



PK2300

C-Programmable Controller

User's Manual

Revision E

PK2300 User's Manual

Part Number 019-0040 • Revision E

Last revised on March 2, 2000 • Printed in U.S.A.

Copyright

© 1999 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[®] is a registered trademark of Z-World, Inc.
 - Windows[®] is a registered trademark of Microsoft Corporation
 - PLCBus[™] is a trademark of Z-World, Inc.
-

Notice to Users

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

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

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

Company Address



Z-World, Inc.

2900 Spafford Street
Davis, California 95616-6800
USA

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

TABLE OF CONTENTS

About This Manual	vii
Chapter 1: Overview	11
PK2300 Overview	12
Features	14
Flexibility and Customization	15
Optional Accessories	15
Development Kit	16
Software Development and Evaluation Tools	16
CE Compliance	16
Chapter 2: Getting Started	17
Connecting the PK2300 to Your PC	18
Running Dynamic C	20
Selecting Communications Rate, Port, and Protocol	20
Running a Sample Program	22
Chapter 3: Input/Output Configuration	25
PK2300 Components	26
Flexible Input/Output Summary	27
Configuring Serial Communication	29
Configuring Inputs and Outputs	30
Selecting IN-08 and IN-09, or RMI Input + and -	30
Selecting IN-12 thru IN-16, or OUT-04 thru OUT-08	31
Selecting IN-10 and IN-11 or RS-485+ and RS-485-	33
CMOS Outputs	34
Chapter 4: System Development	35
PK2300 Operating Modes	36
Changing the PK2300's Operating Mode	37
Running a Program	38
Using the Digital Inputs/Outputs	39
Protected Digital Inputs	39
How to Read the Inputs	40
High-Current Outputs	42
How to Use the Outputs	42

Resistance Measurement Input (RMI)	45
How to Use the Resistance Measurement Input	46
RMI Theory of Operation	49
Serial Communication	50
RS-232 Communication	50
RS-232 Connector Pinouts	50
RS-485 Communication Network	51
Termination and Bias Resistors	52
Additional Features	54
PWM Outputs	54
How to Use the PWM Feature	54
User-Programmable LEDs	58
Memory	58
Real-Time Clock (RTC)	59
Power Supervisor	60
Resetting the Processor	61
PK2300 Subsystems	62

Chapter 5: **Software Reference** **63**

Supplied Software	64
Digital Inputs/Outputs	65
Level-Sensitive Interrupts	68
Interrupt Service Routines (ISR)	69
Resistance Measurement Input	70
PWM Outputs	72
Additional Software Feature References	74
Advanced Input/Output Programming	75
Digital Input Addressing Detail	76
Digital Output Addressing Detail	77
LED Addressing Detail	78
RS-485 Driver IC Addressing Detail	78
Resistance Measurement Input Addressing Detail	79
PWM Addressing Detail	81
PWM Advanced Programming Functions	86

Appendix A: **Troubleshooting** **89**

Out of the Box	90
Dynamic C Will Not Start	91
Dynamic C Loses Serial Link	91
PK2300 Repeatedly Resets	91
Common Programming Errors	91

Appendix B: Specifications	93
Electrical and Mechanical Specifications	94
PK2300 External Dimensions	95
Factory Default Jumper Positions	96
Protected Digital Inputs	98
Frequency Response for IN-01 to IN-05, and IN-08 to IN-16	98
Frequency Response for IN-06 and IN-07	99
Customization	99
Frequency Response and Input Range	99
Default Pull-Up Assignments	100
High-Current Drivers	100
Function of “K”	101
Appendix C: Serial Interface Board 2	103
Introduction	104
External Dimensions	105
Appendix D: Sinking vs. Sourcing Drivers	107
Selecting Sourcing or Sinking Drivers	109
Sinking Driver (Low-Side Drive)	109
Sourcing Driver (High-Side Drive)	110
Appendix E: Power Management	111
Power-Failure Detection Circuitry	112
Power Failure Sequence of Events	112
Appendix F: Enclosure Mounting	115
Enclosure Mounting Considerations	116
Mounting Controllers on DIN Rails	118
Appendix G: Nonvolatile Storage	119
Appendix H: I/O Map and Interrupt Vectors	121
CM7200 Input/Output Map	122
Real-Time Clock Registers	124
Other Input/Output Addresses	125
Interrupt Vectors	126
Interrupt Priorities	127

Appendix I: Battery **129**
Storage Conditions and Shelf Life 130
Instructions for Replacing the Lithium Battery 130
Battery Cautions 131

Index **133**

Schematics

ABOUT THIS MANUAL

This manual provides instructions for installing, testing, configuring, and interconnecting the Z-World PK2300 controller.

Instructions to get started using Dynamic C[®] programming functions are included. 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 a PK2300 will control.
- 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
and/or

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	Tip	Tip
	Caution		High Voltage
	Factory Default		

Conventions

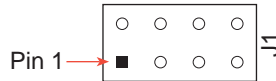
Table 3 lists and defines the typographic conventions that may be used in this manual.

Table 3. Typographic Conventions

Example	Description
while	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).
Edit	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.

Blank



CHAPTER 1: **OVERVIEW**

Chapter 1 provides a comprehensive overview and description of the PK2300. The following sections are included.

- PK2300 Overview
- Features
- Flexibility and Customization
- Developer's Kit

PK2300 Overview

PK2300 Series packaged controllers are small, powerful and extremely flexible system controllers. The PK2300 is programmed using Dynamic C (Z World's version of the C programming language).

The PK2300 Series consists of two separate models.

- The PK2300 is a full-featured controller that includes all possible I/O features. It is recommended for initial system development.
- The PK2310 is identical to the PK2300, with the following exceptions.
 - The Real Time Clock is not included.
 - The Resistive Measurement Input is not included.

As shown in Figure 1-1, the PK2300 is housed in a rugged enclosure that has multiple mounting options including DIN rail.

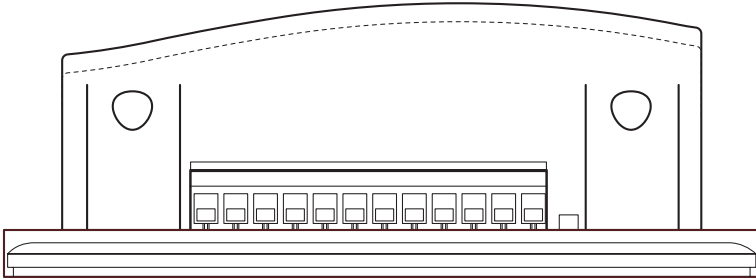




Figure 1-1. PK2300 Enclosure, Side View (Left Side)

 See Appendix F, “Enclosure Mounting,” for further details.

 Appendix B provides detailed specifications for the PK2300.

The PK2300 is available with either quick-release pluggable terminals or fixed screw terminals.

Figure 1-2 illustrates the PK2300 board layout.

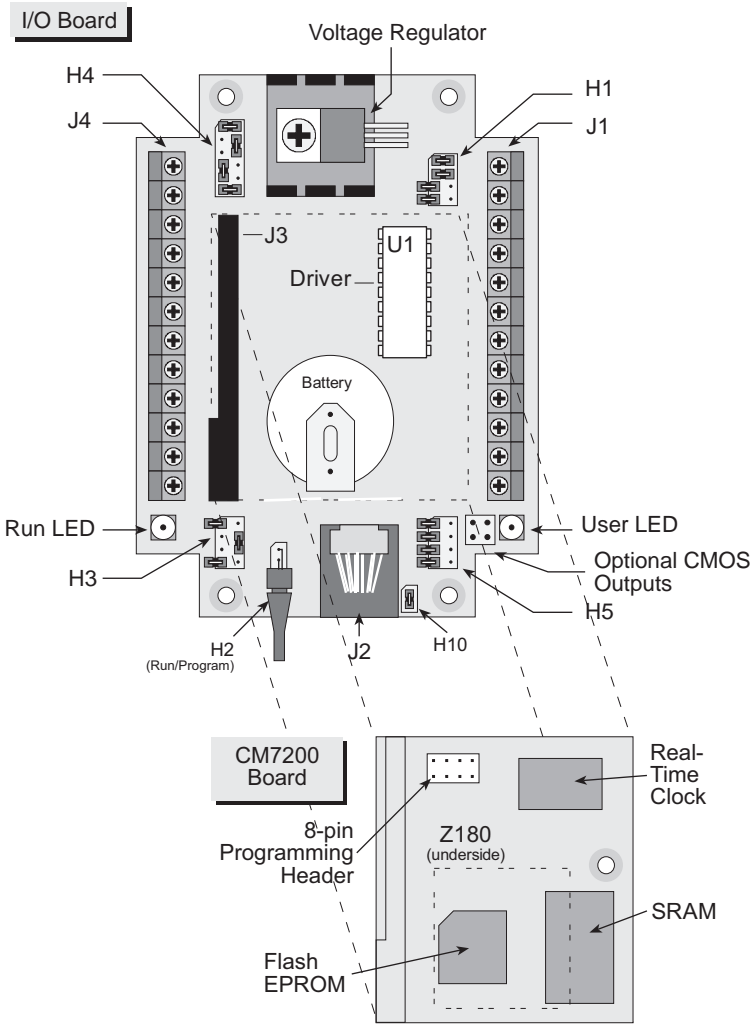


Figure 1-2. PK2300 Component Layout (Top View, Case Removed)

Features

The PK2300 includes the following features.

- Up to 16 protected digital inputs and as many as 8 CMOS or high-current driver outputs. Two of the protected digital inputs have software assignable level sensitive interrupts.
- A Resistance Measurement Input (RMI) circuit allows effortless interfacing to two terminal resistance sensors.
- Two serial communication ports are available. Both RS-232 and RS-485 standards are supported.
- Nineteen pins (out of the 24 total) are provided for assigning various I/O combinations. One RJ-12 modular jack is also provided for RS-232 communications.

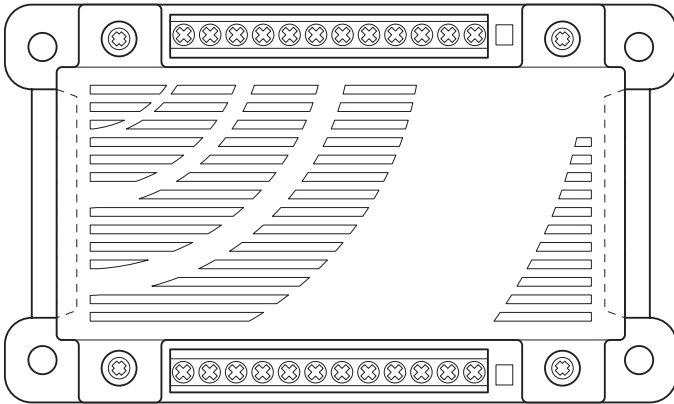


Figure 1-3. PK2300 Top View Showing Screw Terminal Locations

- Seven of the digital outputs can optionally provide Pulse Width Modulation under software control.
- To complete the features, our Dynamic C library provides an assortment of C functions and I/O drivers allowing designers to easily control the inputs, outputs, and serial channels of the PK2300 Series controllers.
- Four optional 5 V CMOS outputs are accessible via through-hole pads on the printed circuit board.
- Two programmable LEDs.

