, compute the value of

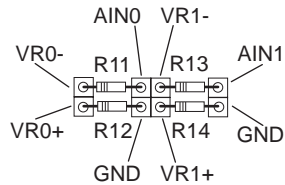


the fixed resistor in the divider.

**Figure 3-12. Voltage Test Points**

**Op-Amp Test Points**

The factory-installed fixed resistors have a 1 percent tolerance. Figure 3-13 shows convenient points on the processor module to make voltage measurements of the op-amps’ circuitry.



**Figure 3-13. Analog Input Test Points**

## Calibrating the Analog-to-Digital Converter

The inherent component-to-component variations of 5 or 1 percent resistors can “swamp” the 0.25 percent resolution of the A/D converter. To achieve the highest accuracy possible, calibrate the analog inputs.

The software drivers for the A/D converter provide routines to compute calibration coefficients (given two reference points) and store them in a defined location in nonvolatile memory. Each reference point is comprised from the following pair of values:

- Actual applied test voltage
- Raw converted A/D value (a 12-bit integer)

The supplied Z-World A/D software library **EZIOCMMN.LIB** automatically uses these coefficients to correct all subsequent A/D readings.



The A/D converter functions are described in Chapter 5, “Software Reference.”

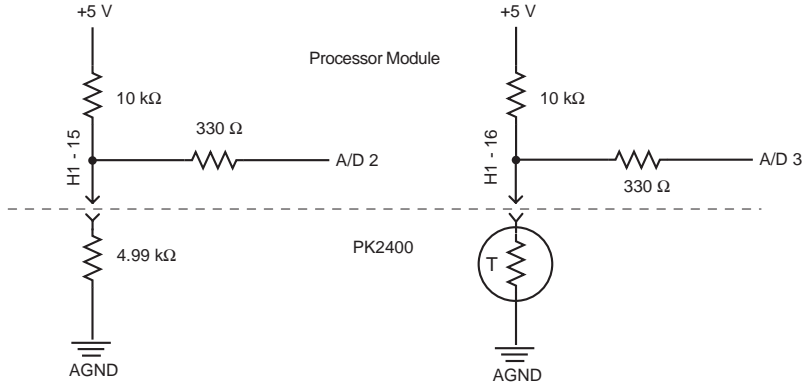
Table 3-2 lists the addresses in simulated EEPROM where the calibration constants are stored.

**Table 3-2. A/D Converter Calibration Constants**

Address in Simulated EEPROM	A/D Channel
10–15	0
16–21	1
22–27	2
28–33	3

## Using Unconditioned Converter Channels

Two additional input channels of the A/D converter are connected to an onboard thermistor and reference network as shown in Figure 3-14. The thermistor and reference resistor can be used to measure the onboard temperature.



**Figure 3-14. Thermistor Circuit**

Table 3-3 lists the Z-World thermistor outputs as a function of temperature.

**Table 3-3. Temperature to Ohms Conversion Table**

°C	°F	Ω	°C	°F	Ω
-70	-94	3,095,611	45	113	4,366
-65	-85	2,064,919	50	122	3,601
-60	-76	1,397,935	55	131	2,985
-55	-67	959,789	60	140	2,487
-50	-58	667,828	65	149	2,082
-45	-49	470,609	70	158	1,751
-40	-40	335,671	75	167	1,480
-35	-31	242,195	80	176	1,256
-30	-22	176,683	85	185	1,070
-25	-13	130,243	90	194	916
-20	-4	96,974	95	203	787
-15	5	72,895	100	212	678.6
-10	14	55,298	105	221	587.3
-5	23	42,314	110	230	510.1
0	32	32,650	115	239	444.5
5	41	25,395	120	248	388.6
10	50	19,903	125	257	340.8
15	59	15,714	130	266	299.8
20	68	12,493	135	275	264.5
25	77	10,000	140	284	234.1
30	86	8,056	145	293	207.7
35	95	6,530	150	302	184.8
40	104	5,324			

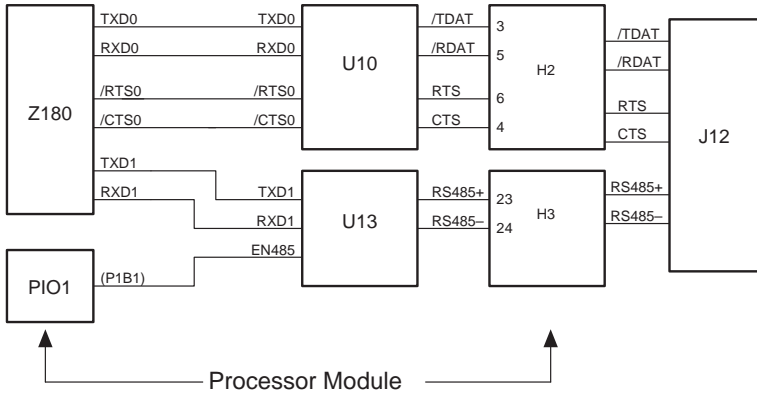


## Serial Communication Ports

Two Z180 serial ports support asynchronous communication at baud rates from 300 to 57,600 bits per second.

1. Port 0 is a 5-wire RS-232 port (with RTS and CTS).
2. Port 1 is a half-duplex RS-485 port, which provides half-duplex asynchronous communication over twisted pair wires to a distance of up to 4 kilometers.

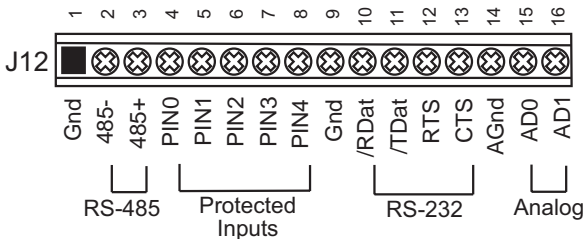
Figure 3-15 illustrates the configuration of Port 0 and Port 1.



**Figure 3-15. Serial Communication Port Configuration**

## RS-485

The processor module has a half-duplex RS-485 serial channel that is brought out to the screw terminal strips. The RS-485 signals are on screw terminal strip J12, screw terminal 2 (RS-485-) and screw terminal 3 (RS-485+). Figure 3-16 illustrates the RS-485 connections.



**Figure 3-16. J12 RS-485 Connections**

## RS-232 and Programming Ports

A five-wire RS-232 interface can be used either for programming or by an application. The RS-232 signals are on PK2400 screw terminal strip J12, screws 10 through 13 (the fifth wire goes to a ground). The PK2400 can also be programmed without using the RS-232 port. Connecting a Z-World Serial Interface Board (SIB) leaves the RS-232 interface available to an application during development.



See Z-World's Dynamic C manuals for specifics on modem communication software.

The PK2400's RS-232 support library is **EZIOPK24.LIB**. Support for serial communication includes the following functions:

- Initializing the serial ports
- Monitoring and reading a private circular receive buffer
- Monitoring and writing to a private circular transmit buffer
- CTS (clear to send) and RTS (request to send) control
- XMODEM protocol for downloading and uploading data
- A modem option
- An echo option

## Modem Communication

Modems and telephone lines allow RS-232 communication across great distances. With a modem, character streams that are read from the receive buffer are automatically scanned for modem commands. The RS-232 library supports communication with a Hayes Smart Modem or compatible. If the modem used is not truly Hayes Smart Modem compatible, the CTS, RTS, and DTR lines on the modem side must be tied together. The CTS and RTS lines on the PK2400 side also have to be tied together. A NULL connection is also required for the TX and RX lines. A commercial NULL modem would already have its CTS and RTS lines tied together on both sides.

The PK2400 supports the XMODEM protocol for downloading and uploading data. Currently, the library supports downloading an array of data that is a multiple of 128 bytes.

Uploaded data is written to a specified area in RAM. The targeted writing area should not conflict with the current resident program or data.

Character echo is automatically suspended during XMODEM transfer.



See Z-World's Dynamic C manuals for specifics on modem communication software.

## Power-Supervisor Integrated Circuit

The power-supervisor integrated circuit is a key component that helps a system survive power fluctuations and power outages. Several vital services provided by the power supervisor are described in the following list.

- **Power-on reset**

The supervisor integrated circuit generates the power-on reset for the PK2400 by holding /RESET low until the chip senses that VCC has risen above reset threshold ( $\gg 4.65$  V) 4.65 V and battery voltage (2.5 V to 4.25 V DC). Also, when VCC falls below threshold, the supervisor integrated circuit disables the RAM to prevent spurious writing of data.

- **RAM protection**

The power supervisor integrated circuit gates the decoded RAM-select line to the RAM's chip-enable line whenever VCC is above the reset threshold and VBAT. When VCC falls below the threshold, the ADM691 de-asserts the chip-enable to prevent spurious writing to the RAM.

- **Watchdog timer**

The watchdog timer cannot be disabled. The watchdog timer guards against system or software faults. If an application program does not "hit" the watchdog timer at least every 1.0 second, the watchdog timer resets the Z180. The supervisor's watchdog output (/WDO) connects to the Z180's /INT1 interrupt line. /WDO is at logic zero level after a watchdog reset and a logic 1 after a power-on reset.



To "hit" the watchdog timer, make a call to the library function **hitwd**. This call makes a dummy one-byte DMA transfer via DMA channel 1, which activates the DMA-end signal, /**TEND1**, "hitting" the watchdog timer.

- **Nonmaskable interrupt**

The supervisor integrated circuit generates a non-maskable interrupt (/NMI) from its power-fail output (/PFO) for the microprocessor if the unregulated direct current input (normally 9 V to 12 V DC) falls below 7.9 V. This gives the PK2400 advanced warning of an impending power failure, so that it can execute shutdown routines.

R16 introduces approximately 830 mV of hysteresis into the supervisor IC's sensing of the raw direct current, preventing noise on DCIN from generating repeated interrupts when the input voltage is low. /NMI also connects to port B (bit 3), of PIO2 (via IC U7B) to allow software to monitor /NMI line after the non-maskable interrupt, and recover from temporary low-input voltage conditions or "brownouts."

### • Backup-battery switchover

The supervisor integrated circuit switches the RAM over to battery power if VCC falls below the battery voltage VBAT (2.5 V to 4.25 V DC).

### Real-Time Clock (RTC)

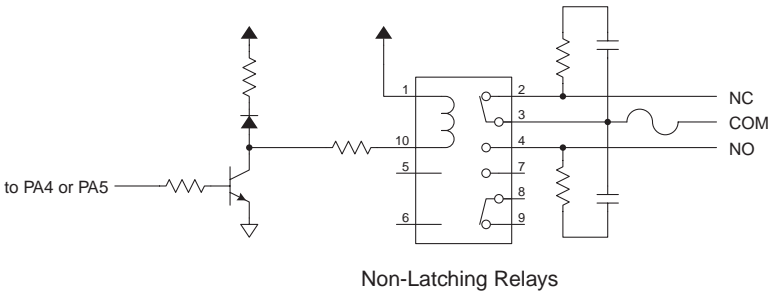
The PK2400 processor module has an RTC on board. The RTC provides time and date functions plus 31 bytes of scratchpad RAM. The RTC also has the following features:

- Automatically adjusts the last date of the month for the number of days in a month and accounts for leap years.
- Reports time in either 24- or 12-hour format (with an a.m. or p.m. indication).
- An external battery (2.5 V to 4.25 V DC) allows the integrated circuit to retain its time and data when power fails.

### Relay Outputs

The PK2400 has two nonlatching SPDT relay outputs capable of driving high-current loads. These outputs are illustrated in Figure 3-17.

The relay outputs are available to the user on screw terminal strip J11.



**Figure 3-17. Nonlatching Relay**

The relays can handle maximum load currents up to 2 A. There is a 2.5 A fuse between the common connection and the terminal to prevent damage from over-current. Each contact on the relays also has a snubber (resistor and capacitor) circuit to help protect the contacts from arcing.



For CE compliance, the maximum switching voltage is less than 50 V AC or 75 V DC.

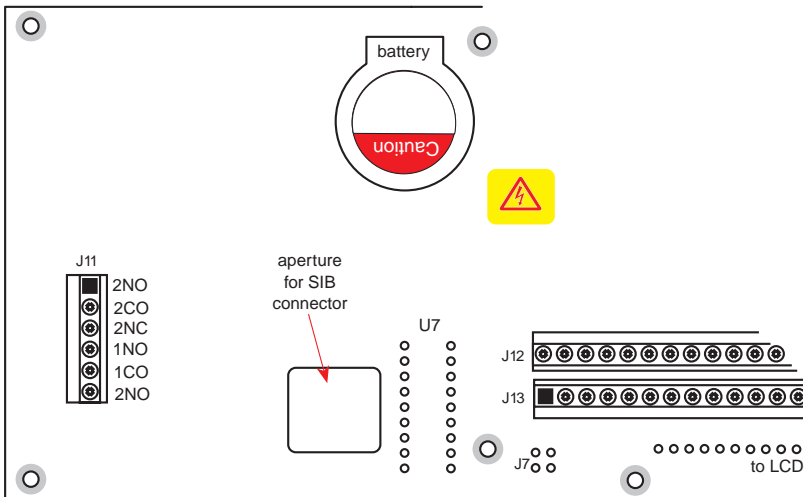
An appropriately rated varistor must be connected at the load if highly inductive loads are connected to the PK2400 relays. Switching off highly inductive loads without the use of a varistor may damage the snubber circuit and the relay. Installing the varistor in parallel with the snubber instead of across the load may result in unintentional energizing of the load if the varistor fails in shorted condition. If the varistor installed across the load fails in the shorted condition, the fuse will open and the load will not be energized.