Flexibility and Customization

The PK2300 was designed with application flexibility in mind. Two levels of flexibility allow you to select the type of I/O specific to your needs.

- **Flexibility Level 1 — Out of the Box**

You can immediately change input and output assignments on nine of the available 24 terminals. I/O features are assignable via jumpers — allowing quick tailoring to your specific I/O count. In many cases, this level of flexibility is all that’s required.



See Chapter 3, “Input/Output Configuration,” for further details.

- **Flexibility Level 2 — Customization**

The circuit layout of the PK2300 has been optimized for quick-turn custom manufacturing. Once your application prototype is defined, our automatic surface mount manufacturing can deliver the PK2300 Series controller with the *exact* hardware your application requires. Example: Jumper blocks can be configured and hardwired at the factory.



For details on PK2300 customization, please contact your Z-World Sales Representative at (530) 757-3737.

Optional Accessories

The following accessories are available for the PK2300.

- Development Kit with manual, schematics, programming cable, AC adapter, and sourcing high-current driver chip
- DIN rail mounting kit
- Sourcing driver kit
- Thermistor that can be used as a rugged temperature sensor
- Serial Interface Board to allow programming through a special programming port, leaving the serial channels available for the application



For ordering information, or for more details about the various options and prices, call your Z-World Sales Representative at (530) 757-3737.

Development Kit

The PK2300 Development Kit includes all the items necessary for software and hardware development using the PK2300. The kit includes the following items.

- **PK2300 User's Manual** (with schematics).
- 12 V DC wall power supply.
- Sourcing high-current drivers, cables, and adapters.

Software Development and Evaluation Tools

Dynamic C, Z-World's Windows-based real-time C language development system, is used to develop software for the PK2300. The host PC downloads the executable code through the Serial Interface Board 2 or one of the serial ports to flash EPROM.



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

CE Compliance

The PK2300 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 PK2300 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 PK2300 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.



CHAPTER 2: **GETTING STARTED**

Chapter 2 provides instructions for connecting the PK2300 to your PC, and running a sample program on the PK2300. The following sections are included.

- Connecting the PK2300 to Your PC
- Establishing Communication with the PK2300
- Running a Sample Program

Connecting the PK2300 to Your PC

The PK2300 can be programmed by a PC through an RS-232 port using the programming cable provided in the Developer's Kit.

To connect the PK2300 to a PC:

1. Install Dynamic C as described in the Dynamic C manuals.
2. Connect the 12 V DC power supply to the +DC and GND terminals as shown in Figure 2-1.

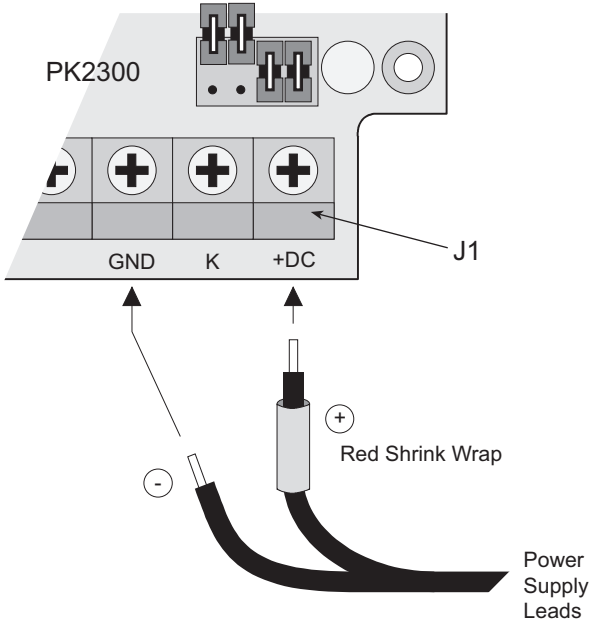


Figure 2-1. Power Supply Connection



Do not plug the transformer into the wall until all the connections and jumpers have been set.



If a transformer other than the one supplied with the developer's kit is used, ensure that the input voltage specifications (9 V to 24 V DC) are not exceeded. Appendix B contains complete specifications for the PK2300.

Tip

Use the programming cable supplied with the Developer's Kit to avoid problems with mismatched plugs.

3. Ensure that the Run/Program jumper (H2) shown in Figure 2-2 is installed.

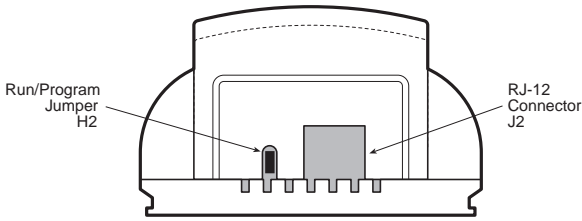


Figure 2-2. Run/Program Jumper Location



Pay particular attention to the installation of the Run/Program jumper at H2. It is possible to install the jumper so that the pins are not connected. The PK2300 will not be in Program Mode if this jumper is installed incorrectly.

4. Establish a serial communication link.

Option 1—Via RS-232 Serial Port

Use the adapter and the programming cable supplied with the developer's kit to connect the PK2300's RJ-12 (J2) socket to the appropriate COM port of your computer. See Figure 2-3.

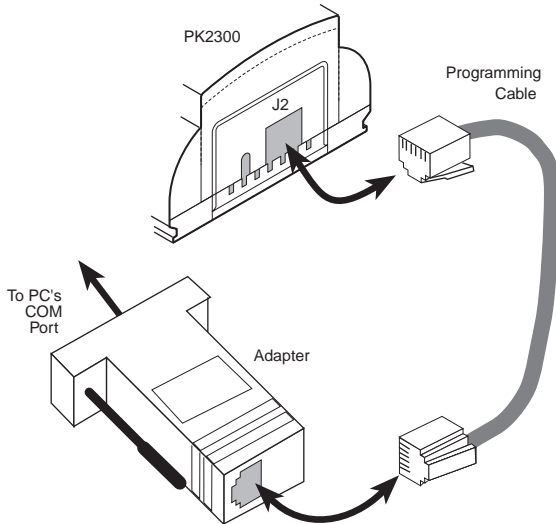


Figure 2-3. Programming Connections

Option 2—Via Optional Serial Interface Board 2

The Serial Interface Board 2 (SIB2) is an optional development tool that allows both of the PK2300's RS-232 ports to be available to an application. The PK2300 uses a Z-World CM7200 core module as the microprocessor core. The CM7200 has an additional programming port, JP1, that connects to the SIB via a 2 mm, 8-pin connector, bypassing the PK2300's RJ-12 modular jack (J2).

Connect an RJ-12 cable between the SIB2 and the RJ-12/DB9 adapter attached to the host PC, as shown in Figure 2-4.

Plug the SIB2's 8-pin connector into header JP1 located on the CM7200.



Observe the polarity of the cable used to connect the SIB2 to JP1. The side of the cable closest to pin 1 is marked in blue, as indicated in Figure 2-4.



See Appendix C, “Serial Interface Board 2,” for additional information on the SIB2.

5. Plug in the transformer to an AC source only after double-checking all connections. The PK2300 is now ready for programming.

Running Dynamic C

Double-click the Dynamic C icon to start the software. Note that the PC attempts to communicate with the PK2300 each time Dynamic C is started. No error messages are displayed once communication is established.



See Appendix A, “Troubleshooting,” if an error message such as **Target Not Responding** or **Communication Error** appears.



Once the necessary changes have been made to establish communication between the host PC and the PK2300, use the Dynamic C shortcut **<Ctrl Y>** to reset the controller and initiate communication.

Selecting Communications Rate, Port, and Protocol

The communication rate, port, and protocol are all selected by choosing **Serial Options** from Dynamic C's **OPTIONS** menu.

The PK2300 supports a communication rate up to 57,600 bps. However, the Dynamic C software shipped by Z-World may be initialized for a different rate. To begin, select a communication rate of 57,600 bps.

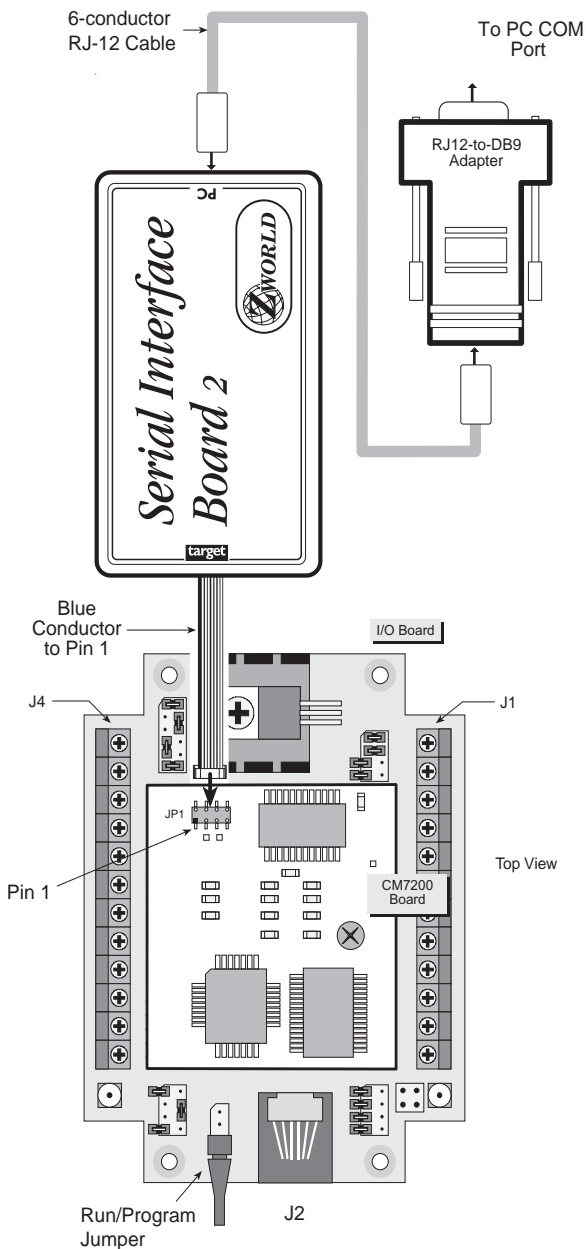


Figure 2-4. Programming Connections to Serial Interface Board

A slower rate may be required, in which case the controller should be reset with a **Ctrl Y** after the lower rate is selected. The Serial Interface Board automatically adjusts to the PC's communication rate at 9600 bps, 19,200 bps, 28,800 bps, and 57,600 bps.

Make sure that the PC serial port used to connect the serial cable (COM1 or COM2) is the one selected in the Dynamic C **OPTIONS** menu. Select the 1-stop-bit protocol.

Running a Sample Program

1. Open the sample program (**PK23FLSH.C**) located in the Dynamic C **SAMPLES\PK23XX** subdirectory. This program flashes the PK2300's LEDs at differing flash rates.
2. Compile the program by pressing **F3** or by choosing **Compile** from the **Compile** menu. Dynamic C compiles and downloads the program into the PK2300's flash memory.

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



See Appendix A, "Troubleshooting," if an error message such as **Target Not Responding** or **Communication Error** appears.

3. Run the program by pressing **F9** or by choosing **Run** from the **Run** Menu.
4. To halt the program, press **<Ctrl Z>**. This action halts program execution.
5. To restart program execution when required, press **F9**.

The Dynamic C **SAMPLES** subdirectory contains other sample programs that illustrate the features of the PK2300. These programs may be used as a basis for new applications.

PK23FLSH.C

```
/*
PK23FLSH.C
This program flashes the LEDs of the PK2300 controller.
The two LEDs have different flash rates.
*/
#include vdriver.lib

main() {
    VdInit();
    for (;;) {
        costate {
            output(0x4141,1);           // Turn on LED D1
            waitFor(DelayMs(400));      // Delay .4 seconds
            output(0x4141,0);          // Turn off LED D1
            waitFor(DelayMs(550));      // Delay .55 seconds
        }
        costate {
            output(0x4142,1);           // Turn on LED D2
            waitFor(DelayMs(600));      // Delay .6 seconds
            output(0x4142,0);          // Turn off LED D2
            waitFor(DelayMs(380));      // Delay .38 seconds
        }
    }
}
```

Blank



CHAPTER 3: ***INPUT/OUTPUT CONFIGURATION***

Chapter 3 describes the built-in flexibility of the PK2300 Series controller and describes how to configure the available inputs/outputs (I/O).

The following sections are included.

- PK2300 Components
- Flexible I/O Summary
- Configuring Serial Communications
- Configuring Inputs and Outputs
- Selecting IN-08 and IN-09, or RMI Input + and -
- Selecting IN-12 through IN-16, or OUT-04 through OUT-08
- Selecting IN-10 and IN-11 or RS-485 + and -

PK2300 Components

The PK2300 provides six types of I/O for your application.

- Protected digital inputs
- High-current driver outputs
- Serial communication channels
- Resistance measurement circuit
- LEDs
- +5 V CMOS I/O

At the heart of every PK2300 lies one of Z-World's CM7200 microprocessor core modules. The CM7200 provides the processor, real time clock, supervisor, memory, and control of the various I/O.

Figure 3-1 illustrates the headers and the signal names for the PK2300's pins. Use this figure for reference throughout this chapter.

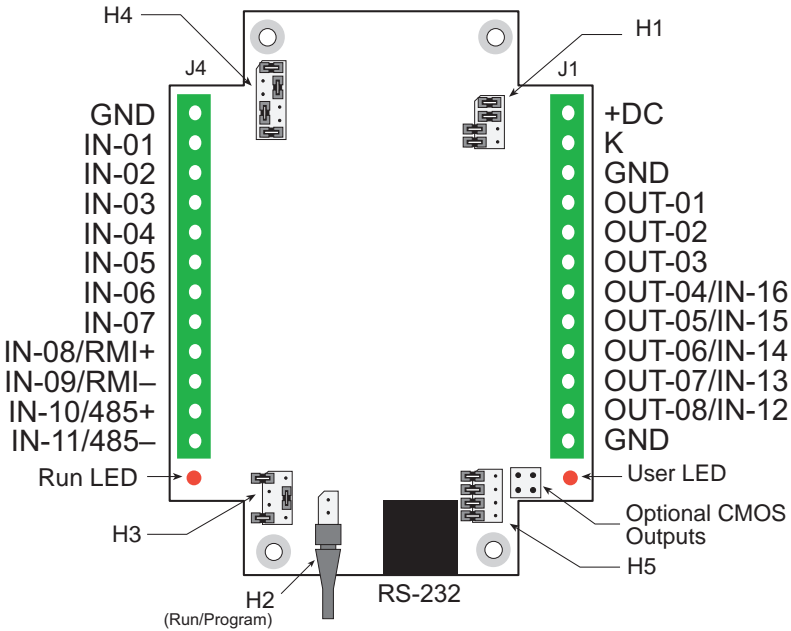



Figure 3-1. PK2300 Pin and Header Layout (Top View, Case Removed)

Flexible Input/Output Summary

The PK2300 includes 19 pins dedicated to I/O. Some have fixed functions, while others can be configured using jumpers. You can mix and match inputs and outputs to suit your particular application.

Table 3-1 and Table 3-2 list all possible I/O function combinations.


**Table 3-1. PK2300 I/O Functions
J4 Pins 1 through 12**

Pin	Label	Function 1 	Function 2 (user assignable)
1	GND	Ground	N/A
2	IN-01	Protected Digital Input 1	N/A
3	IN-02	Protected Digital Input 2	N/A
4	IN-03	Protected Digital Input 3	N/A
5	IN-04	Protected Digital Input 4	N/A
6	IN-05	Protected Digital Input 5	N/A
7	IN-06	Protected Digital Input 6	Protected Digital Input 6 with level-sensitive interrupt (/INT0)
8	IN-07	Protected Digital Input 7	Protected Digital Input 7 with level-sensitive interrupt (/INT1)
9	IN-08/ RMI+	Protected Digital Input 8	Resistive Measurement Input +
10	IN-09/ RMI-	Protected Digital Input 9	Resistor Measurement Input -
11	485+/ IN-10	RS-485+ serial communication	Protected Digital Input 10
12	485-/ IN-11	RS-485- serial communication	Protected Digital Input 11



The optional level-sensitive interrupts (pins 7 and 8) are software-assignable. Refer to Chapter 5, “Software Reference,” for further details.

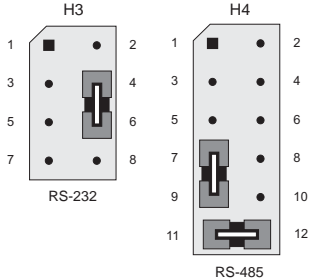

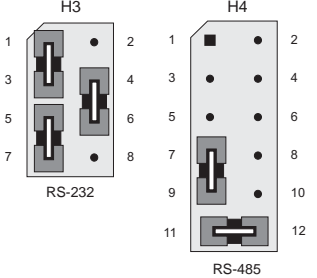
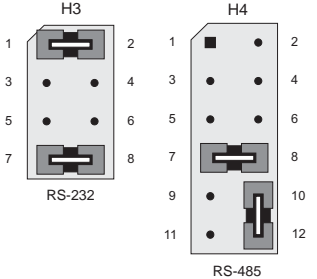
**Table 3-1. PK2300 I/O Functions
J1 Pins 1 through 12**

Pin	Label	Function 1 	Function 2 (user assignable)
1	+DC	Power supply input (9 V to 24 V DC)	N/A
2	K	High-current driver supply	N/A
3	GND	Ground (power supply)	N/A
4	OUT-01	High-Current Output 1	N/A
5	OUT-02	High-Current Output 2	N/A
6	OUT-03	High-Current Output 3	N/A
7	OUT-04/ IN-16	High-Current Output 4	Protected Digital Input 16
8	OUT-05/ IN-15	High-Current Output 5	Protected Digital Input 15
9	OUT-06/ IN-14	High-Current Output 6	Protected Digital Input 14
10	OUT-07/ IN-13	High-Current Output 7	Protected Digital Input 13
11	OUT-08/ IN-12	High-Current Output 8	Protected Digital Input 12
12	GND	Ground	

Configuring Serial Communication

The PK2300 has two serial-communication ports. As shown in Table 3-3, there are three *mutually exclusive* combinations of the RS-232 and RS-485 serial protocols that can be applied to the ports. Note that Configurations I, II, and III are used for reference later in this chapter.

Table 3-3. Serial Communication Configurations

Header Jumpers	Configurations
<p style="text-align: center;">Configuration "I"</p>  <p style="text-align: center;">RS-232 RS-485</p>	<p>One 3-wire RS-232 (no handshaking) and one RS-485.</p> <div style="text-align: center;">  </div>
<p style="text-align: center;">Configuration "II"</p>  <p style="text-align: center;">RS-232 RS-485</p>	<p>One 5-wire RS-232 (RTS/CTS handshaking) and one RS-485.</p>
<p style="text-align: center;">Configuration "III"</p>  <p style="text-align: center;">RS-232 RS-485</p>	<p>Two 3-wire RS-232 (no handshaking).</p>

Configuring Inputs and Outputs

This section describes how to configure the I/O to your specific application requirements.

Selecting IN-08 and IN-09, or RMI Input + and -

Header **H4**'s jumpers can be set to configure **J4** pins 9 and 10 as Protected Digital Inputs IN-08 and IN-09, or as Resistance Measurement Inputs (RMI+ and RMI-).



The RMI function is not available on the PK2310.

Figure 3-2 illustrates header **H4**.

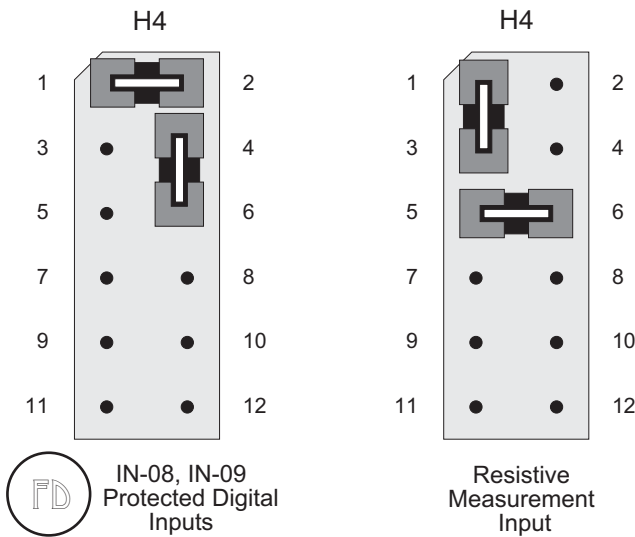


Figure 3-2. Header H4 Configurations



Header **H4** must be configured for either Protected Digital Input operation or RMI operation. In either case, ensure that both jumpers shown above are installed.