Table 3-4 lists the terminal relay connections.

**Table 3-4. 23-Key Keypad Connections**

J11 Terminal Number	Relay 2 Connection	J11 Terminal Number	Relay 1 Connection
1	Normally open	4	Normally open
2	Common	5	Common
3	Normally closed	6	Normally closed

Figure 3-18 illustrates the location of screw terminal strip J11.



**Figure 3-18. J11 Screw Terminal Strip Location**

## Keypad and Liquid Crystal Display

The PK2400 supports operator input/output through the keypad and the LCD. Features include a 128 column by 64 row backlit graphic LCD module and a 4 row by 5 column keypad with three additional soft keys.

The graphic LCD has software controllable electroluminescent backlighting installed as a standard feature. The keypad connects through connector J-2.

### Using the Keypad and Display

The PK2400 keypad and display are supported by a large number of software drivers. The keypad and display can be used for a variety of user interface applications, for example:

- User code or password entry
- System status display
- Multiple language/character-set displays
- Parameter monitoring and adjustment

Because the three function keys are close to the LCD screen, it is possible to program on-screen prompts that correspond to the adjacent key.

### PK2400 Keypad

The keypad is designed to accept a paper insert. Inserts can be customized by photocopying the keypad template located behind the “Schematic” section in this manual.

Figure 3-19 provides measurements for the keypad template. All dimensions are in inches. Inserts can be secured by taping the portion of the insert that extends beyond the keypad to the supporting bracket.

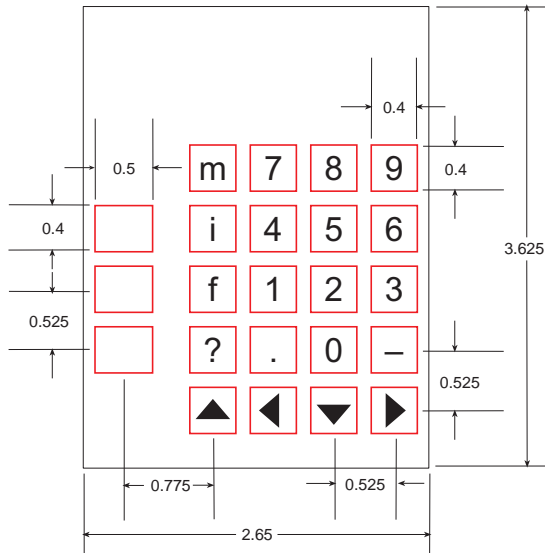
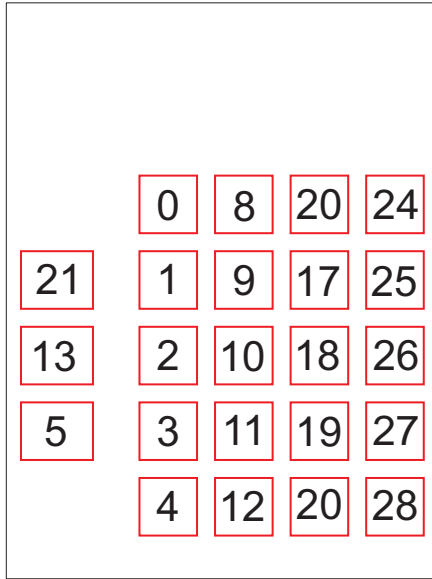


Figure 3-19. 4 x 5 Plus 3 Keypad Default Legend

Figure 3-20 illustrates the keypad connections in Table 3-5.



**Figure 3-20. Keypad Key Numbers**

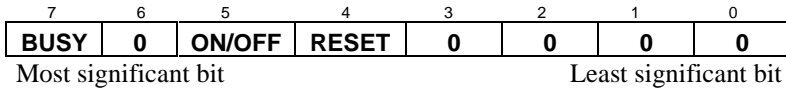
**NOTE:** Keys are numbered for orientation. They do not come labeled.

**Table 3-5. 23-Key Keypad Connections**

Key	Connection	Key	Connection
0	R0, C0	16	R2, C0
1	R0, C1	17	R2, C1
2	R0, C2	18	R2, C2
3	R0, C3	19	R2, C3
4	R0, C4	20	R2, C4
5	R0, C5	21	R2, C5
8	R1, C0	24	R3, C0
9	R1, C1	25	R3, C1
10	R1, C2	26	R3, C2
11	R1, C3	27	R3, C3
12	R1, C4	28	R3, C4
13	R1, C5		

## PK2400 Liquid Crystal Display

The PK2400 LCD is easy to use with Dynamic C software libraries. Several of the Dynamic C graphic library functions return the operating status of the LCD. The LCD status bits are shown in the following diagram.



**BUSY** - Reading a “1” indicates LCD is performing an operation. Reading a “0” indicates the LCD is ready to accept more data.

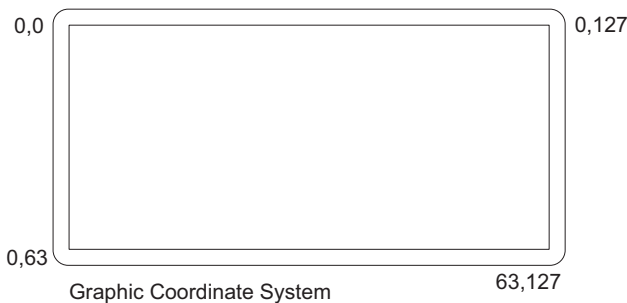
**ON/OFF** - When the ON/OFF bit is set (1) the display is on, any image on the screen will be visible. When the bit is reset (0) any images on the display will not be visible. The image is still in the display memory.

**RESET** - Resets the LCD module when low (0).

### Bitmapped Graphics

Many of the Dynamic C functions that operate on the graphic LCD use bitmaps. These bitmaps represent the images on a section of the display. Each dot, or pixel, is represented by one bit in the bitmap. If the pixel is on, the corresponding bit is set. If the pixel is off, the corresponding pixel is reset.

The image on the display is two dimensional (width and height). The bitmap used to store that display information is a one-dimensional array. Two-dimensional images are stored in column major, byte-aligned bitmap format. Figure 3-21 illustrates the two-dimensional graphic coordinate system.



**Figure 3-21. 64 x 28 Graphic Coordinates**



Column major means that bits are stored in the bitmap column by column. The first pixel of the first column (row 0, column 0) of the image is stored in the first bit position in the bitmap. The second pixel in the first column is stored in the second bit position in the bitmap and so on. When the entire first column is stored in the bitmap, the process begins again with the second column and repeats until all of the columns of the image are stored.

Byte aligned means that a column data will end on a byte boundary. If a column has a number of bits that is not evenly divisible by eight, then the remaining bits of the last byte representing a column will be left unused. Image data from the next column will be stored starting in the next byte.

This page is blank intentionally.



## *CHAPTER 4: SYSTEM DEVELOPMENT*

---

Chapter 4 describes how to use and/or implement features of the PK2400 series controller.

## Beginning Development

Before beginning development, check the following items:

- Verify that the PK2400 runs in standalone mode before connecting any I/O devices.
- Verify that the entire host system has good, low-impedance, separate grounds for analog and digital signals. Often the PK2400 is connected between the host PC and another device. Any differences in ground potential from unit to unit can cause serious problems that are hard to diagnose.
- Verify that the host PC's COM port works by connecting a good serial device to the COM port. Remember that on a PC, COM1/COM3 and COM2/COM4 share interrupts. User shells and mouse drivers, in particular, often interfere with proper COM port operation. For example, a mouse running on COM1 can preclude running Dynamic C on COM3.
- Use the supplied Z-World cables. The most common fault of user-made cables is failure to properly assert CTS at the RS-232 port of the PK2400. Without CTSs being asserted, the PK2400's RS-232 port will not transmit. Assert CTS by either connecting the RTS signal of the PC's COM port or looping back the PK2400's RTS.
- Experiment with each peripheral device connected to the PK2400 in order to determine how it appears to the PK2400 when powered up, powered down, and/or when its connecting wiring is open or shorted.

## Operating Modes

The PK2400 has two operating modes that are mutually exclusive: run mode and program mode. Key features of each mode are explained in detail below.

### ***Program Mode***

- The PK2400 controller runs under the control of a host PC running Dynamic C. The PK2400 must be in program mode when attempting to compile a program to the PK2400 or debug a program.
- The PK2400 matches the baud rate of a PC's COM port up to 57.6 kbps. Possible baud rates are 9600 bps and 19.2, 28.8, and 57.6 kbps.

### ***Run Mode***

- The PK2400 controller runs standalone. Upon power-up, the PK2400 checks to see if its onboard memory contains a program. If a program exists, the PK2400 controller executes the program immediately after power-up.
- The PK2400 does not respond to Dynamic C running on a host PC. A program cannot be compiled or debugged when the PK2400 is in run mode.

## Developing with the RS-232 Port

Although the ideal development method is with a SIB2, the RS-232 port is the PK2400's onboard development port.

PK2400 connectors are individual screw terminals that are not polarized or keyed. Carefully observe terminal identification and wire alignment before making a connection and before applying power.

Using the programming cable provided in the development kit, connect the PK2400 to a host PC with the following steps.

1. Disconnect power source to the PK2400.
2. Move the factory installed jumper from pins 7 and 8 to pins 1 and 2 of header J1 on the processor module. Header J1 can be accessed through the SIB2 connector aperture.
3. Connect the 5-wire programming serial cable to the appropriate screw terminals on screw terminal strip J12 as shown in Figure 4-1 and described in Table 4-1.
4. Connect the serial cable's DB9 connector to the host PC's COM port.
5. Reconnect power source. The PK2400 is now in RS-232 programming mode.

Figure 4-1 illustrates the screw terminal strip and Table 4-1 defines the connection between the PK2400 and a host PC.

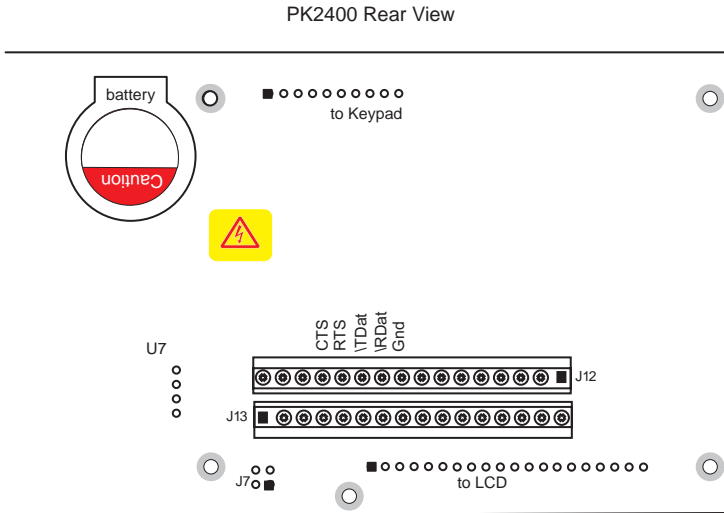


Figure 4-1. RS-232 Screw Terminals

Table 4-1. RS-232 Programming Cable Connections


PC		Signal Direction	Serial Cable Wire Number	PK2400		
RS-232 Signal	DE-9 Pin Number			Serial Cable Signal	J12 Connection	Signal Name
DCD	1	←	1	None	nc	—
DSR	6	←	2	None	nc	—
RD	2	←	3	TXD	11	\TDAT
RTS	7	→	4	CTS	13	CTS
SD	3	→	5	RXD	10	\RDAT
CTS	8	←	6	RTS	12	RTS
DTR	4	→	7	None	nc	—
RI	9	←	8	None	nc	—
SG	5	—	9	None	9	GND

**Nomenclature**

CTS	Clear to Send	RI	Ring Indicator
DCD	Data Carrier Detect	RTS	Request to Send
DSR	Data Set Ready	SD	Send Data
DTR	Data Terminal Ready	SG	Signal Ground
RD	Receive Data	nc	no connection

## Developing with Serial Interface Board

Z-World's SIB2 is an interface adapter useful for PK2400 software development. Contained in an ABS plastic enclosure, the SIB is rugged and reliable. Since the SIB2 permits the PK2400 to communicate with Dynamic C via the Z180's rarely used clocked serial I/O (CSI/O) port, the PK2400's serial port is freed up. The serial port can then be utilized by a user's application during programming and debugging as well as during run time.

 See Appendix E for more information on the SIB.

To connect a SIB to a host PC follow these steps:

1. Disconnect power to the PK2400 if connected.
2. Connect the supplied 6-conductor RJ-12 cable from the PC's serial-port adapter to the SIB.
3. Remove the jumper from pin 7 and 8 of header J1 on the processor module. Save the jumper if the RS-232 port will ever be used for programming via a ribbon cable connector.
4. Connect the SIB's small ribbon cable to the header J1 on the processor module. J1 is visible through the aperture in the back of the PK2400 (see Figure 4-2). Match the arrow on the SIB connector to Pin 1 of J1.