Selecting IN-12 thru IN-16, or OUT-04 thru OUT-08

This section provides information for setting the jumpers on headers **H1** and **H5** to configure J1 pins 7 through 11 as High-Current Outputs OUT-04 through OUT-08, or as Protected Digital Inputs IN-12 through IN-16.

Figure 3-3 illustrates the jumper settings for headers **H1** and **H5**. You can set up each channel individually as desired.

- When a jumper is installed, the corresponding pin is configured as a protected digital input.
- When a jumper is not installed, the corresponding pin is configured as a high-current output.

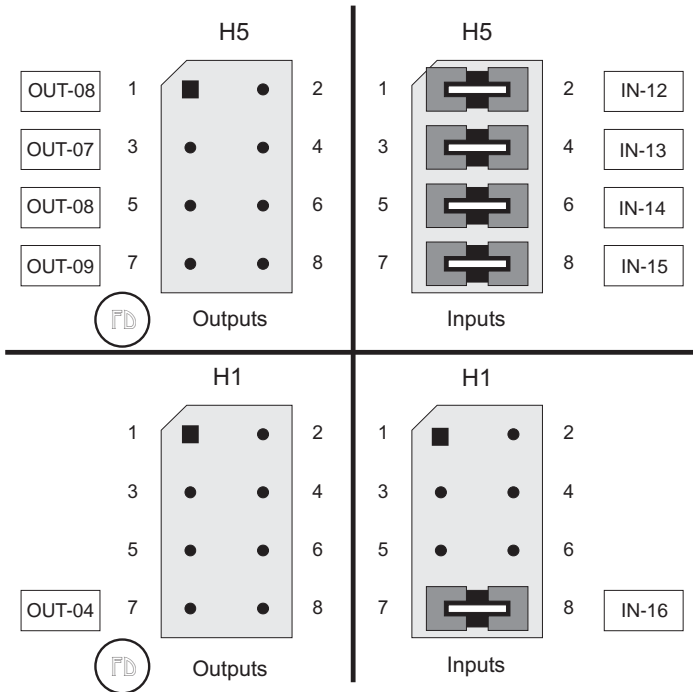


Figure 3-3. Header H5 and H1 Configurations



When you install jumpers on headers **H1** or **H5** to connect an input IC to the terminals, the output IC remains hard-wired to the terminal. Do not enable (turn on) the output drivers on any channel that have been configured as inputs via headers **H1** and **H5**.

The following example shows how to configure J1 pins 9, 10, and 11 as Protected Digital Inputs IN-12 through IN-14, and J1 pins 7 and 8 as High-Current Outputs OUT-05 and OUT-04.

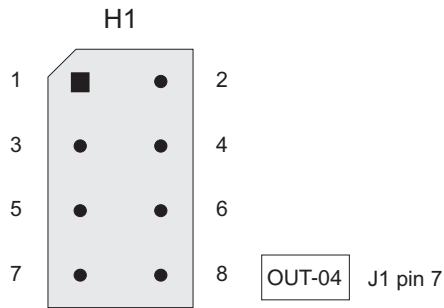
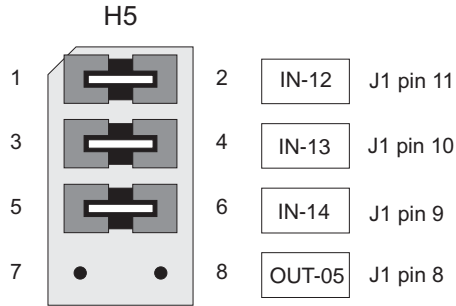


Figure 3-4. Sample Configuration

Selecting IN-10 and IN-11 or RS-485+ and RS-485–

This section provides information for setting the jumpers on header **H4** to configure J4 pins 11 and 12 as Protected Digital Inputs IN-10 and IN-11, or as an RS-485 communication port (RS-485+ and RS-485–).

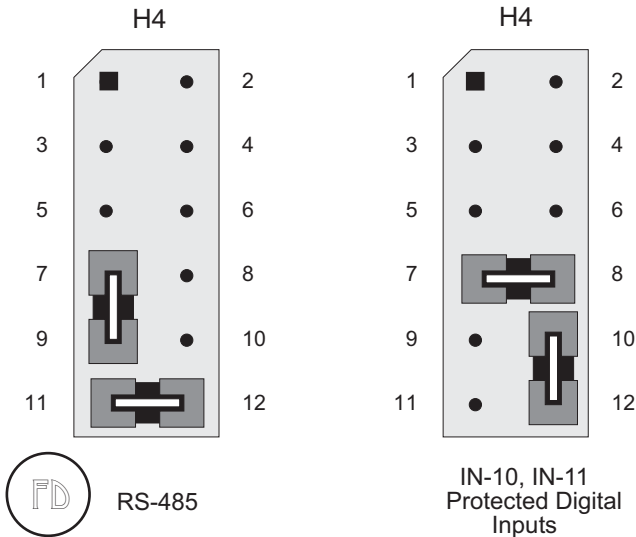


Figure 3-5. Header H4 Settings for RS-485 and IN-10, IN-11



Header H4 must be configured either as a protected digital input or for RS-485 operation. In either case, both sets of jumpers must be installed as shown in Figure 3-5.

Tip

Use tinned wires for connections to the terminals on J1 and J4 to avoid wire oxidation and poor connections. When possible, use nonstranded wire.

CMOS Outputs

The eight output channels may be configured as CMOS outputs by removing the driver IC at U1 and jumpering across the driver IC socket as shown in Figure 3-6.

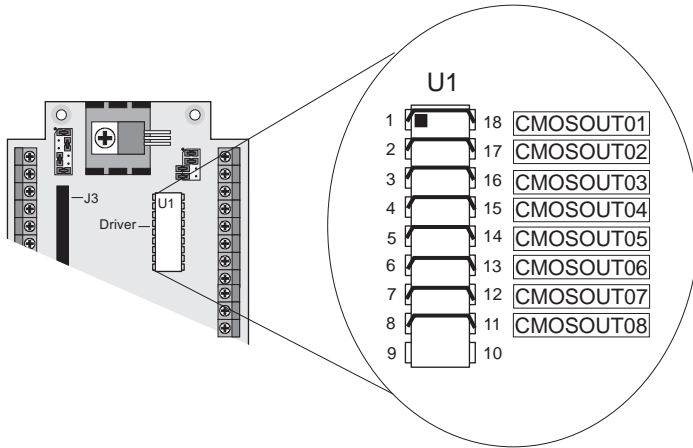


Figure 3-6. Configuration of Driver IC Socket for CMOS Outputs



Note that high-current outputs are unavailable when the driver IC at U1 has been removed and the socket is jumpered as shown in Figure 3-6.

Protected digital inputs IN-12 to IN-16 may be selected instead of CMOS outputs on J1 pins 7-11 by jumpering the pins on headers H1 and H5 as shown in Figure 3-3.

Four additional optional CMOS outputs are available next to the User LED, as shown in Figure 3-1.



The PK2300 is available in quantity with the driver IC removed and surface-mount jumpers factory-installed to customer specifications in socket U1. For ordering information, call your Z-World Sales Representative at (530) 757-3737.



CHAPTER 4: **SYSTEM DEVELOPMENT**

Chapter 4 describes how to use the features of the PK2300 Series controller. The following major sections are included.

- PK2300 Operating Modes
- Running a Program Stand Alone
- Using the Digital I/O
- Protected Digital Inputs
- High-Current Outputs
- Resistance Measurement Input (RMI)
- Serial Communication
- RS-485 Communication Network
- PWM Outputs
- User-Programmable LEDs
- Memory
- Real-Time Clock (RTC)
- Power Supervisor

PK2300 Operating Modes

The PK2300 has two *mutually exclusive* operating modes, only one of which responds to Dynamic C. Each mode is explained in detail below.

- **Program Mode**

In Program mode, the PK2300 controller runs under the control of a PC running Dynamic C. The PK2300 must be in this mode when you attempt to compile a program to the PK2300 or debug a program.

In Program mode, the PK2300 matches the baud rate of the PC COM port up to 57,600 bps. Baud rates of 9600 bps, 19,200 bps, 28,800 bps, and 57,600 bps are possible. User LED, shown in Figure 4-1, is enabled to remain on continuously; Run LED is disabled, or off. Both LEDs are available for the application being developed.

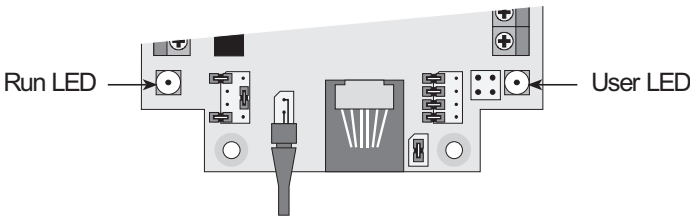


Figure 4-1. Run and User LED Locations

- **Run Mode**

In Run Mode, the PK2300 controller runs *standalone*. At power-up, the PK2300 checks to see if its onboard memory contains a program. If such a program exists, the PK2300 controller executes the program immediately after power-up.

Both LEDs are under the control of the application while the PK2300 is in Run Mode. The default is for the Run LED to be off and for the User LED to be on.



The PK2300 in Run Mode will not respond to Dynamic C running on a PC. Programs cannot be compiled or debugged while the PK2300 is in Run Mode.



In Run Mode, the PK2300 takes approximately 60 ms to boot up and begin execution of a program in RAM.

Table 4-1 lists the permissible activities in the Program/Run modes.

Table 4-1. Permissible Activities in Program/Run Modes

Operating Mode	Jumper H2	Permissible Activities
Program Mode	Installed	<ul style="list-style-type: none">• Compile a program• Run a program under debugger control• Run a program without “polling.” See your Dynamic C manuals for a description of program polling.
Run Mode	Removed	Run program (in your application).

Changing the PK2300’s Operating Mode

These steps describe how to change the PK2300 operating mode.

1. Remove power from the PK2300.
2. Locate the **Run/Program** jumper at **H2** that protrudes through the end of the case next to the RJ-12 connector (**J2**). Figure 4-2 shows the location of the **Run/Program** jumper at **H2**.

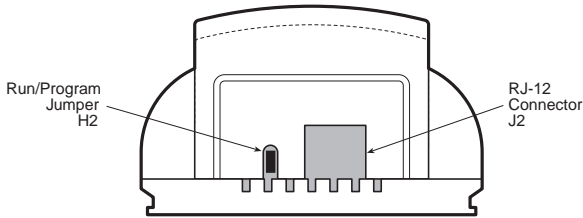


Figure 4-2. Location of Run/Program Jumper at H2

3. Select the desired jumper state:
 - Install jumper at **H2** to select Program Mode.
 - Remove jumper at **H2** to select Run Mode.
4. Reapply power to restart the PK2300 in the selected mode.



Pay particular attention to the installation of the Run/Program jumper at H2. It is possible to install the jumper so that the pins are not connected. The PK2300 will not be in Program Mode if the **Run/Program** jumper is installed incorrectly.

Running a Program

1. Place the PK2300 in Program Mode (with the jumper at **H2** installed), and cycle the unit's power.
2. Open a program if one is not already open.
3. Select the **Compile** command from the **Compile** menu, or press **F3** on your keyboard.
4. If no errors are detected, Dynamic C compiles the program and automatically downloads it into the PK2300's onboard flash memory.
5. Remove power from the PK2300.
6. Remove the **Run/Program** jumper.
7. Reapply power to the PK2300. This action resets the PK2300 in the Run Mode, and the downloaded program begins to run.



The downloaded program begins to run as soon as power is reapplied. Pay close attention to any electronic or mechanical devices connected to the PK2300 that could cause injury or damage.

Your program is now loaded permanently in the PK2300's onboard flash EPROM. Until you load another program, the program runs automatically every time the PK2300 powers up in Run Mode.



The flash EPROM has a rated lifetime of only 100,000 writes (unlimited reads). Do not write the flash EPROM from within a loop. The flash EPROM should be written to only in response to a human request for each write.

Follow these steps to return to Program Mode.

1. Remove power from the PK2300.
2. Reinstall the **Run/Program** jumper on header **H2**. Refer to Figure 4-2 for the jumper location.
3. Reapply power to the PK2300.



Refer to the previous section in this chapter, "Changing the PK2300's Operating Mode," for more detailed information on changing modes.

Using the Digital Inputs/Outputs

This section provides descriptions of the PK2300's digital I/O and information on how to use them.

Protected Digital Inputs

The PK2300 provides up to 16 protected digital inputs. These inputs are designed as logical data inputs, returning either 1 “ON” or 0 “OFF” (Boolean). The inputs accept voltages in the range of from -20 V DC to +24 V DC with a logic threshold of 2.5 V DC. This means that a protected digital input returns a 0 “OFF” if the input voltage is below 2.5 V DC and a 1 “ON” if the input voltage is above 2.5 V DC.

Protected digital inputs can be used with +5 V DC CMOS or TTL compatible hardware drivers and sensors. This compatibility allows your system to interface directly with other electronic hardware such as peripheral controllers, as well as many mechanical switches including contacts on relays.



Refer to Appendix B, “Specifications,” for complete specifications on the PK2300's protected inputs.

The PK2300 also provides two level sensitive interrupts that are wired in parallel with protected inputs IN-06 and IN-07. The level sensitive interrupts are software assignable.



Refer to Chapter 5, “Software Reference,” for details on the level-sensitive interrupts.

How to Read the Inputs

This section provides information on using the Dynamic C software drivers for the PK2300's protected digital inputs.

The following software driver reads the status of a specified protected digital input. A sample program is provided.

- **int eioBrdDI (unsigned chanNum)**

Reads the state of an input channel.

PARAMETER: **chanNum** must be a number ranging from 0 (for IN-01) through 15 (for IN-16).

RETURN VALUE:

- 0 if and only if the input channel reads low.
- 1 if and only if the input channel reads high.



Refer to Chapter 5, "Software Reference," for further details.

The Input Demonstration Program shows how to read the status of a digital input.

Input Demonstration Program

```
#use eziopk23.lib    // Use the PK2300 I/O
                    // library
#define INPUT1      0 // Assign INPUT1 protected
                    // digital input IN-01

main() {
    int I;           // Create integer I
    eioBrdInit(0);  // Initialize the PK2300
    I = eioBrdDI(INPUT1); // Assign integer I the
                    // status of INPUT1
    printf("IN-01 Status: %d\n", I);
                    // Print status of IN-01
                    // to the STDIO window
A → I = eioBrdDI(INPUT1); // Assign integer I the
                    // status of INPUT1
    printf("IN-01 Status: %d\n", I);
                    // Print status of IN-01
                    // to the STDIO window
B → }               // End of program
```


The following steps explain how to use the Input Demonstration Program.

1. Key in the Input Demonstration Program.
2. Save the program by choosing **Save** from the **File** menu.
3. Compile the program by pressing **F3** or by choosing **Compile** from the **Compile** menu.
4. Connect a wire from J1 pin 1 (+DC) on the PK2300 to J4 pin 2 (IN-01). This connection provides a logic level of 1 at IN-01.



The PK2300's protected inputs are factory-configured to be "pulled up." The inputs then return a logic level of 1 when not connected to ground.

5. Using the Dynamic C command **F8** (run/step over), single-step through the program to the line marked **A**. At this point, the Dynamic C **STDIO** window opens and displays the status of IN-01. The status of IN-01 should be 1. If the status is not 1 or the **STDIO** window did not open, verify your hardware connections and the program.



Refer to your Dynamic C manuals for more information about single-stepping.

6. Remove the wire from J1 pin 1 (+DC) and connect that end to J4 pin 1 (GND). Leave the other end connected to J4 pin 2 (IN-01).
7. Using the Dynamic C command **F8** (run/step over), single-step your program to line **B**. At this point, the Dynamic C **STDIO** window opens and displays the status of IN-01. The status of IN-01 should be 0. If the status is not 0 or the **STDIO** window did not open, verify your hardware connections and the program.
8. Continue to press **F8** until the program terminates.
9. Repeat steps 4 through 7 as necessary.

High-Current Outputs

The PK2300 provides up to eight high-current driver outputs. These outputs use open-collector Darlington transistors to switch an external device ON and OFF. The high-current driver outputs can provide up to 500 mA, which is useful for driving small loads, including inductive loads such as relays.

High-current driver outputs are factory configured as “sinking” drivers, and can optionally be configured as “sourcing” drivers.



Refer to Appendix B, “Specifications,” for complete specifications on high-current drivers. Appendix D, “Sinking and Sourcing Drivers,” provides more information on sinking and sourcing.



High-current driver outputs can be configured for pulse-width modulation. Refer to the “PWM Outputs” section in this chapter for further details.

How to Use the Outputs

This section provides information on the Dynamic C software drivers for the PK2300’s high-current driver outputs.



Refer to Chapter 5, “Software Reference,” for further details.

The following software function turns a specified high-current driver ON or OFF.

```
• int eioBrdDO( unsigned chanNum, char state )
```

Changes the state of an output channel.

PARAMETERS: **chanNum** must range from 0 (for OUT-01) through 7 (for OUT-08).

state is 0 if and only if the corresponding output is to be disabled “OFF”; 1 if and only if the corresponding output is to be enabled “ON.”

RETURN VALUE:

- 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.

The following demonstration program shows how to turn on and turn off a high-current output.

Output Demonstration Program

```
#use eziopk23.lib // Use the PK2300 I/O
// library
#define OUTPUT1 0 // Assign OUTPUT1 high-
// current output OUT-01
#define ON 1 // Assign ON the logic
// value 1
#define OFF 0 // Assign OFF the logic
// value 0

main() {
    eioBrdInit(0); // initialize the PK2300
    eioBrdDO (OUTPUT1,OFF); // Turn-off OUT-01
A → eioBrdDO (OUTPUT1,ON); // Turn-on OUT-01
B → eioBrdDO (OUTPUT1,OFF); // Turn-off OUT-01
C → } // End of program
```

The following steps explain how to use the Output Demonstration Program.

1. Key in the Output Demonstration Program.
2. Save the program by choosing **Save** from the file menu.
3. Compile the program by pressing **F3**, or by choosing **Compile** from the **Compile** menu.
4. Connect a wire between J1 pin 1 (+DC) and J1 pin 2 (K). Connect a device such as a voltmeter between J1 pin 4 (OUT-01) and J1 pin 2 (K), as shown in Figure 4-3. This allows you to monitor the status of the driver.

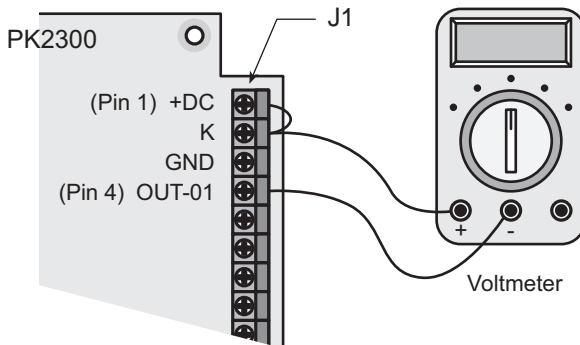


Figure 4-3. Example of Monitoring OUT-01 Status



Use caution when connecting any device to a high-current driver, making sure that you do not exceed the high-current driver's voltage and current limitations. Refer to Appendix B, "Specifications," for complete specifications on high-current drivers.

- Using the Dynamic C command **F8** (run/step over), single-step through your program to the line marked **A**. At this point, your device should be "OFF." If your device is not OFF, verify your hardware connections and the program.



Refer to your Dynamic C manuals for more information about single-stepping.

- Using the Dynamic C command **F8** (run/step over), single-step through your program to the line marked **B**. At this point, your device should be "ON." If your device is not ON, verify your hardware connections and the program.
- Using the Dynamic C command **F8** (run/step over), single-step through your program to the line marked **C**. At this point, your device should be "OFF" again. If your device is not OFF, verify your connections and the program.
- Continue to press **F8** until the program terminates.
- Repeat steps 5 through 8 as necessary.

Resistance Measurement Input (RMI)

The PK2300 provides a single-channel Resistance Measurement Input (RMI). This input allows the PK2300 to measure a single two-terminal resistance sensor with a resistance of 0 Ω to 275 k Ω at a frequency of approximately six conversions per second.



The resistive device being measured must be connected between J4 pins 9 (RMI+) and 10 (RMI-). Configure jumper block H4 as shown in Figure 4-4 to enable the RMI function.

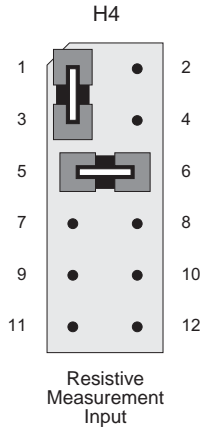


Figure 4-4. Header H4 Settings for RMI

A thermistor or potentiometer can be quickly interfaced to the PK2300. A small set of sample applications is listed below.