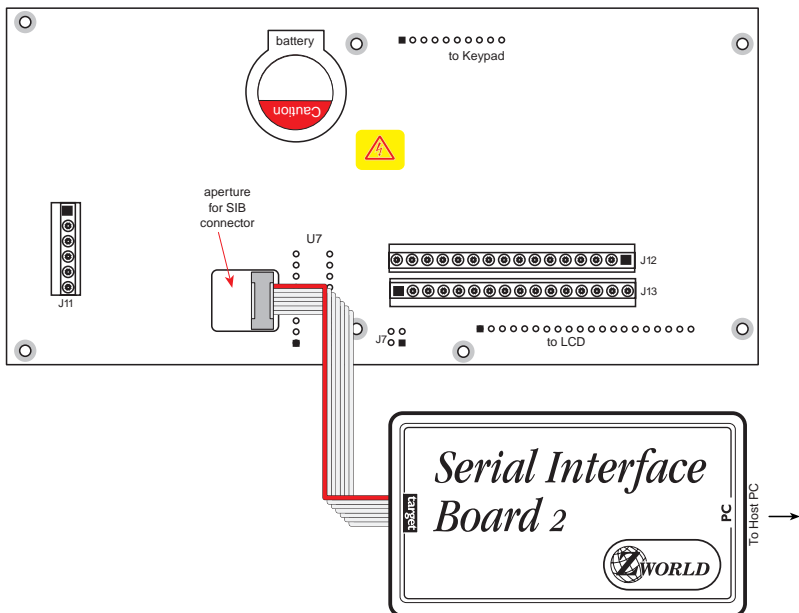


Figure 4-2. SIB Connection to Processor Module

5. Verify that the baud rate of the host PC's is set at 9600, 19200, or 57600.
6. Reconnect the power supply to the PK2400. A minimum 9 V direct current must be connected to the PK2400.

The system is now ready for programming in SIB2 programming mode.

## **Running a Program Standalone**

Use the following steps to run a program standalone (not under Dynamic C control):

1. Download compiled program to flash EPROM.
2. Power down the PK2400.
3. Remove either the SIB2 or the jumper from pins 1 and 2 of J1, depending upon which programming mode was used.
4. Reapply power. The PK2400 begins executing a program automatically.

## ***Returning To Programming Mode***

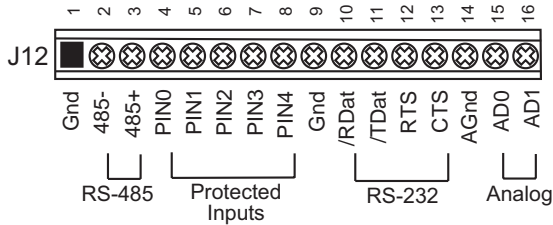
Use the following steps to return to programming mode:

1. Remove power.
2. Reinstall the jumper connecting pins 1 and 2 of J1, or reconnect the SIB2. See Figure 4-2.
3. Reapply power.



# Developing an RS-485 Network

The 2-wire RS-485 serial communication port and Dynamic-C network software allow network development. Screw terminal strip J12, screws 2 and 3 provide a half-duplex RS-485 interface.

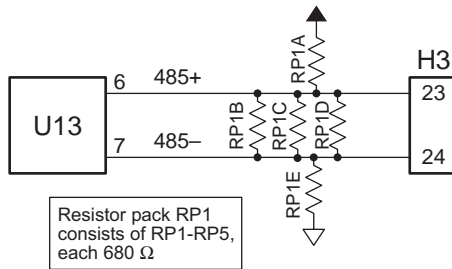


**Figure 4-3. Pin Location of RS-485 Signals**

The PK2400 and/or other controllers can be linked together over several kilometers. When configuring a multidrop network, use single twisted pair wires on all controllers to connect RS-485+ and RS-485- of each controller to RS-485+ and RS-485- on the next controller board.

Any one of Z-World’s controllers can be a master or a slave. Even though a network can have up to 255 slave controllers, only one controller can be the master.

In a multidrop network, termination and bias resistors are required to minimize reflections (echoing) and to keep the network line active during an idle state (see Figure 4-4).



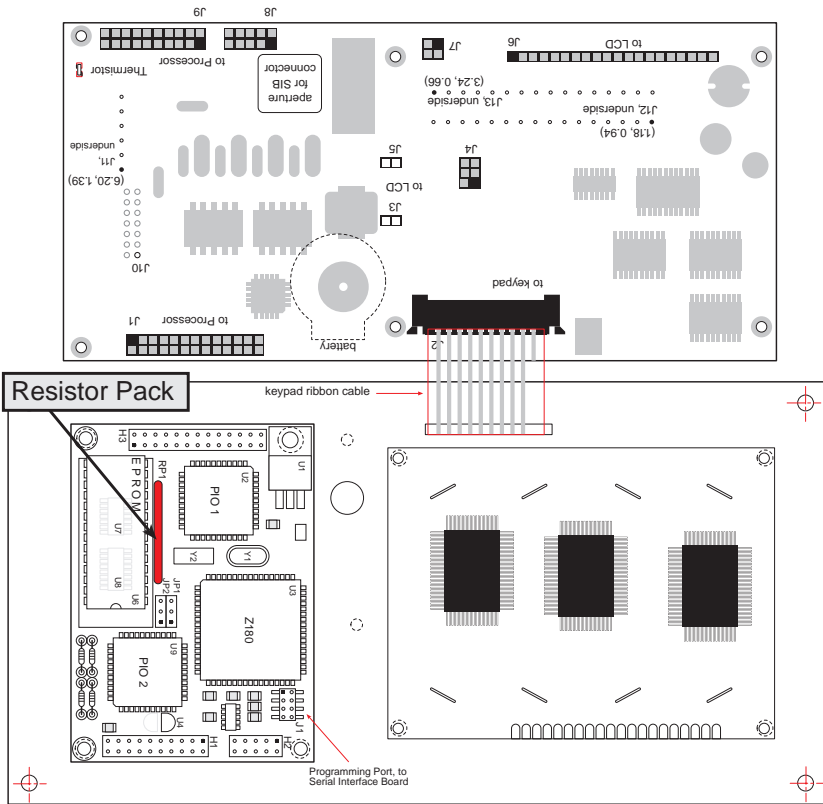
**Figure 4-4. Termination Resistors**

Only the first and last controller in a multidrop network should have termination resistors. Therefore, when networking multiple boards, leave the resistor pack RP1 in the first and last boards, but remove the RP1 from all other boards in the network.

Follow these steps to access and remove the terminator resistors on the processor module:

1. Remove screws from the standoffs that connect the front panel to the back plate.
2. Unfold the rear processor assembly. Do not disconnect the keypad ribbon cable from the rear processor assembly.
3. Remove resistor pack.

Figure 4-5 illustrates an unfolded PK2400 with the resistor pack exposed.



**Figure 4-5. Processor Module Resistor Pack RP1 Location**



## *CHAPTER 5: SOFTWARE REFERENCE*

---

The functions described in this chapter operate the PK2400 input and output interfaces. Most functions are in the **EZIOPK24.LIB** library. Other software functions are in the Dynamic C BIOS of the PK2400's onboard flash EPROM.

## Software Development Options

Applications can be developed on the PK2400 series by using either the RS-232 port or the Serial Interface Board (SIB) port. However, since the SIB port is a dedicated programming port, using it for programming leaves the RS-232 port free for the embedded application.



See Chapter 4 and Appendix E for information on programming with the SIB.

### ***Dynamic C Development Software***

The PK2400 is programmed with Z-World's Dynamic C, an integrated development environment that runs under Windows 3.x, 95, and NT. Dynamic C includes an editor, compiler, downloader, and debugger. Dynamic C is available in the following two versions:

***Standard:*** Maximum 80K of code

This version of Dynamic C is suitable for programs up to 80K with limited access to extended memory. (Data items can not be declared in extended memory.)

***Deluxe:*** Maximum 512K of code, 512K of data

This version supports programs (up to 512K code and 512K data) with full access to extended memory.

### **Dynamic C Manuals**

Z-World offers three Dynamic C manuals for programming reference:

- ***Dynamic C Technical Reference***
- ***Dynamic C Application Frameworks***
- ***Dynamic C Function Reference***

### **Supplied Software**

Software drivers for controlling the PK2400's I/O are provided with Dynamic C 5. In order to use the supplied routines defined in this chapter, use the **EZIOPK24.LIB** Dynamic C library. An application program can utilize the library by including the following line at the top of the program:

```
#use eziopk24.lib
```

Some functions are located in other libraries and these exceptions are noted in the following individual entries:

- **EZIOCMMN.LIB**

I/O functions that are not specific to the PK2400 series controllers. This library is automatically used by **EZIOPK24.LIB**.

- **KP.LIB**

Keypad-related functions. The application should explicitly **#use** this library if the keypad is used.

- **GLCD.LIB**

Graphic LCD related functions. The application should explicitly **#use** this library if the graphic LCD is used.

- **KDI.LIB**

Keypad/display interface-related functions. This library is automatically used by **GLCD.LIB** and **KDI.LIB**. Note that backlight, buzzer, and contrast control are all implemented in this library.

Functions that are not controller specific are not discussed in this manual. Refer to the Dynamic C manuals for information about functions such as serial communication, string, and math functions.

## Memory

### *Flash EPROM Functions*

- **int WriteFlash(unsigned long physical\_addr, char \*buf, int count )**

Writes **count** number of bytes pointed to by **buf** to the flash EPROM absolute data location **physical\_addr**. Allocates data location by declaring the byte arrays as initialized arrays or declares initialize xdata array. If byte array is declared, converts logical memory to physical memory with **phy\_addr(array)**. For initialized xdata, the array name can be passed directly.

PARAMETER1: Flash EPROM absolute data location

PARAMETER2: Pointer to bytes to write

PARAMETER3: Number of bytes to write

RETURN VALUE: 0 if **WriteFlash** is successful

-1 if flash EPROM is not used

-2 if **physical\_address** is inside BIOS area

-3 if **physical\_address** is within Symbol Area or the simulated EEPROM area

-4 if write time-out to the flash EPROM

LIBRARY: **DRIVERS.LIB**

## Nonvolatile Storage

The top 512 bytes of PK2400's flash EPROM are reserved as nonvolatile memory. This area can be used to store important system state information such as A/D conversion constants in order for the PK2400 to recover from power outages.

To ensure compatibility with Z-World's product line, the nonvolatile storage area simulates the EEPROM that certain Z-World controllers use for nonvolatile storage. The high level Dynamic C functions are exactly the same for all versions of all Z-World's controllers. These drivers automatically take care of all low-level differences between physical memory devices.

The logical addresses of the simulated EEPROM are 0 to 511.

Dynamic C BIOS reserves flash EPROM location 0 to store operation mode and location 1 to store the baud code. The remainder of the locations are available for use.

- **int ee\_rd ( int address )**

Reads and returns data from flash EPROM storage location address. The function returns -1 if a non-flash EPROM is used.

- **int ee\_wr ( int address, char data )**

Writes data to simulated-EEPROM storage location address. The function returns -1 if a non-flash EPROM is used.

- **int WriteFlash ( unsigned longaddr, char\* buf, int num )**

Writes **num** bytes from **buf** to flash EPROM, starting at **addr**. The term **addr** is an absolute physical address.

To do this, allocate flash EPROM data in the Dynamic C program by declaring *initialized* variables or arrays, or initialized **xdata**. For **xdata**, the data name must pass directly to this function:

```
xdata my_data { 0, 0xFF, 0x08 };  
...  
WriteFlash( my_data, my_buffer, my_count );
```

For normal data, the physical address of the data must pass to this function:

```
char xxx[] = { 0, 0xFF, 0x08 };  
...  
WriteFlash( phy_adr( xxx ), my_buffer,  
my_count );
```

The function returns

- 0 if the operation was successful
- 1 if no flash EPROM is present
- 2 if a physical address is within the BIOS area (low 8K)
- 3 if a physical address is within the symbol table
- 4 if the write times out

In order for this function to work, some form of initialization has to be included when declaring the data. If the data is not declared, it will be placed in RAM instead of ROM and this function will not work.

Writing to “read-only” memory may seem contradictory, but flash EPROM, despite its name, is not really read-only memory. Writing to flash EPROM, in essence, treats the flash memory as read/write nonvolatile memory.



Flash EPROM manufacturers rate their devices conservatively at 10,000 writes. In tests, a flash EPROM can last for 100,000 writes. When this limit is reached, the system will fail and the integrated circuit must be replaced.

## Digital Input and Output

### *Digital Input/Output Functions*

The digital I/O functions provide easy software access to the PK2400’s protected inputs, high-current outputs, and relay outputs. The state of the latching relays is determined by the number of on-off cycles for the relay. Each time the relay is deenergized or reenergized, it changes states. Latching relays maintain their state even when power is removed from the PK2400. When using latching relays, make sure that any equipment connected to the latching relays will continue to operate safely or shut down in the event of a controller malfunction or power loss.



When using nonlatching relays, the state is determined by the state of the corresponding output. If the output is on, the relay is in the energized state. If the output is off, the relay is in the deenergized state.

- **void eioBrdInit( int flags )**

Initializes the PK2400 board. The application should call this function before performing *any* I/O operations, including digital input, digital output, analog input, graphic LCD, or keypad operations.

PARAMETER: **flags** should be zero.

RETURN VALUE: N/A.

- `int eioBrdDI( unsigned chanNum )`

Reads the state of an input channel.

PARAMETER: `chanNum` must be a number ranging from 0 (for PIN-0) through 8 (for PIN-8) inclusively.

RETURN VALUE: returns 0 if and only if the input channel reads low; returns 1 if and only if the input channel reads high; sets the flag `EIO_NODEV` in `eioErrorCode` and returns -1 if and only if the channel does not exist (i.e., if `chanNum` is greater than 8).

LIBRARY: `EZIOPK24.LIB`

Table 5-1 lists software input channel assignments.

**Table 5-1. Software Input Channel Assignments**

Label	Channel Assignment
PIN-0	0
PIN-1	1
PIN-2	2
PIN-3	3
PIN-4	4
PIN-5	5
PIN-6	6
PIN-7	7
PIN-8	8

If digital input PIN-0 (input channel 0) is grounded, the digital output OUT-0 (output channel 0) is disabled “OFF.” Otherwise, the digital output is enabled.

**Program 5-1. Reflect Input State to Output**

```
#use ezio.lib           // general I/O definitions
#use eziopk24.lib      // pk2400 specific defns
#define IN_CHAN 0      // define input channel
#define OUT_CHAN 0     // define output channel
main() {
    eioBrdInit(0);
    while (1) {        // do this indefinitely
        eioBrdDO(OUT_CHAN,eioBrdDI(IN_CHAN));
        hitwd();      // hit watchdog
    }
}
```



- **int eioBrdDO( unsigned chanNum, char state )**

Changes the state of an output channel.

PARAMETER 1: **chanNum** must range from 0 (for OUT-0) through 7 (for OUT-7) inclusively. OUT-4 and OUT-5 will also control nonlatching relays 1 and 2. OUT-6 and OUT-7 will also control latching relays 1 and 2.

PARAMETER 2: **state** is 0 if and only if the corresponding output is to be disabled “OFF” or 1 if and only if the corresponding output is to be enabled “ON.”

RETURN VALUE: returns 0 if and only if **chanNum** is within range; sets the flag **EIO\_NODEV** in **eioErrorCode** and returns -1 if and only if **chanNum** is out of range.

LIBRARY: **EZIOPK24.LIB**

Table 5-2 lists software output channel assignments.

**Table 5-2. Software Output Channel Assignments**

Label	Channel Assignment
Out-0	0
Out-1	1
Out-2	2
Out-3	3
Out-4	4
Out-5	5
Out-6	6
Out-7	7



Refer to the Dynamic C subdirectory **SAMPLES\PK24XX** for additional sample programs.

## Advanced Input/Output Programming

Most I/O devices on the PK2400 series controllers, including digital input, digital output, analog input, 485 driver, keypad, and graphic LCD are controlled through the two Zilog PIO chips (Z84C20) on the PK2400's processor module. Z-World recommends that the application programmer use the supplied functions documented in this chapter for a simple interface to the hardware.

However, if necessary, the application programmer may write custom low-level driver routines for size, speed, or both. In order to write the low-level routines that interact with the controller hardware, the following items are necessary:

- PK2400 Schematics
- BL1500 Schematics (processor module of the PK2400)
- Specifications of the control chip(s). Probably including the specifications of the PIO (Z84C20) chip from Zilog

The low-level driver programmer also needs to know how the PIO chips are mapped in the Z180 I/O space. The following list describes how the PIO chips are interfaced to the Z180 processor.

### ***U2 (PIO No. 1)***

- Port A uses vector 0x12 in the interrupt table
- Port B uses vector 0x14 in the interrupt table
- Port A data register is at 0xc0 in the I/O space
- Port A control register is at 0xc2 in the I/O space
- Port B data register is at 0xc1 in the I/O space
- Port B control register is at 0xc3 in the I/O space

### ***U9 (PIO No. 2)***

- Port A uses vector 0x22 in the interrupt table
- Port B uses vector 0x24 in the interrupt table
- Port A data register is at 0x80 in the I/O space
- Port A control register is at 0x82 in the I/O space
- Port B data register is at 0x81 in the I/O space
- Port B control register is at 0x83 in the I/O space

## Serial Communication Functions

There are several libraries that can be used for serial communication. Serial communication functions, constants, and definitions are described in the *Dynamic C Function Reference Manual* and *Dynamic C Application Framework Manual*. **AASC.LIB** is the recommended serial communication library.

### **RS-485 Functions**

- **void on\_485( void )**  
Turns the RS-485 transmitter on.  
PARAMETER: none  
RETURN VALUE: none  
LIBRARY: **DRIVERS.LIB**
  
- **void off\_485(void)**  
Turns the RS-485 transmitter off.  
PARAMETER: none  
RETURN VALUE: none  
LIBRARY: **DRIVERS.LIB**

## Analog-to-Digital Converter Drivers

- **void eioBrdInit( int flags )**

Initializes the PK2400 board. The application program should call this function before performing *any* I/O operations, including digital input, digital output, analog input, graphic LCD, or keypad operations.

PARAMETER: **flags** should be zero

RETURN VALUE: N/A

LIBRARY: **EZIOCMN.LIB**

- **int \_eioBrdAI(unsigned int eioAddr )**

Reads an input and performs analog-to-digital conversion. Does not check validity of the parameter and does not convert according to calibration coefficients (see **eioBrdAI**).

PARAMETER: **eioAddr** specifies an input number of 0, up to 15. **eioAddr** values 0 to 1 represent analog inputs AD0 and AD1 and will cause the function to return the 12-bit A/D value read on an input. If **eioAddr** is between 11 and 15, the function returns the A/D values of internal channels of the ADC chip.

RETURN VALUE: A 12-bit unsigned

LIBRARY: **EZIOPK24.LIB**

- **float eioBrdAI(unsigned int eioAddr )**

Reads an input and performs analog-to-digital conversion. Sets **eioErrorCode** if **eioAddr** is out of range.

PARAMETER: **eioAddr** specifies an input number of 0 to 1 or 16 to 17 to be read. **eioAddr** values 0 and 1 represent analog inputs AD0 and AD1 and will cause the function to return the voltage read on an input. **eioAddr** values 16 and 17 also represent analog inputs AD0 and AD1, but cause the function to return a 12-bit raw data value for the analog input.

RETURN VALUE: For **eioAddr** values 0 and 1, if the read is successful, the function will return the voltage read. For **eioAddr** values 16 and 17, if the read is successful, the function returns the 12-bit raw data value read from the A/D converter.

LIBRARY: **EZIOCMN.LIB**

- **int eioBrdACalib( int eioAddr, unsigned int d1, unsigned int d2, float v1, float v2 )**

Calculates the calibration constants for an analog input channel using two known voltages and two corresponding raw data readings. Stores the calibration constants in EEPROM.

PARAMETER 1: **eioAddr** is the analog input channel

PARAMETER 2: **d1** is the raw data corresponding to v1

PARAMETER 3: **d2** is the raw data corresponding to v2

PARAMETER 4: **v1** is the known voltage used to obtain d1

PARAMETER 5: **v2** is the known voltage used to obtain d2

RETURN VALUE: 0 if successful, -1 if **eioAddr** is out of range.

LIBRARY: **EZIOCMN.LIB**



Since the PK2400 is calibrated at the factory, use the above function only if it is necessary to recalibrate the PK2400.

## Using the Liquid Crystal Display and Keypad

The following functions provide routines that write to the LCD, and read from the keypad. Include the following directives:

```
#use wintek.lib
#use kp.lib
#use glcd.lib
```

These directives provide information to the compiler about the graphic LCD and keypad.

- **void eioBrdInit( int flags )**

Initializes the PK2400 board. The application program should call this function before performing *any* I/O operations, including digital input, digital output, analog input, graphic LCD, or keypad operations.

PARAMETER: **flags** should be zero.

RETURN VALUE: N/A

- **void glSetBrushType( int type )**

Sets the brush type for all following graphics operations in this library. Controls how pixels are drawn on the screen with respect to existing pixels.

PARAMETER 1: This is the type of the brush. Possible values are **GL\_SET** for forcing pixels on, **GL\_CLEAR** for forcing pixels off, **GL\_XOR** for toggling the existing pixels and **GL\_BLOCK** to overwrite the entire memory location corresponding to the pixel.

RETURN VALUE: None

- **int glInit()**

Initializes the LCD module (software and hardware). This function should be called after calling **eioBrdInit**. Call this function before performing any LCD operations.

RETURN VALUE: Returns the status of the LCD. If the initialization was successful, this function returns 0. Otherwise, the returned value indicates the LCD status.

- **int glBlankScreen()**

Blanks the screen of the LCD.

RETURN VALUE: The returned value indicates the status of the LCD after the operation.

- **int glPlotDot( int x, int y )**

Plots one pixel on the screen at coordinate (x,y).

PARAMETER1: the x-coordinate of the pixel to be drawn

PARAMETER2: the y-coordinate of the pixel to be drawn

RETURN VALUE: Status of the LCD after the operation

- **void glPlotLine( int x1, int y1, int x2, int y2 )**

Plots a line on the LCD.

PARAMETER1: x-coordinate of first end-point

PARAMETER2: y-coordinate of first end-point

PARAMETER3: x-coordinate of second end-point

PARAMETER4: y-coordinate of second end-point

RETURN VALUE: None

- **void glPutBitmap( int x, int y, int bmWidth, int bmHeight, char \*bm )**

Displays a bitmap stored in root memory on the LCD. For bitmaps defined in **xmem** memory, use **glXPutBitmap**.

PARAMETER1: x-coordinate of the bitmap (left edge)

PARAMETER2: y-coordinate of the bitmap (top edge)

PARAMETER3: width of the bitmap

PARAMETER4: height of the bitmap

PARAMETER5: pointer to the bitmap

RETURN VALUE: None

- **void glXPutBitmap( int x, int y, int bmWidth, int bmHeight, unsigned long bmPtr )**

Displays a bitmap stored in **xmem** on the LCD. For bitmaps stored in root memory, use **glPutBitmap**.

PARAMETER1: x-coordinate of the bitmap (left edge)

PARAMETER2: y-coordinate of the bitmap (top edge)

PARAMETER3: width of the bitmap

PARAMETER4: height of the bitmap

PARAMETER5: pointer to the bitmap

RETURN VALUE: None

- **void glGetBitmap( int x, int y, int bmWidth, int bmHeight, char \*bm )**

Gets a bitmap from the LCD.

PARAMETER1: x-coordinate of the bitmap (left edge)

PARAMETER2: y-coordinate of the bitmap (top edge)

PARAMETER3: width of the bitmap

PARAMETER4: height of the bitmap

PARAMETER5: pointer to the bitmap

RETURN VALUE: None

- **void glFontInit( struct \_fontInfo \*pInfo, char pixWidth, char pixHeight, unsigned startChar, unsigned endChar, char bitmapBuffer )**

Initializes a font descriptor with the bitmap defined in the root memory. For fonts with bitmaps defined in **xmem**, use **glXFontInit**.

PARAMETER1: pointer to the font descriptor to be initialized

PARAMETER2: width of each font item (must be uniform for all items)

PARAMETER3: height of each font item (must be uniform for all items)

PARAMETER4: offset to the first usable item (useful for fonts for ASCII or other fonts with an offset)

PARAMETER5: index of the last usable font item

PARAMETER6: pointer to a linear array of font bitmap

RETURN VALUE: None

- **void glXFontInit( struct \_fontInfo \*pInfo, char pixWidth, char pixHeight, unsigned startChar, unsigned endChar, ulong xmemBuffer )**

Initializes a font descriptor that has the bitmap defined in **xmem**. For bitmaps defined in root memory, use **glFontInit**.

PARAMETER1: pointer to the font descriptor to be initialized

PARAMETER2: width of each font item (must be uniform for all items)

PARAMETER3: height of each font item (must be uniform for all items)

PARAMETER4: offset to the first usable item (useful for fonts for ASCII or other fonts with an offset)

PARAMETER5: index of the last usable font item

PARAMETER6: pointer to a linear array of font bitmap

RETURN VALUE: None

- **void glPutFont( int x, int y, struct fontInfo \*pInfo, unsigned code )**

Puts an entry from the font table to the LCD.

PARAMETER1: x-coordinate of the entry (left edge)

PARAMETER2: y-coordinate of the entry (top edge)

PARAMETER3: pointer to the font descriptor that describes the font table to be indexed

PARAMETER4: code (offset) in the font table that indexes the bitmap to display

RETURN VALUE: None

- **void glVPrintf( int x, int y, struct fontInfo \*pInfo, char \*fmt, void \*firstArg )**

Prints a formatted string on the LCD screen, similar to **vprintf**.