- Dependable temperature measurements using thermistors.
- Moving table position using linear potentiometer with mechanical cable attachment.
- Continuous control wheel for a high-quality man/machine interface.
- Rotary switch with different resistors for each position, representing different operating modes.
- Position locator using R-2R ladder and multiple contact switches.
- Light level measurement using a photo-resistor.
- Accelerometers based on linear potentiometers.
- Linear position sensors.

How to Use the Resistance Measurement Input

This section provides information on the Dynamic C software drivers for the PK2300's RMI.



Refer to Chapter 5, “Software Reference,” for further details.

The following software function returns a value based on the resistance from the RMI.

- **float eioBrdAI(unsigned chanNum)**

Reads the Resistance Measurement Input.

PARAMETER: **chanNum** must be either **0** or **1**.

- **chanNum** is 0 if you want to read the calibrated and converted value (using the calibration constants stored in the PK2300).
- **chanNum** is 1 if you want to read the input's “raw” value. This function returns a number between 0 and 65535 when the raw value is read (i.e., channel 1).
- If **chanNum** is out of range, the global variable **eioErrorCode** has the **EIO_NODEV** flag bit set. In this case, the expression **(eioErrorCode & EIO_NODEV)** evaluates to non-zero, indicating an error has occurred.

The following program demonstrates how to read the resistance measurement input. Make sure the PK2300 is set up for RMI.

RMI Demonstration Program

```
#use eziopk23.lib // Use the PK2300 I/O
// library
#define RINPUT 1 // Assign RINPUT RMI
// Channel 1

main() {
    float f; // Create floating point
// variable f
    eioBrdInit(0); // Initialize PK2300
    _eioSetupAI1st(); // Initialize RMI Channel
    while(1){
        hitwd
        f = eioBrdAI(RINPUT); // Assign f the value
// of RINPUT
        printf("Raw value = %f \n", f);
// Print the value of the
// RMI channel to the
// STDIO window
A → f = eioBrdAI(RINPUT); // Assign f the value
// of RINPUT
        printf("Raw value = %f \n", f);
// Print the value of the
// RMI channel to the
// STDIO window
B → }
    } // End of program
```

The following steps explain how to use the RMI demonstration program.

1. Key in the RMI Demonstration Program.
2. Save the program by choosing **Save** from the **File** menu.
3. Compile the program by pressing **F3**, or by choosing **Compile** from the **Compile** menu.
4. Connect a resistor with a value ranging from 0 Ω to 275 k Ω between J4 pin 9 (RMI+) and J4 pin 10 (RMI-).
5. Using the Dynamic C command **F8** (run/step over), single-step through your program to the line marked **A**. At this point, the Dynamic C **STDIO** window opens and displays the raw value of the RMI input. The value should be between 0 and 65535. If you do not see a value, verify your hardware connections and the program.



Refer to your Dynamic C manuals for more information about single-stepping.

6. Remove the resistor installed in step 4, and replace it with one of a *different value* between 0 Ω and 275 k Ω .
7. Using the Dynamic C command **F8** (run/step over), single-step through your program to the line marked **B**. At this point, the Dynamic C **STDIO** window opens and displays the raw RMI input value of the new resistor.

If you do not see a change in RMI values, verify your connections and the program.
8. Continue to press **F8** until the program terminates.
9. Repeat steps 4 through 8 as necessary.



If there is no resistor across the RMI terminals, the RMI value printed in the **STDIO** window should be 65535.

RMI Theory of Operation

The PK2300 provides a current-limited regulated DC voltage at RMI+ (J4 pin 9). Current travels from RMI+ through the resistance sensor and returns through RMI- (J4 pin 10) to the PK2300. This return current changes in proportion to the resistance of the sensor. The return current charges a high quality capacitor on the PK2300. As the capacitor charges, its voltage rises until it reaches a trip point at two-thirds of the reference voltage. The PK2300 processor is interrupted when the voltage reaches the trip point, and the PK2300 then stores the count from one of its internal counters. The counter is then reset to zero and the capacitor is discharged. This completes one cycle.

In summary, the larger the resistance, the smaller the current sourced from the reference power supply to the capacitor. The smaller this current, the longer it takes to charge this capacitor. The longer the time to charge the capacitor, the longer the count interval (greater accumulated count). *The count value is directly proportional to the resistance.*

Figure 4-5 illustrates the operation of the RMI circuit.

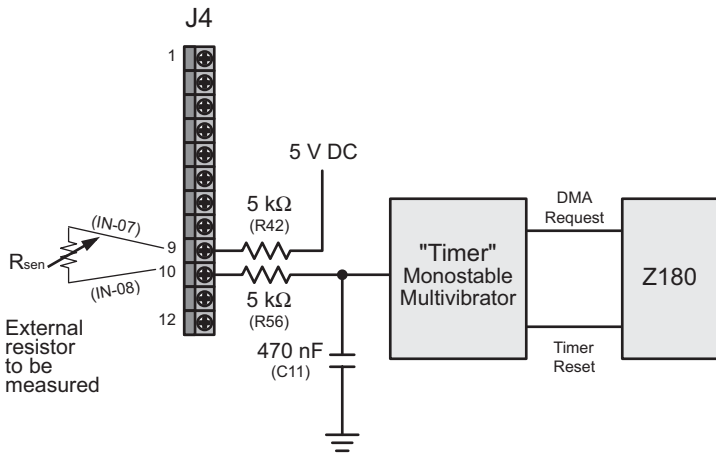


Figure 4-5. PK2300 Resistance Measurement Input Circuit

Serial Communication

The PK2300 provides two serial communication ports that can be configured as RS-232 and/or RS-485.



Table 3-3 in Chapter 3, “Input/Output Configuration,” provides the configurations for the serial channels.

RS-232 Communication

RS-232 is an asynchronous serial communications protocol that is full-duplex (simultaneous bidirectional data transfer). The RS-232 ports and the supplied Dynamic C software allow the PK2300 to communicate with other computers or controllers. The addition of a modem enables remote communication (including remote downloading) using the X-modem protocol. RS-232 software drivers for this protocol can be found in the Dynamic C \SAMPLES\AASC library.



Refer to your Dynamic C manuals for instructions on how to use the RS-232 channels.

Tip

The optional Serial Interface Board 2 leaves both of the RS-232 ports available to the application during software development. A special cable has to be made to access J2 if both RS-232 ports are needed.

RS-232 Connector Pinouts

The 6-pin RJ-12 modular phone jack (**J2**) facilitates all RS-232 connections. Table 4-2 lists the pin assignments for connector **J2**.

Table 4-2. J2 Pin Assignments

J2 Pin No.	Handshaking (Configuration II)	No Handshaking (Configuration I or III)
1	RTS RS-232 (0)	Transmit RS-232 (1)
2	GND	GND
3	Transmit RS-232 (0)	Transmit RS-232 (0)
4	Receive RS-232 (0)	Receive RS-232 (0)
5	CTS RS-232 (0)	Receive RS-232 (1)
6	+5 V regulated	+5 V regulated



Do not pull more than 10 mA from pin 6 of **J2**.

RS-485 Communication Network

The PK2300 can be configured to provide one channel of RS-485 communication. RS-485 is an asynchronous multidrop half-duplex standard that provides multidrop networking for cable lengths up to 1200 m (4000 feet).

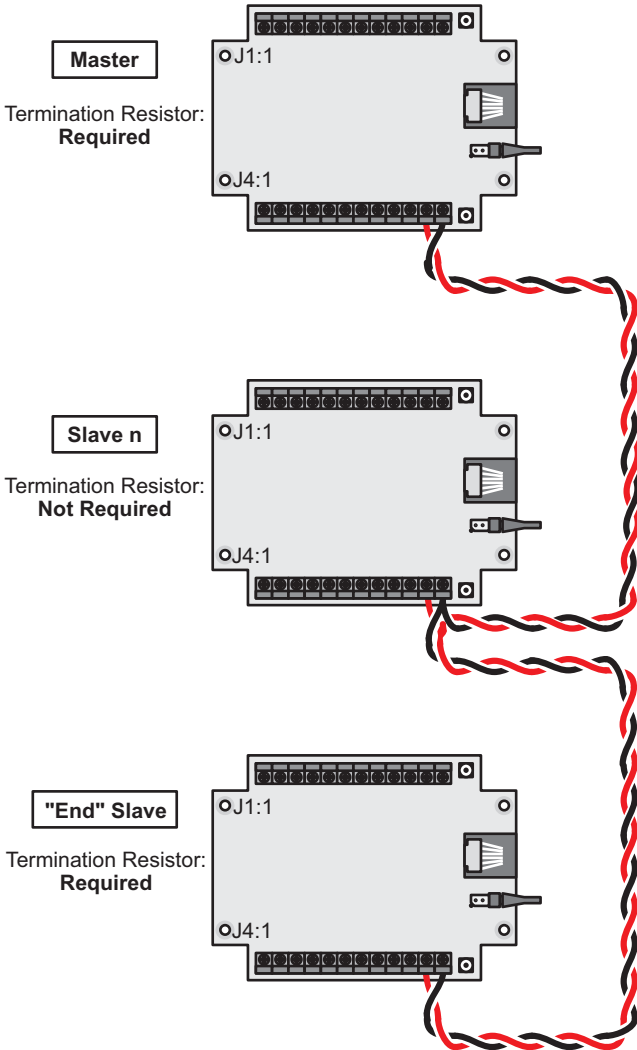


Figure 4-6. Multidrop Network

Dynamic C provides library functions for master-slave two-wire half-duplex RS-485 9th-bit binary communication.

This RS-485 hardware standard supports up to 32 controllers on one network. The software supports one master unit, plus up to 255 slave units (which may consist of any combination of Z-World controllers that support the RS-485 protocol).

The resulting multidrop network, shown in Figure 4-6, can span over a kilometer, facilitating the design of a robust distributive control system.

Follow these steps to configure a multidrop network.

1. Configure the PK2300's J4 pins 11 and 12 for RS-485 communication as outlined in the "Configuring Serial Communication" section in Chapter 3, "Input/Output Configuration."
2. On all networked controllers, connect RS-485+ to RS-485+ and RS-485- to RS-485- using nonstranded, tinned, single twisted-pair wires. Refer to Figure 4-6.



Refer to your Dynamic C manuals for details on master-slave networking.

Termination and Bias Resistors

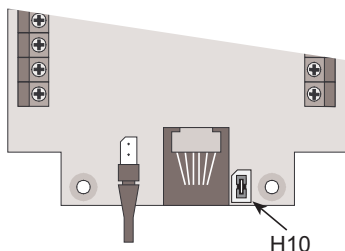
Termination and bias resistors are required in a multidrop network to minimize reflections (echoing) and to keep the network line active when it is in an idle state.