PARAMETER1: x-coordinate of the text (left edge)

PARAMETER2: y-coordinate of the text (top edge)

PARAMETER3: pointer to font descriptor that describes the font used for printing the text

PARAMETER4: pointer to the string that describes the format

PARAMETER5: pointer to the first argument

RETURN VALUE: None



- **void glPrintf( int x, int y, struct \_fontInfo \*pInfo, char \*fmt, ... )**  
 Prints a formatted string (much like **printf**) on the LCD screen  
 PARAMETER1: x-coordinate of the text (left edge)  
 PARAMETER2: y-coordinate of the text (top edge)  
 PARAMETER3: pointer to the font descriptor used for printing on the LCD screen  
 PARAMETER4: pointer to the format string  
 RETURN VALUE: None
- **void glPlotCircle( int xc, int yc, int rad )**  
 Draws a circle on the LCD.  
 PARAMETER1: x-coordinate of the center  
 PARAMETER2: y-coordinate of the center  
 PARAMETER3: radius of the circle  
 RETURN VALUE: None
- **int wtDisplaySw( int onOff )**  
 Switches the display on and off.  
 PARAMETER1: If this parameter is 1, the display is turned on. If this parameter is 0, the display is turned off.  
 RETURN VALUE: Status of the LCD after the operation
- **void kdiELSw( int value )**  
 Switches the EL backlight of the LCD.  
 PARAMETER1: 1 to turn the backlight on, 0 to turn the backlight off  
 RETURN VALUE: None
- **void kdiBuzSw( int value )**  
 Switches the buzzer on the keypad display interface board.  
 PARAMETER1: 1 to turn the buzzer on, 0 to turn the buzzer off  
 RETURN VALUE: None
- **void kdiSetContrast( unsigned content )**  
 Sets the contrast control to “content.”  
 PARAMETER1: Specifies the contrast (the higher the value, the higher the contrast)  
 RETURN VALUE: None

- **void kpInit( int (\*changeFn) () )**

Initializes the **kp** module. This function should be called before other functions of this module are called. If the default keypad scanning routine will be used, use **kpDefInit** instead of this function.

PARAMETER1: This is a pointer to a function which will be called when the driver detects a change (when **kpScanState** is called). Two arguments are passed to the call-back function. The first argument is a pointer to an array that indicates the current state of the keypad. The second is a pointer to an array that indicates what keypad positions are changed and detected by **kpScanState**. The byte offset in the array represents the line pulled high (row number), and the bits in a byte represents the positions (column number) read back.

RETURN VALUE: None.

- **int kpScanState ()**

Scans the keypad and detect any changes to the keypad status. It returns non-zero if there is any change. If **kpInit** is called with a non-NULL function pointer, that function will be called with the state of the keypad. This function should be called periodically to scan for keypad activities.

RETURN VALUE: 0 if there is no change to the keypad, non-zero if there is any change to the keypad.

- **int kpDefStChgFn( char \*curState, char \*changed )**

This is the default state change function for the default get key function **kpDefGetKey**. This function is called back by **kpScanState** when there is a change in the keypad state. If the current key is not read by **kpDefGetKey**, the new key pressed will not be registered.

PARAMETER1: Points to an array that reflects the current state of the keypad (bitmapped, 1 indicates key is not currently pressed).

PARAMETER2: Points to an array that reflects the CHANGE of keypad state from the previous scan. (bitmapped, 1 indicates there was a change).

RETURN VALUE: -1 if no key is pressed. Otherwise it returns the normalized key number. The normalized key number is

**8\*row+col+edge\*256**. **edge** is 1 if the key is released, and 0 if the key is pressed.

- **int kpDefGetKey()**

This is the default get key function. It returns the key previously pressed (i.e., from the one-keypress buffer). The key pressed is actually interpreted by **kpDefStChgFn**, which is called back by **kpScanState**. **kpDefInit** should be used to initialize the module.

RETURN VALUE: -1 if no key is pressed. Otherwise, it returns the normalized key number. The normalized key number is  $8 * \text{row} + \text{col} + \text{edge} * 256$ . **edge** is 1 if the key is released, and 0 if the key is pressed.

- **void kpDefInit()**

Initializes the module to use the default state change function to interpret key presses when **kpScanState** is called. Use **kpDefGetKey** to get the code of the last key pressed.

RETURN VALUE: N/A

# Real-Time Clock Integrated Circuit

The RTC IC has the following features:

- A clock/calendar that accounts for leap years and the varying number of days in a month
- A 31-byte scratchpad RAM

An application program can write to both the clock/calendar and the RAM in “burst mode.” In this mode, the software specifies a start location and then writes multiple, consecutive characters to the RTC integrated circuit. The integrated circuit automatically increments the internal address for the next write. This approach is more efficient than setting the address for each byte in the non-burst mode.

## Global Time and Date Structure

Dynamic C automatically creates the global structure shown in Program 5-2 to hold the time and date.

**Program 5-2. Global Time and Date Structure**

```
struct tm{
    char tm_sec;    // 0-59
    char tm_min;    // 0-59
    char tm_hour;   // 0-23
    char tm_mday;   // 1-31
    char tm_mon;    // 1-12
    char tm_year;   // 0-150 (1900-2050)
    char tm_wday;   // 0-6 where 0 means Sunday
};
```

Since Dynamic C automatically creates this structure, it should not be declared in an application program.

- **int tm\_rd ( struct tm \*t )**

Stores the real-time clock (RTC) IC’s current contents into the structure \*t.

If the RTC is halted, this function stores the value for midnight, 1 January 1980 in \*t, and returns -1. Otherwise, the function stores the current time in \*t and returns 0.

- **int tm\_wr ( struct tm \*t )**

Writes the values in the structure \*t to the RTC.

The function returns 0 if successful, or -1 if it is unable to start the RTC.

- **int WriteRam1302 ( int ram\_loc, byte data )**  
Writes data to any of the 31 (0–30) RAM locations of the DS1302.  
The function returns 1 if successful and –1 if **ram\_loc** is out of range.
- **int ReadRam1302 ( int ram\_loc )**  
Reads data from any of the 31 (0–30) RAM locations of the DS1302.  
The result of the function is the data value or –1 if **ram\_loc** is out of range.
- **void WriteBurst1302 ( void\* pdata, int count )**  
Writes count bytes, in burst mode, to the DS1302, starting at RAM location 0.  
The parameter **pdata** points to the data.
- **void ReadBurst1302 ( void\* pdata, int count )**  
Reads count bytes, in burst mode, from the DS1302, starting at RAM location 0.  
The parameter **pdata** points to a storage area for the data.
- **void Write1302 ( int reg, byte data )**  
Writes data to a specified register of the DS1302.
- **int Read1302 (int reg)**  
Reads data from a specified register of the DS1302.  
The result of the function is the data value.

## Sample Programs

Table 5-3 lists and describes sample programs in the subdirectory **SAMPLES\PK24XX** in the Dynamic C directory.

*Table 5-3. Sample Programs in SAMPLES\PK2400XX*

Program	Description
<b>DEMO.C</b>	Dynamic C Deluxe version only. A program to demonstrate graphic LCD and keypad features.
<b>KPDEFLT.C</b>	A program to demonstrate how to use the default keypad scanning routine.
<b>SHOWIN.C</b>	Displays the current state and history of all digital input lines on the graphic LCD as waveforms.
<b>LCHRELAY.C</b>	Demonstrates how to use the latching relays.
<b>FANCY1.C</b>	Dynamic C Deluxe version only. Uses the LCD and keypad to reflect the states of the digital input lines and to toggle the digital output lines.

## Additional Software

For **Watchdog** information, refer to descriptions of the function **hitwd** in the Dynamic C manuals.

For reading and writing **Simulated EEPROM** data, refer to descriptions of the functions **ee\_rd** and **ee\_wr** in the Dynamic C manuals.

For **Power Fail Flag** information, refer to the descriptions of the function **\_sysIsPwrFail** and **sysIsPwrFail** in the Dynamic C manuals.

For **Resetting the Board** information, refer to descriptions of the functions **sysForceSupRst**, **sysIsSuperReset**, **\_sysIsSuperReset**, **sysForceReset**, **\_sysIsWDTO**, and **sysIsWDTO** in the Dynamic C manuals.



## *APPENDIX A: TROUBLESHOOTING*

---

Appendix A provides procedures for troubleshooting system hardware and software.

## Out of the Box

Check the items mentioned in this section before starting development.

- Verify that the PK2400 runs in standalone mode before connecting any expansion boards or I/O devices.
- Verify that the entire host system has good, low-impedance, separate grounds for analog and digital signals. Often the PK2400 is connected between the host PC and another device. Any differences in ground potential from unit to unit can cause serious problems that are hard to diagnose.
- Do not connect analog ground to digital ground anywhere.
- Double-check the connecting ribbon cables to ensure that all wires go to the correct screw terminals on the PK2400.
- Verify that the host PC's COM port works by connecting a known to be good serial device to the COM port. Remember that COM1/COM3 and COM2/COM4 share interrupts on a PC. User shells and mouse drivers, in particular, often interfere with proper COM port operation. For example, a mouse running on COM1 can preclude running Dynamic C on COM3.
- Use the supplied Z-World power supply. If another power supply must be used, verify that it has enough capacity and filtering to support the PK2400.
- Use the Z-World cables supplied. The most common fault of user-made cables is failure to properly assert CTS at the RS-232 port of the PK2400. Without CTSs being asserted, the PK2400's RS-232 port will not transmit. Assert CTS by either connecting the RTS signal of the PC's COM port or looping back the PK2400's RTS.
- Experiment with each peripheral device connected to the PK2400 to determine how it appears to the PK2400 when powered up, powered down, and/or when its connecting wiring is open or shorted.



## Dynamic C Will Not Start

In most situations, when Dynamic C will not start an error message announcing a communication failure will be displayed. Following is a list of situations causing an error message and possible resolutions.

- *Wrong Baud Rate* — In rare cases, the baud rate has to be changed when using the Serial Interface Board for development.
- *Wrong Communication Mode* — Both sides must be talking RS-232.
- *Wrong COM Port* — A PC generally has two serial ports: COM1 and COM2. Specify the port being used in the Dynamic C “Target Setup” menu. Use trial and error, if necessary.
- *Wrong Operating Mode* — Communication with Dynamic C will be lost if the PK2400’s jumper is set for standalone operation. Reconfigure the board for programming mode.
- *Wrong Memory Size* — Jumpers JP1 and JP2 of the PK2400 specify the EPROM size.

If all else fails, connect the serial cable to the PK2400 after power up. If the PC’s RS-232 port supplies a large current (most commonly on portable and industrial PCs), some RS-232 level converter ICs go into a nondestructive latch-up. Connect the RS-232 cable after power up to eliminate this problem.

## Dynamic C Loses Serial Link

If the program disables interrupts for a period greater than 50 ms, Dynamic C will lose its serial link with your program. Make sure that interrupts are not disabled for a period greater than 50 ms.

## PK2400 Repeatedly Resets

The PK2400 resets every 1.0 seconds if the watchdog timer is not “hit.” If a program does not “hit” the watchdog timer, then the program will have trouble running in standalone mode. To “hit” the watchdog, make a call to the Dynamic C library function `hitwd`.

## Common Programming Errors

- Values for constants or variables out of range. Table A-1 lists and defines ranges for variables and constants.

**Table A-1. Constant and Variable Ranges**

Type	Range
<b>int</b>	-32,768 ( $-2^{15}$ ) to +32,767 ( $2^{15}-1$ )
<b>long int</b>	-2,147,483,648 ( $-2^{31}$ ) to +2147483647 ( $2^{31}-1$ )
<b>float</b>	$1.18 \times 10^{-38}$ to $3.40 \times 10^{38}$
<b>char</b>	0 to 255

- Mismatched “types.” For example, the literal constant **3293** is of type **int** (16-bit integer). However, the literal constant **3293.0** is of type **float**. Although Dynamic C can handle some type mismatches, avoiding them is the best practice.
- Counting up from, or down to, one instead of zero. In software, ordinal series often begin or terminate with zero, not one.
- Confusing a function’s definition with an instance of its use in a listing.
- Not ending statements with semicolons.
- Not inserting commas as required in functions’ parameter lists.
- Leaving out an ASCII space character between characters forming a different legal, but unwanted operator.
- Confusing similar looking operators such as **&&** with **&**, **==** with **=**, or **//** with **/**.



## *APPENDIX B: **SPECIFICATIONS***

---

Appendix B presents the dimensions and specifications for the PK2400 controller and its components.

## Specifications

Table B-1 lists the electrical, mechanical, and environmental specifications for the PK2400.

**Table B-1. General Specifications**

Parameter	Specification
Product Size	7.75" × 4.0" × 1.125" (197 mm × 102 mm × 28 mm)
Enclosure Size	7.85" × 4.1" × 1.5" (200 mm × 104 mm × 38 mm)
Bezel Size	8.5" × 4.75" × 0.125" (216 mm × 121 mm × 3.2 mm)
Operating Temperature	0°C to +50°C
Humidity	5% to 95%, noncondensing
Input Voltage, Current	9 V to 12 V DC, 150 mA plus 30 mA per active relay
Configurable I/O	Digital inputs can be pulled up or down
Digital Inputs	9 protected, -20 V to +24 V
Digital Outputs	6 channels, sourcing (2985). Sinking drivers (2803) optional.
Relays	2 nonlatching relays rated 2 A at 30 V DC or 0.5 A at 120 V AC
Analog Inputs	2, conditioned, 12-bit (two other A/D channels internally support the temperature sensor)
Analog outputs	None
Resistance Measurement Input	None
Processor	Z180
Clock	9.216 MHz
SRAM	128K, socketed; 256K optional
EPROM	None
Flash	128K, socketed
EEPROM	None
Counters	Software-implementable

continued...

**Table B-1. General Specifications (concluded)**

Parameter	Specification
Serial Ports	One RS-232 (with RTS/CTS handshake) One RS-485 (two-wire)
Serial Rate	Up to 57,600 bps
Watchdog/Supervisor	Yes
Time/Date Clock	Yes
Backup Battery	Yes, 3 V lithium coin-type, 560 mA·h
Keypad and LCD	Yes
Expansion Port	None



For CE compliance, the maximum relay switching voltage is less than 50 V AC or 75 V DC.

## Hardware Dimensions

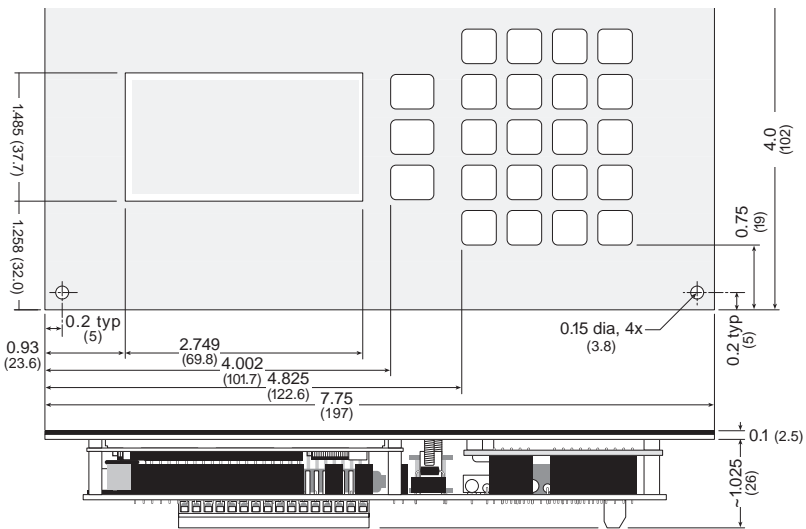
The PK2400 measures  $7.75^2\text{W} \times 4.0^2\text{L} \times 1.125^2\text{H}$ .

The keys in the  $4 \times 5$  matrix are  $0.4''$  square. The simulated bezels are  $0.5^2$  square. The keys are spaced  $0.525''$  apart in each direction. The radius on the key aperture is  $0.0625''$  ( $1/16''$ ).

The keys in the  $1 \times 3$  array are  $0.4'' \times 0.5''$ . The simulated bezels are  $0.5'' \times 0.6''$ . The keys are spaced  $0.525''$  apart vertically. The radius on the key aperture is  $0.0625''$  ( $1/16''$ ).

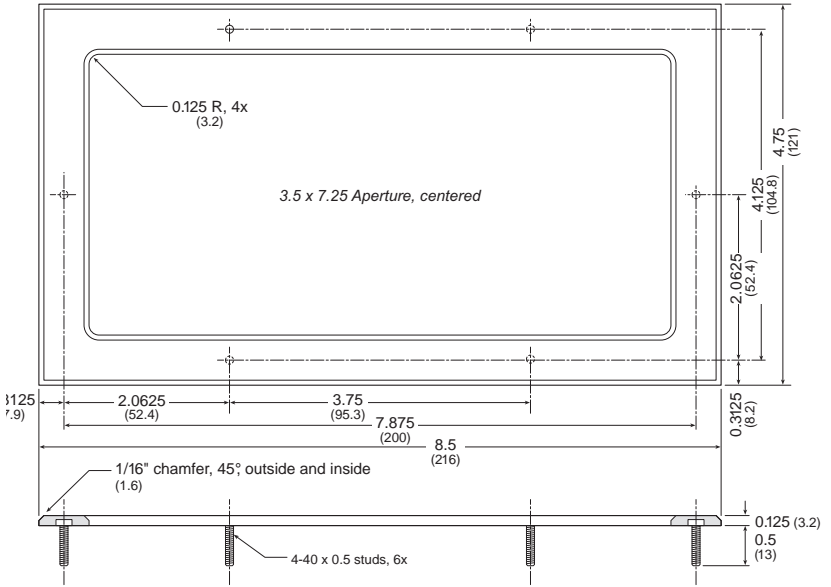
The mounting holes are  $0.15''$  in diameter.

Figure B-1 illustrates the PK2400's dimensions and mounting hole locations.



**Figure B-1. PK2400 Dimensions and Mounting Hole Locations**

Figure B-2 illustrates the optional mounting bezel dimensions.

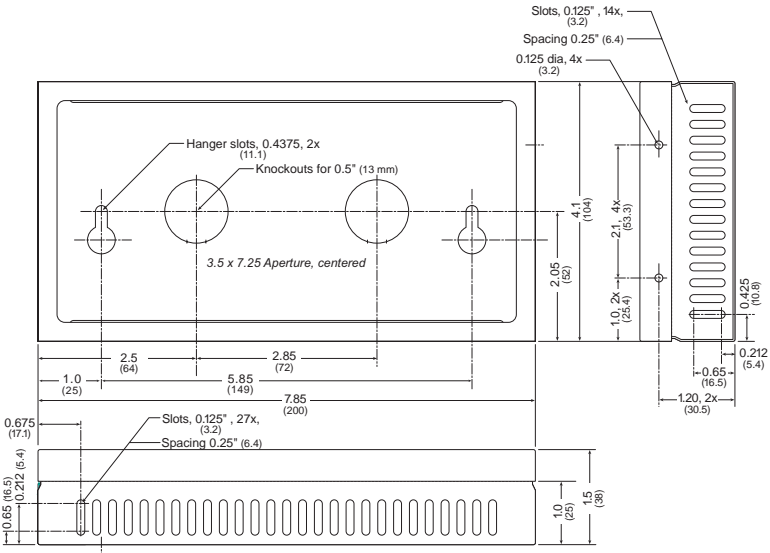


**Figure B-2. Optional Mounting Bezel Dimensions**



To ensure a proper fit, bezel mounting holes must be cut to exactly 8"W x 4.5"H.

Figure B-3 illustrates the optional wall-mount enclosure dimensions.



**Figure B-3. Optional Wall-Mount Enclosure Dimensions**

The keys in the 4 × 5 matrix are 0.4" square. Keys are spaced 0.525" apart in each direction. The radius on the key aperture is 0.0625" (1/16").

The keys in the 1 × 3 array are 0.4" × 0.5". Keys are spaced 0.525" apart vertically. The radius on the key aperture is 0.0625" (1/16").



## Suggested Operating Temperature

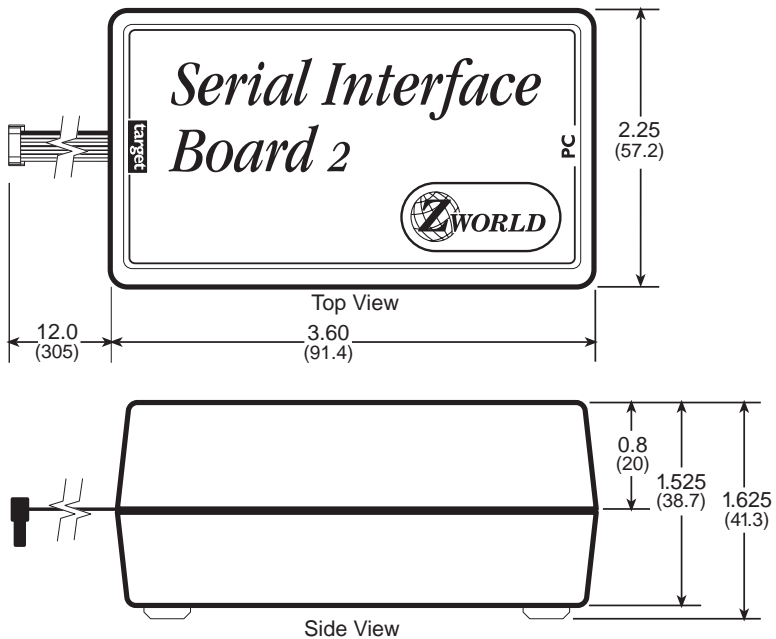
The PK2400 operates at ambient temperatures from 0°C to 50°C and may be stored at temperatures from -55°C to 85°C.

## Power

An onboard 5 V regulator accepts unregulated 9 to 12 V DC. The processor module draws 80 mA. A separate switching regulator is provided on the PK2400. Up to 750 mA is available to the user through screw terminal strip J13 at screw terminal 13. On the PK2410, up to 100 mA is available to the user at this terminal.

## Serial Interface Board

Figure B-4 illustrates the connections and shows the dimensions of a SIB.



**Figure B-4. SIB2 Connections and Dimensions**



Appendix E, “Serial Interface Board 2,” provides further information on the SIB2.

## Jumper and Header Specifications

Figure B-5 shows locations of the headers and jumpers on the rear of the PK2400.

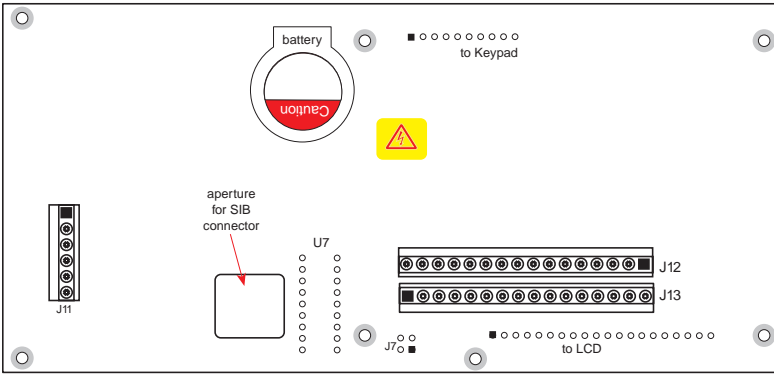


Figure B-5. PK2400 Screw Terminal Strip Locations



Due to the proximity of solder pins, use caution when inserting wires into J12 and/or J13 to prevent a short circuit on the board.

## J12, RS-232, and RS-485 Ports

Screw terminal strip J12, screws 10 through 13 carry the RS-232 signals. Screw terminal strip J12, screws 2 and 3 carry the RS-485 signals. Figure B-6 shows the locations of the communication screw terminals.

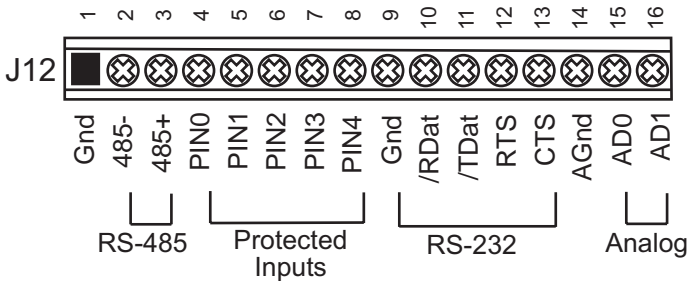


Figure B-6. Screw Terminal Strip J12 Signals

## **Processor Module Jumpers**

Two jumpers, JP1 and JP2, specify flash EPROM size.

## **Processor Module Gain and Bias Resistors**

Table B-2 lists the processor module's bias and gain resistors number and function.

**Table B-2. Processor Module  
Bias and Gain Resistors**

<b>Resistor</b>	<b>Function</b>
R11	Gain resistor, Channel 0
R12	Bias resistor, Channel 0
R13	Gain resistor, Channel 1
R14	Bias resistor, Channel 1

## **RS-232 Programming**

When developing through the RS-232 port instead of the Serial Interface Board, place a jumper across pins 1 and 2 of header J1 on the processor module (normally the SIB port) to make the processor module come up in RS-232 programming mode after power up. If neither a jumper nor a SIB is present, the processor module comes up in run mode after power up and executes an application program from EPROM. In run mode, the processor module will not respond to Dynamic C.

This page is blank intentionally.



## *APPENDIX C: POWER MANAGEMENT*

---

Appendix C provides detailed information on power systems and sources.

## Direct Current Input

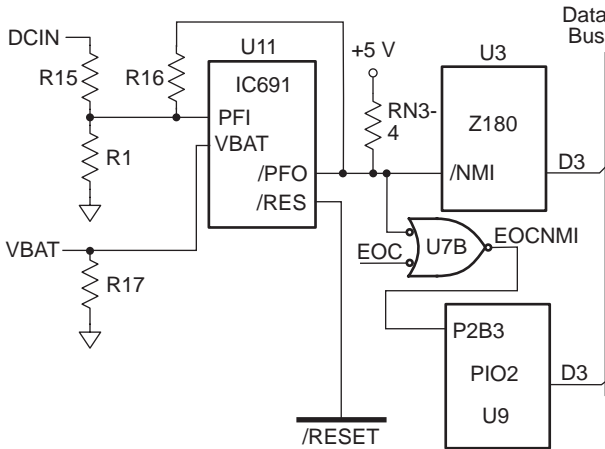
A direct current power supply of approximately 12 volts is required to operate the PK2400 or PK2410. Z-World's part number 101-0107 is a suitable source of power.