- Bias resistors are required in most cases on the master network controller to keep the network line reliable.
- A termination resistor is required on the master network controller and on the "end" slave (the last controller in the series).

The PK2300 is shipped with the following factory defaults.

- 10 k Ω bias resistors are installed at locations R33 and R5.
- Termination resistor R5 is *not* installed.

A multidrop network must have 120 Ω termination resistors installed on both the master network controller and the “end” slave controller. Header H10 allows a 120 Ω termination resistor to be enabled by leaving the jumper across header H10 connected as shown in Figure 4-7. Remove this jumper from all the other controllers except the two end controllers when using the PK2300s in a multidrop network.



**Figure 4-7. Enabling Termination Resistor
(PK2300 Top View, Cover Removed)**



The RS-485 drivers supplied by Z-World support up to 32 nodes. As additional nodes (over the benchmark quantity of 32) are added to your network, the transmission bandwidth may be reduced.

Contact Z-World Technical Support at (530)757-3737 for assistance with large-scale network design.



The PK2300 is available in quantity with a 0 Ω surface-mounted resistor installed in lieu of header H10 to enable the RS-485 termination resistor permanently. For ordering information, or for more details, call your Z-World Sales Representative at (530) 757-3737.

Additional Features

This section provides information on the PK2300's additional features, including PWM Outputs, User-Programmable LEDs, the Real-Time Clock (RTC), and the Power Supervisor.

PWM Outputs

The PK2300 can produce fixed-frequency, pulse-width modulated (PWM) signals from seven of its high-current outputs simultaneously.

The supplied software provides two levels of support. The first level provides easy-to-use fixed PWM functions for only four of the outputs. The periods of the PWM signals are fixed at 13.3 ms (75 Hz), with a resolution of 256 divisions per period (8-bit resolution). Using the supplied software, the generation of PWM signals consumes about 8% of controller's processing power. The second PWM support level allows you to create custom PWM functions for seven of the outputs.



Serial communication baud rates may be affected when PWM functions are used because PWM functions place a further demand on the microprocessor's resources. Serial data rates for Serial Port 1 become fixed at 4800 bps, and you must reset the baud rates for Dynamic C to 4800 bps.

Contact Z-World Technical Support at (530)757-3737 for further assistance with PWM functions.



Refer to the "Advanced Input/Output Programming" section in Chapter 5, "Software Reference," for advanced PWM programming information.

How to Use the PWM Feature

The PK2300 can produce fixed-frequency, fixed-phase, variable-duty-cycle square waves from up to seven of its outputs. This section first presents a simple, easy-to-use PWM function that drives only four of the PK2300's outputs. A more complex set of functions requiring a more in-depth understanding of DMA and PWM generation is presented later.

Figure 4-8 and Figure 4-9 provide PWM transition and DMA timing diagrams.

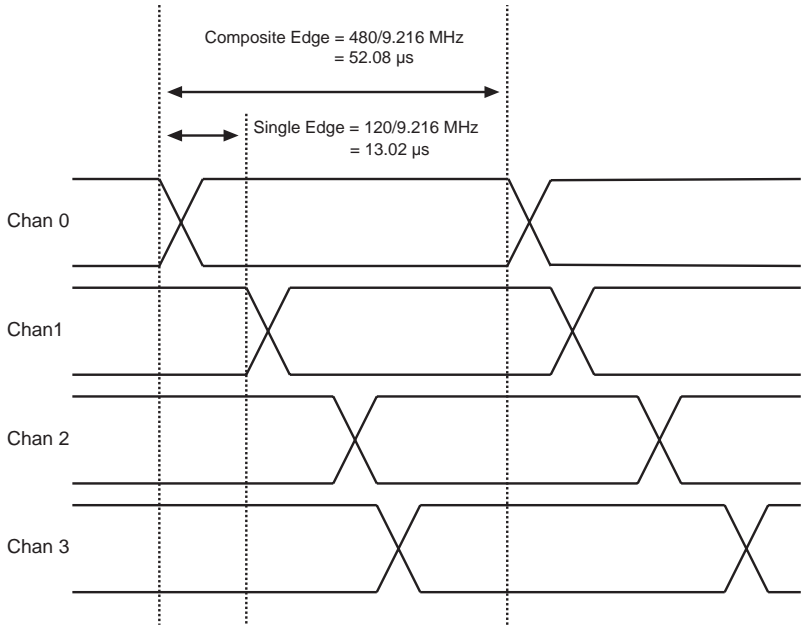


Figure 4-8. Transition Timing

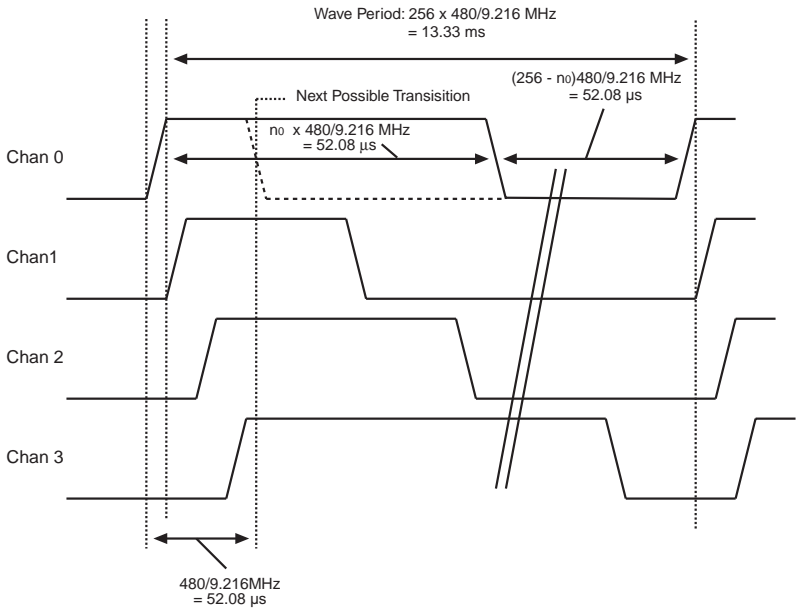


Figure 4-9. DMA Timing

The function produces PWM square waves on outputs OUT-01, OUT-02, OUT-03, and OUT-04. Notice that each square wave's period is exactly 1024 "divisions." One division equals 120 clock cycles ($120/9.216 \text{ MHz} = 13.02 \mu\text{s}$) for the PWM function. Consequently, the period of each square wave is $1024 \times 13.02 \mu\text{s} = 13.33 \text{ ms}$.

Notice also that the square waves are displaced slightly from each other in phase. That is, OUT-02 starts and ends one division after OUT-01, OUT-03 starts one division after OUT-02, and OUT-04 starts one division after OUT-03. As a result, although the period of each wave is 1024 divisions, a change to one particular channel is possibly only every four divisions. Therefore, the resolution of the transition edge in the wave is $1/256$.

- **int eioBrdAO(unsigned chanNum, unsigned state)**

Specifies the duty cycle for a particular output channel.

PARAMETERS: **chanNum** is a number ranging from 0 (for OUT-01) to 3 (for OUT-04).

state is a placeholder for a number ranging from 0 (to turn off the channel) to 256 (to turn on the channel, 100% duty cycle). The duty cycle is **state**/256 (e.g., 128 for 50% duty cycle, 64 for 25% duty cycle).



While the PWM functions are simple to use, you must be aware of their side effects. The functions use the Z180's built-in DMA hardware. Using the DMA-driven PWM functions limits the communication speed of the Z180's Serial Port 1 to 4800 bps. The Z180 also effectively runs at least 8% slower.

Your application must call **_eioBrdAORf** at least every 25 ms to capture the DMA counter.

Contact Z-World Technical Support at (530)757-3737 for further assistance.

The following sample program in the Dynamic C **SAMPLES\PK23XX** subdirectory ramps the PWM output from OUT-01 up and down.

A01.C

```
#use eziopk23.lib // PK2300 specific defs
#define OUT_CHAN 0 // define output channel
main() {
    auto unsigned dutyCycle;
    auto int sign;
    eioBrdInit(0); // initialize general I/O
    _eioSetupAO1st(); // initialize PWM
    dutyCycle = 0; // duty cycle starts at 0
    sign = 1; // ramp up
    while (1) { // do this forever
        eioBrdAO(OUT_CHAN,dutyCycle); // change cycle
        if (_eioBrdAORf()==-1) break; // refresh OK?
        if (dutyCycle == 256) sign = -1; // reverse
        else if (dutyCycle == 0) sign = 1; // reverse
        dutyCycle = dutyCycle + sign; // ramp
        hitwd(); // hit watchdog
    }
    printf("AO refresh failed\n");
}
```

User-Programmable LEDs

The PK2300 provides two clearly visible surface-mounted LEDs, the Run LED (D1) and the User LED (D2). The user's program can control both LEDs. During program development, the User LED indicates the various operating modes, illuminating steadily to indicate power-on and that the Z-World factory default BIOS is functioning.

The following software drivers can be used to turn the LEDs ON or OFF.

- LED **D1** (User)
 - ~ `output(0x4141, 1)` - turns the LED ON
 - ~ `output(0x4141, 0)` - turns the LED OFF
- LED **D2** (Run)
 - ~ `output(0x4142, 1)` - turns the LED ON
 - ~ `output(0x4142, 0)` - turns the LED OFF



Refer to the sample program `PK23FLASH.C` in Chapter 2, "Getting Started," for an example.

Memory

The PK2300 comes with 128K of nonvolatile flash EPROM and 32K of battery-backed surface-mounted SRAM. The flash EPROM retains its contents even if power is removed, allowing safe storage of programs, data, and parameters.



Refer to your Dynamic C manuals for complete specifications on reading and writing to the PK2300's memory.



Writing to flash EPROM takes approximately 20 ms, and receives priority over all other functions within your application. Pay special attention to interrupts and PWM functions when writing to flash EPROM.

Real-Time Clock (RTC)

The PK2300's RTC maintains the current time and date, accounts for the number of days in differing months, and accounts for leap years. A backup battery keeps the RTC running when power is removed.



The RTC functions cannot be used with the PK2310.



Do not use the RTC to create short delays in your application. This could result in loss of accuracy.

The Dynamic C function library `DRIVERS.LIB` provides the RTC functions listed in Table 4-3.

Table 4-3. Real Time Clock Functions

Function	Description
<code>tm_rd</code>	Read time and date values from the RTC.
<code>tm_wr</code>	Write time and date values into the RTC.



Refer to the *Dynamic C Function Reference* manual for a description of the RTC functions and the associated `tm` data structure.



The RTC cannot update during a read cycle. Reading the clock constantly in a tight loop may lead to a loss of accuracy. Event scheduling from the RTC is not recommended.