## Power Regulator

The PK2400 has a 5 V switching regulator, which powers the interface board and has approximately 750 mA of excess capacity available to the user at screw terminal strip J13, screw terminal 13. The PK2410 uses the linear regulator on the processor module to supply all 5 V requirements of the PK2410, including the interface board and user requirements. If using the PK2410, take care to not overload the processor module regulator. The total dissipation of the regulator should be held to less than 4 W. The processor module regulator dissipates heat through the PK2400's front panel via the mounting standoff and an aluminum strap connected from the regulator to a stud on the front panel.

## Power Failure

Figure C-1 shows the power-failure detection circuitry of the PK2400.



**Figure C-1. Power-Failure Detection Circuitry**

## Power Failure Sequence of Events

The following events occur as the input power fails.

1. The power-management integrated circuit triggers a power-failure **/NMI** (nonmaskable interrupt) when the unregulated direct current input voltage falls below approximately 7.9 volts (as determined by the voltage divider R1 and R15). This provides a “holdup” period ( $t_H$ ) during which the power-fail routine may store important state data. At some point, the raw input voltage level will not exceed the required regulated voltage level by the regulator’s dropout voltage. At that point, the regulated output begins to drop.
2. The power-management integrated circuit triggers a system reset (**/RESET**) when the regulated +5 V supply falls below approximately 4.65 volts. The power-management integrated circuit forces the chip-enable line of the SRAM high (standby mode), which prevents the system from writing to the RAM.
3. If an external backup battery is installed, the watchdog integrated circuit preserves the RAM’s data by switching to battery power when the regulated voltage falls below the battery’s voltage (2.5 V to 4.25 VDC).
4. The power management integrated circuit keeps **/RESET** enabled until the regulated voltage drops below 1 volt. Below 1 volt the power-management integrated circuit ceases operating, and the portion of the circuit that is not backed by battery has already ceased functioning.

### ***Multiple Power-Line Fluctuations***

The power-fail detection system can fail when multiple power fluctuations rapidly follow each other, a relatively common occurrence. If the PK2400’s Z180 microprocessor receives multiple **/NMI**s, it overwrites an internal register, making a correct return from the first **/NMI** impossible. Depending on the number of fluctuations of the raw direct current input (and hence, the number of stacked **/NMI**s), the microprocessor’s stack could possibly overflow, corrupting the program’s code or data.

When the Z180 senses an **/NMI**, it saves the program counter (**PC**) on its processor stack, copies the maskable interrupt flag IEF1 to IEF2, and then zeroes IEF1. The Z180 restores IEF2’s saved state information when it executes a **RETN** (Return from Nonmaskable Interrupt) instruction.

## Holdup Time

Z-World cannot predict how much time is available to save important data. The ratio of a direct current power supply's output capacitor value to the circuit's current draw determines the actual duration of the holdup-time interval,  $t_H$ .

A few milliseconds of computing time remain until the regulated +5 V supply falls below approximately 4.65 volt, even if power cuts off abruptly.

The amount of time depends on the size of the power supply capacitors. The standard wall power supply provides about ten milliseconds. If the power cable is abruptly removed from the PK2400, only the capacitors on the board are available, reducing computing time to a few hundred microseconds. These times can vary considerably, depending on the system's configuration and loads on the 5- or 9-volt power supplies.

The interval between the power-failure detection and the entry to the power-failure interrupt routine is approximately  $6 \mu\text{s}$  or less. Figure C-2 illustrates the power-failure sequence.

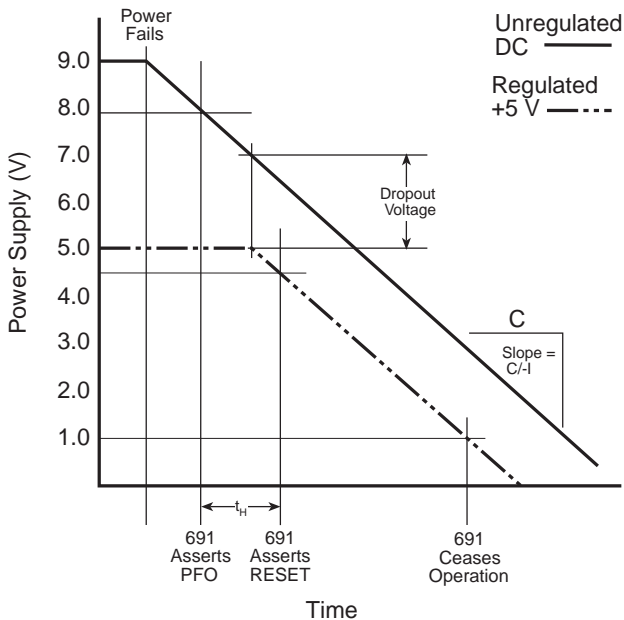


Figure C-2. Power-Failure Sequence



## Recommended Power-Failure Routine

Z-World recommends the following routine to handle an /NMI. The routines monitor the state of the /PFO line (via U7B, PIO 2, and the data bus) to determine if the brownout condition is continuing or if the power has returned to normal levels. If this routine is used, multiple power-failure /NMIs will not be a problem because the routine never returns from the first /NMI unless the power returns.

### Program C-1. Power Failure Routine

```
main(){
    ...
}
...
char dummy[24];      // reserve dummy stack
                    // for /NMI processing
...
#define NMI_BIT 3    // routine will test data
#define NMI PIOB2   // bit 3 to determine
                    // state of /NMI line
#JUMP_VEC NMI_VEC myint
#asm
myint::
    ld    sp,dummy+24 ; force stack pointer
                    ; to top of "dummy"
                    ; array to prevent
                    ; overwriting of code
                    ; or data
;do whatever service, within allowable execution
;time
loop:
    call hitwd      ; make sure no
                    ; watchdog reset
                    ; during brownout
    ld    bc,NMI    ; load the read-NMI
                    ; register to bc
    in    a,(c)     ; read the read-NMI
                    ; register to /PFO
    bit   NMI_BIT, a ; check /PFO status
    jr   z,loop     ; wait until brownout
                    ; condition clears
timeout:
                    ; then... a tight loop
                    ; to force a watchdog
                    ; timeout
    jp   timeout    ; which will reset the
                    ; Z180
#endasm
```

If the direct current input voltage continues to decrease, the controller powers down. The routine calls **hitwd** to make sure that watchdog does not timeout and thereby reset the processor. The controller can continue to run at low voltage (and might not be able to detect the low-voltage condition) because the Z180's **/NMI** input needs to see a high-to-low transition edge.

A situation similar to a brownout will occur if the power supply is overloaded. In such a case, when a high-current load turns on, the raw voltage supplied to the Z180 may dip below 7.9 V. In response, the interrupt routine performs a shutdown that turns off the high-current load, clearing the problem. However, if the cause of the overload persists, the system “hunts,” alternately overloading and then resetting. To correct this situation, obtain a larger, “stiffer” power supply.



## *APPENDIX D: INPUT / OUTPUT MAP AND INTERRUPT VECTORS*

---

## Memory Map

Table D-1 provides the memory map for the flash EPROM and the SRAM.

**Table D-1. Memory Map**

Memory Type	Low Address	High Address
Flash EPROM	00000h	3FFFFh
SRAM	80000h	FFFFFh

## Input / Output Map

Other than peripherals on the Z180, the PK2400 has two byte-wide devices: PIO 1 and PIO 2. Table D-2 lists each PIO's data and control registers.

**Table D-2. PIO Data and Control Registers**

PIO	Name	Description	Address
PIO 1 H3	PIODA	Port A Data Register	00C0h
	PIODB	Port B Data Register	00C1h
	PIOCA	Port A Control Register	00C2h
	PIOCB	Port B Control Register	00C3h
PIO 2 H1	PIODA2	Port A Data Register	0080h
	PIODB2	Port B Data Register	0081h
	PIOCA2	Port A Control Register	0082h
	PIOCB2	Port B Control Register	0083h

The PIO and Z180 pins used for onboard housekeeping functions are listed in Table D-3.

**Table D-3. Dedicated PIO Pins**

Bit	Usage
PIODB.0	RTC Reset (Active Low)
PIODB.1	RS-485 Transmitter Enable (Active High)
PIODB.2	RTC Data (Input/Output)
PIODB.3	RTC Clock
PIODB2.0	ADC Clock
PIODB2.1	ADC Data Out
PIODB2.2	ADC Data In
PIODB2.3	End of Conversion or Power Fail Output (Active Low)
TEND1	Toggle to Reset Watch
INT1	Watchdog Reset Status (Set if WD Timeout)

# Interrupt Vectors

Most interrupt vectors can be altered under program control. The addresses are relative to the start of the interrupt vector page, which is determined by the contents of the I-register. Table D-4 lists the default interrupt vectors set by the boot code in the Dynamic C EPROM.

In order to “vector” an interrupt to a user function in Dynamic C, a directive such as the following is used:

```
#INT_VEC 0x10 myfunction
```

This example causes the interrupt at offset 10H (Serial Port 1 of the Z180) to invoke the function `myfunction()`. The function must be declared with the `interrupt` keyword:

```
interrupt myfunction() {  
  ...  
}
```

*Table D-4. Interrupt Vectors*

Address	Signal	Description
0x00	<b>INT1_VEC</b>	Expansion bus attention INT1 vector
0x02	<b>INT2_VEC</b>	INT2 vector
0x04	<b>PRT0_VEC</b>	PRT Timer Channel 0
0x06	<b>PRT1_VEC</b>	PRT Timer Channel 1
0x08	<b>DMA0_VEC</b>	DMA Channel 0
0x0A	<b>DMA1_VEC</b>	DMA Channel 1
0x0C	<b>CSIO_VEC</b>	Clocked Serial I/O
0x0E	<b>SER0_VEC</b>	Asynchronous Serial Port Channel 0
0x10	<b>SER1_VEC</b>	Asynchronous Serial Port Channel 1
0x12	<b>PIOA_VEC</b>	PIO Channel A (through INT0)
0x14	<b>PIOB_VEC</b>	PIO Channel B (through INT0)
0x22	<b>PIOA2_VEC</b>	PIO Channel A2 (through INT0)
0x24	<b>PIOB2_VEC</b>	PIO Channel B2 (through INT0)

## Interrupt Priorities

Interrupt priorities are listed in Table D-5 from highest to lowest priority.

**Table D-5. Interrupt Priorities**

Interrupt Priorities	
(Highest Priority)	Trap (Illegal Instruction)
	NMI (Non-Maskable Interrupt)
	INT 0 (Maskable Interrupt, Level 0, 3 modes, PIO interrupts)
	INT 1 (Maskable Interrupt, Level 1, PLCBus attention line interrupt)
	INT 2 (Maskable Interrupt, Level 2)
	PRT Timer Channel 0
	PRT Timer Channel 1
	DMA Channel 0
	DMA Channel 1
	Clocked Serial I/O
	Serial Port 0
(Lowest Priority)	Serial Port 1



## *APPENDIX E:* **SERIAL INTERFACE BOARD**

---

Appendix E provides technical details and baud rate configuration data for Z-World's SIB (Serial Interface Board).

## Features

The SIB is an interface adapter used to program the PK2400. The SIB is contained in an ABS plastic enclosure, making it rugged and reliable. The SIB enables the PK2400 to communicate with Dynamic C via the Z180's clocked serial I/O (CSI/O) port, freeing the PK2400's serial ports for use by the application during programming and debugging.

The SIB's 8-pin cable plugs into the target PK2400's processor through an aperture in the backplate, and a 6-conductor RJ-12 phone cable connects the SIB to the host PC. The SIB automatically selects its baud rate to match the communication rates established by the host PC (9600 bps; 19,200 bps; or 57,600 bps). However, the SIB determines the host's communication baud rate only on the first communication after reset. To change baud rates, change the COM baud rate, reset the target PK2400 (which also resets the SIB), then select **Reset Target** from Dynamic C.



Chapter 4 provides detailed information on connecting the SIB to the PK2400.

The SIB receives power and resets from the target PK2400 via the 8-pin connector J1. Therefore, do not unplug the SIB from the target PK2400 while power is applied. To do so could damage both the PK2400 and the SIB; additionally, the target may reset.



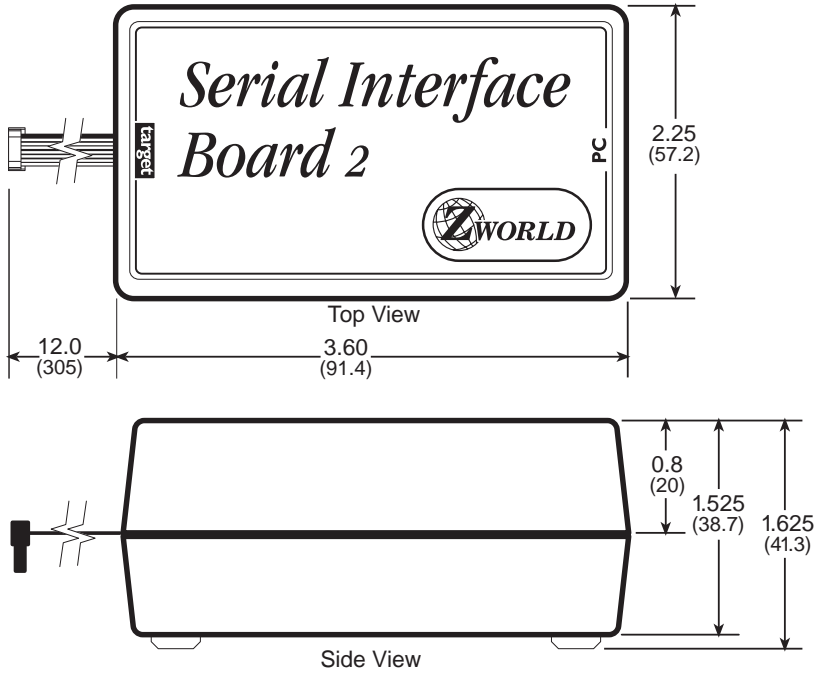
Never connect or disconnect the SIB with power applied to the controller.

The SIB consumes approximately 60 mA from the +5 V supply. The target-system current consumption therefore increases by this amount while the SIB is connected to the PK2400.



# External Dimensions

Figure E-1 illustrates the external dimensions.



**Figure E-1. Serial Interface Board External Dimensions**

This page is blank intentionally.



## *APPENDIX F: **BATTERY***

---

Appendix F provides information about the onboard lithium battery.

## Storage Conditions and Shelf Life

The battery on the PK2400 will provide approximately 9,000 hours of backup for the real-time clock and static RAM as long as proper storage procedures are followed. Units should be kept sealed in the factory packaging at room temperature until field installation. The unit

should not be exposed to extremes in temperature, humidity, or contaminants. The backup time is affected by many factors including the amount of time the unit is unpowered, size of static RAM, temperature, humidity, and exposure to contaminants including dust and chemicals. Protection against environmental extremes will help maximize battery life.

## Instructions for Replacing the Lithium Battery

Use the following steps to replace the battery.

1. Locate the three pins on the opposite side of the printed circuit board that secure the battery to the board.
2. Carefully de-solder the pins and remove the battery. Use a solder sucker to clean up the holes.
3. Install the new battery and solder it to the board. Use only a CR2354-1GU or equivalent.

## Battery Cautions

- **Caution** (English)

There is a danger of explosion if battery is incorrectly replaced. Replace only with the same or equivalent type recommended by the manufacturer. Dispose of used batteries according to the manufacturer's instructions.

- **Warnung** (German)

Explosionsgefahr durch falsches Einsetzen oder Behandeln der Batterie. Nur durch gleichen Typ oder vom Hersteller empfohlenen Ersatztyp ersetzen. Entsorgung der gebrauchten Batterien gemäß den Anweisungen des Herstellers.

- **Attention** (French)

Il y a danger d'explosion si la remplacement de la batterie est incorrect. Remplacez uniquement avec une batterie du même type ou d'un type équivalent recommandé par le fabricant. Mettez au rebut les batteries usagées conformément aux instructions du fabricant.

- **Cuidado** (Spanish)

Peligro de explosión si la pila es instalada incorrectamente. Reemplace solamente con una similar o de tipo equivalente a la que el fabricante recomienda. Deshágase de las pilas usadas de acuerdo con las instrucciones del fabricante.

- **Waarschuwing** (Dutch)

Explosiegevaar indien de batterij niet goed wordt vervagen. Vervanging alleen door een zelfde of equivalent type als aanbevolen door de fabrikant. Gebruikte batterijen afvoeren als door de fabrikant wordt aangegeven.

- **Varning** (Swedish)

Explosionsfara vid felaktigt batteribyte. Använd samma batterityp eller en likvärdigt typ som rekommenderas av fabrikanten. Kassera använt batteri enligt fabrikantens instruktion.

This page is blank intentionally.

**Symbols**

/ADCS .....	28
/NMI .....	28, 43
/PFO .....	43, 97
/RAMSEL .....	43
/RESET .....	43, 95
/WDO .....	43
=(assignment) use .....	82

**A**

A/D converter	
/ADCS .....	28
absolute mode .....	30, 32
ADDCLK .....	28
ADDIN .....	28
ADDOUT .....	28
ARDY .....	28
calibrating .....	38
calibration constants .....	38
simulated EEPROM addresses .....	38
EOC .....	28
internal voltages .....	29
maximum conversion rate .....	28
setup .....	32
useful range .....	29
virtual channels .....	29
AD680 .....	30
AD680JT drift .....	30
ADCLK .....	28
ADCS .....	28
ADDIN .....	28
ADDOUT .....	28
analog inputs .....	68–78
drivers .....	68
functions .....	68
voltage limits .....	28
ARDY .....	28

**B**

baseplate	
power .....	89
battery	
cautions .....	109
replacing .....	108
shelf life .....	108
bitmap graphics .....	48
brownouts .....	43, 98

**C**

calibration constants	
A/D converter .....	38
capacitors	
feedback .....	31
CE compliance .....	15
common problems	
programming errors .....	82
communication	
modem .....	42
connectors .....	94
converter channels	
unconditioned .....	39

**D**

Development Kit .....	13
digital inputs	
pull-up configuration .....	24
digital input and output	
advanced .....	66
digital inputs .....	23
digital outputs .....	25
driver routines	
custom .....	66
drivers	
analog-to-digital converter .....	68
sinking .....	26, 27
sourcing .....	26
DS1302 .....	77–78

Dynamic C .....	60	<b>glBlankScreen</b> .....	70
additional software .....	78	<b>glFontInit</b> .....	71
development software .....	60	<b>glGetBitmap</b> .....	71
sample programs .....	78	<b>glInit</b> .....	70
technical reference .....	60	<b>glPlotCircle</b> .....	73
versions .....	60	<b>glPlotDot</b> .....	70
<b>E</b>		<b>glPlotLine</b> .....	70
<b>ee_rd</b> .....	62	<b>glPrintf</b> .....	73
<b>ee_wr</b> .....	62	<b>glPutBitmap</b> .....	70
<b>EIO_NODEV</b> .....	64, 65	<b>glPutFont</b> .....	72
<b>eioBrdACali</b> .....	69	<b>glSetBrushType</b> .....	69
<b>eioBrdAI</b> .....	68	<b>glVPrintf</b> .....	72
<b>eioBrdDI</b> .....	64	<b>glXFontInit</b> .....	72
<b>eioBrdDO</b> .....	65	<b>glXPutBitmap</b> .....	71
<b>eioBrdInit</b> .....	63, 68, 69	graphic functions .....	69–78
<b>eioErrorCode</b> .....	64, 65	<b>H</b>	
electrical specifications .....	84	Hayes Smart Modem .....	42
environmental specifications .....	84	high-current outputs .....	24
EOC .....	28	high-voltage drivers .....	24
external dimensions		<b>hitwd</b> .....	98
bezel .....	87	holdup time .....	96
PK2400 .....	84, 86	<b>I</b>	
SIB .....	105	IEF1 .....	95
wall-mount enclosure .....	88	IEF2 .....	95
<b>EZIOCMMN.LIB</b> .....	38	inductive loads .....	45
<b>EZIOPK24.LIB</b> .....	60	input/output	
<b>F</b>		advanced programming .....	66
features .....	13	inputs	
filter		conditioned .....	29
low-pass .....	29	digital .....	23
flash EPROM .....	61–78	protected .....	24
nonvolatile storage .....	62–78	installation	
test ratings .....	63	connectors .....	94
<b>float</b>		temperature .....	89
use .....	82	<b>int</b>	
<b>G</b>		type specifier, use .....	82
gain and bias		interrupt line .....	43
initial setup .....	31	interrupts	
voltage .....	31	nonmaskable .....	95
		power failure .....	95



<b>J</b>		<b>literal</b> (C term)	
JP2 .....	24	use .....	82
jumper connections		lithium battery .....	108
J1 .....	91	LM385 .....	30
J2 .....	91	logic-level signals .....	23
		low-pass filter .....	29
<b>K</b>		<b>M</b>	
KDR.LIB .....	61	mechanical specifications .....	84
kdiBuzSw .....	73	memory .....	61
kdiELSw .....	73	modem	
kdiSetContrast .....	73	communication .....	42
keypad		null .....	42
features .....	46	<b>N</b>	
functions .....	69–78	NMI .....	28, 43, 95, 97
initialization .....	74	nonlatching relays .....	63
reading .....	74, 75	nonmaskable interrupts. <i>See</i> NMI	
software drivers .....	46	nonvolatile storage .....	62–78
states .....	74	<b>O</b>	
KP.LIB .....	61, 69	off_485 .....	67
kpDefGetKey .....	75	on_485 .....	67
kpDefInit .....	75	op-amp	
kpDefStChgFn .....	74	offset voltage .....	30
kpInit .....	74	test points .....	37
kpScanState .....	74	operating modes	
		program mode .....	53
<b>L</b>		run mode .....	53
latching relays .....	63	output	
LCD		digital .....	25
backlight .....	73	<b>P</b>	
backlighting .....	46	PFR .....	43
buzzer .....	73	PIO	
contrast .....	73	mode 3 .....	28
features .....	46	PIO chips .....	66
functions .....	69–78	U2 .....	66
graphic		U9 .....	66
bitmaps .....	71	PK2400	
drawing .....	69, 70, 73	dimensions .....	86
fonts .....	71, 72	Port A	
initialization .....	70	A/D operation .....	28
read .....	70		
writing .....	72, 73		
graphic coordinates .....	49		
software drivers .....	46, 48		

Port B .....	28	resistor pack removal .....	58
ports		resistors	
serial .....	42	choosing values .....	34
RS-232 .....	41, 53	confirming values .....	36
RS-485 .....	41	gain and bias .....	32
power		values .....	32
direct current .....	89	reference .....	39
SIB2 .....	104	temperature coefficient .....	30
power failure .....	43	tolerance .....	35
brownouts .....	98	user-selectable .....	31
holdup time .....	96	RJ-12 .....	104
interrupt routine .....	98	RS-232	
interrupts .....	95	development .....	60
overload .....	98	RS-485	
routine .....	97	developing network .....	57
power supervisor IC		functions .....	67
hysteresis .....	43	run mode .....	53
noise .....	43	standalone .....	56
reset threshold .....	43		
power-failure routine .....	95	<b>S</b>	
power-on reset .....	43	sample programs	
program mode .....	53	<b>SAMPLES\PK24XX</b> .....	65, 78
programming mode		serial communication	
returning to .....	56	functions .....	67
		ports .....	41
<b>R</b>		support .....	42
RAMSEL .....	43	SIB .....	42
<b>Read1302</b> .....	77	baud rate .....	104
<b>ReadBurst1302</b> .....	77	connection to PK2400 .....	55
<b>ReadRam1302</b> .....	77	development .....	60
real-time clock .....	76–78	development with .....	55
storing contents .....	76	dimensions .....	89, 105
time and date structure .....	76	power .....	104
writing contents .....	76	sinking drivers .....	26, 27
REF+ .....	28	snubber .....	45
regulated input voltage .....	95	software	
relays		input channel assignments .....	64
connections .....	45	output channel assignments .....	65
maximum load .....	44	power fail flag .....	78
nonlatching .....	44	reset .....	78
outputs .....	44	simulated EEPROM .....	78
terminal connections .....	45	supplied .....	60
RESET .....	43	watchdog .....	78

sourcing drivers .....	26	UDN2985A .....	26
specifications		ULN2803 .....	26, 27
electrical .....	84	unipolar variations .....	34
environmental .....	84	unregulated input voltage .....	95
mechanical .....	84		
standby mode .....	95	<b>V</b>	
<b>T</b>		VBAT .....	43
temperature .....	89	voltage	
measuring .....	39	divider .....	34
terminator resistor .....	58	gain and bias .....	31
test voltages .....	29	VR0- .....	34
tH .....	96	VRI- .....	34
thermistor .....	39	VREF .....	34
output table .....	40	<b>W</b>	
<b>tm_rd</b> .....	76	watchdog timer .....	43
<b>tm_wr</b> .....	76	WDO .....	43
troubleshooting		<b>WINTEK.LIB</b> .....	69
baud rate .....	81	<b>Write1302</b> .....	77
cables .....	80	<b>WriteBurst1302</b> .....	77
communication		<b>WriteFlash</b> .....	61, 62
port .....	80	<b>WriteRam1302</b> .....	77
communication mode .....	81		
communications		<b>X</b>	
port .....	81	XModem	
expansion boards .....	80	file transfer .....	42
grounds .....	80	<b>Z</b>	
memory size .....	81	Zilog PIO chips .....	66
operating mode .....	81		
power supply .....	80		
repeated resets .....	81		
<b>U</b>			
U4 .....	30		
U5 .....	30		
U7 .....	28		
U7B .....	97		

This page is blank intentionally.



This page is blank intentionally.



**Z-World, Inc.**

2900 Spafford Street  
Davis, California 95616-6800 USA

Telephone: (530) 757-3737  
Facsimile: (530) 753-5141  
Web Site: <http://www.zworld.com>  
E-Mail: [zworld@zworld.com](mailto:zworld@zworld.com)

Part No. 019-0052  
Revision F