Power Supervisor

The PK2300 provides a power supervisor IC that controls the power-on reset function. This function holds the reset line low until VCC rises above a threshold of ~ 4.75 V.

When VCC falls below this threshold, the supervisor disables the SRAM to prevent writing spurious data. The supervisor also switches the SRAM to battery power when VCC falls below the threshold voltage to preserve the SRAM's data until power is restored.

The supervisor has a watchdog timer that guards against system or software faults. If the application's software does not reset the timer at least once every second, the PK2300's microprocessor will reset. The Dynamic C function `hitwd` resets the supervisor's watchdog timer.

Tip

Use the function `hitwd` in any program loop that may take more than one second to execute.



Refer to your Dynamic C manuals for further information on `hitwd`.

In addition, the supervisor generates a nonmaskable interrupt (NMI) when unregulated DC input (normally 9 V to 12 V DC) falls below 8.02 V DC; this allows the PK2300's microprocessor time to execute a safe shutdown. This circuitry, along with the appropriate interrupt service routine and external power supply capacitance, will allow the PK2300 to recover from brownouts.

The PK2300 can distinguish between a power-on reset and a watchdog reset.



Refer to the "Power Failure Sequence of Events" section in Appendix E, "Power Management," for further details.



Refer to your Dynamic C manuals for further information on power failure interrupt service routines, or contact Z-World Technical Support at (530)757-3737.

Resetting the Processor

To reset the processor, temporarily short pins 5 and 6 on header H1 as shown in Figure 4-10.

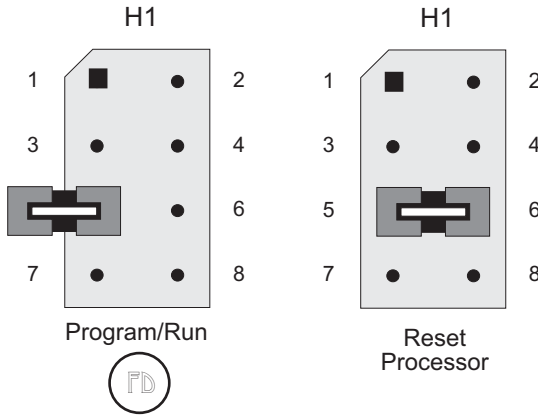


Figure 4-10. Header H1 Configurations to Reset Processor

The PK2300 normally operates with the jumper attached as shown in Figure 4-10. Leave the jumper attached, but not connected, to ensure it is available when needed. The processor may be reset either as shown in Figure 4-10, or by removing power from the PK2300 for approximately 10 ms.

PK2300 Subsystems

Figure 4-11 illustrates the PK2300's subsystems.

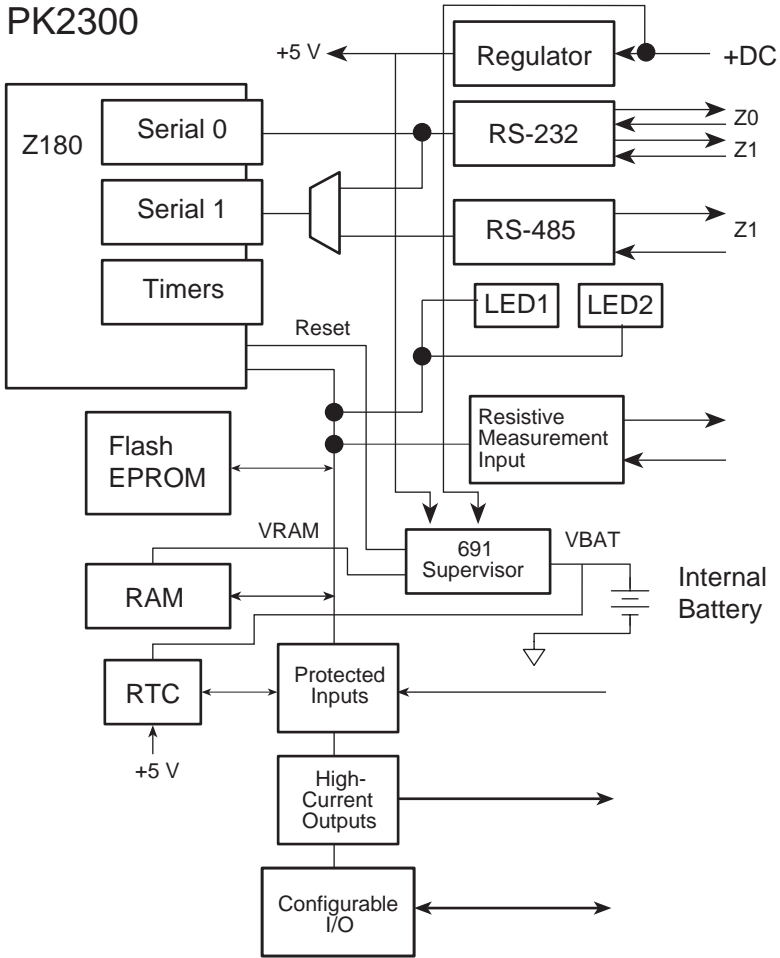


Figure 4-11. PK2300 Subsystems



CHAPTER 5: SOFTWARE REFERENCE

Chapter 5 describes the Dynamic C functions that initialize the PK2300 and perform input/output operations. The following major sections are included.

- Supplied Software
- Digital Inputs/Outputs
- Level-Sensitive Interrupts
- Resistance Measurement Input
- PWM Outputs
- Additional Software Feature Reference
- Advanced Input/Output Programming
- Digital Input Addressing Detail
- Digital Output Addressing Detail
- LED Addressing Detail
- RS-485 Driver IC Addressing Detail
- Resistance Measurement Input Addressing Detail
- PWM Addressing Detail
- PWM Advanced Programming Functions

Supplied Software

A series of software drivers for controlling the PK2300's inputs/outputs is provided with Dynamic C 5.x. In order to use the supplied routines defined in this chapter, use the **EZIOPK23.LIB** Dynamic C library. An application program can use the library by including the following line at the top of the program.

```
#use eziopk23.lib
```

The following functions are located in the **EZIOPK23.LIB** library.

- **void eioBrdInit(int param)**

Initializes the software.

Call this function in the initialization section of your program before using any other functions. Always pass 0 for **param**. This function:

- Does not call **_eioSetupAI1st**. If you use the Resistance Measurement Input, you must call **_eioSetupAI1st** separately.
- Does not call **_eioSetupAO1st**. If you use the DMA-driven PWM output, you must call **_eioSetupAO1st** separately.

- **long int eioErrorCode**

Represents a global bit-mapped register with flags that reflect error occurrences.

This register is initially set to 0 by **eioBrdInit**. If the application tries to access an invalid channel, the flag **EIO_NODEV** (bit 1 flag) is set in this register.

Digital Inputs/Outputs

The following digital I/O functions are located in the **EZIOPK23.LIB** library.

- **int eioBrdDI(unsigned chanNum)**

Reads the state of an input channel.

PARAMETER: **chanNum** must be a number ranging from 0 (for IN-01) through 15 (for IN-16).

RETURN VALUE:

- 0 if and only if the input channel reads low.
- 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 15).

Table 5-1 lists the software input channel assignments.

Table 5-1. Software Input Channel Assignments

Input	Channel Assignment
IN-01	0
IN-02	1
IN-03	2
IN-04	3
IN-05	4
IN-06	5
IN-07	6
IN-08	7
IN-09	8
IN-10	9
IN-11	10
IN-12	11
IN-13	12
IN-14	13
IN-15	14
IN-16	15

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

Changes the state of an output channel.

PARAMETERS: **chanNum** must range from 0 (for OUT-01) through 7 (for OUT-08).

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:

- 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.

Table 5-2 lists the software output channel assignments.

Table 5-2. Software Output Channel Assignments

Output	Channel Assignment
OUT-01	0
OUT-02	1
OUT-03	2
OUT-04	3
OUT-05	4
OUT-06	5
OUT-07	6
OUT-08	7

The program `DIO1.C` in the Dynamic C `SAMPLES\PK23XX` subdirectory enables the PK2300 to function as a relay. If digital input IN-01 (input channel 0) is grounded, the digital output OUT-01 (output channel 0) is disabled (OFF). Otherwise, the digital output is enabled.

DIO1.C

```
#use ezio.lib           // general I/O definitions
#use eziopk23.lib      // pk2300 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
    }
}
```



Refer to the Dynamic C `SAMPLES\PK23XX` subdirectory for additional sample programs.

Level-Sensitive Interrupts

The PK2300 can generate two level-sensitive processor interrupts under software control. The interrupts respond to a logic level “0” or OFF.

Protected digital inputs IN-06 and IN-07 are directly connected to the /INT0 and /INT1 interrupt lines of the Z180 processor as shown in Figure 5-1.

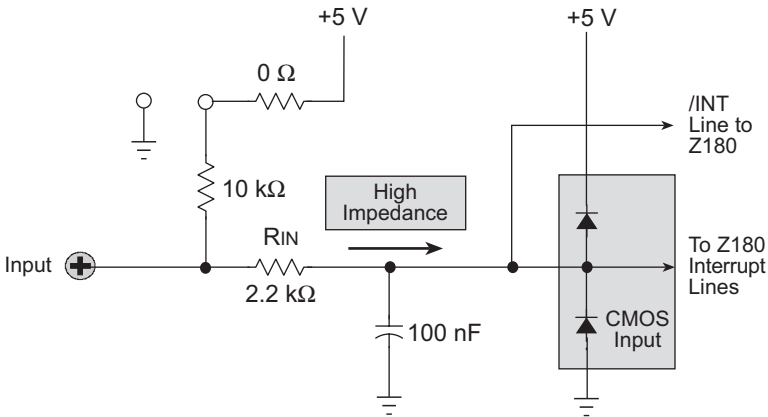


Figure 5-1. IN-06 and IN-07 Connection for /INT0

By default, the interrupt 0 and 1 are disabled “OFF,” allowing inputs IN-06 and IN-07 to be used as standard protected digital inputs.

When level-sensitive interrupts are required, interrupts 0 and 1 can be enabled “ON” or disabled “OFF” by including the following commands in your code.

- Enable “ON” Interrupt 0
~ `outport(ITC, inport(ITC) | 1)`
- Enable “ON” Interrupt 1
~ `outport(ITC, inport(ITC) | 2)`
- Disabled “OFF” Interrupt 0
~ `outport(ITC, inport(ITC) & 0xfe)`
- Disabled “OFF” Interrupt 1
~ `outport(ITC, inport(ITC) & 0xfd)`

Interrupt Service Routines (ISR)

ISRs are packets of code that the processor jumps to and executes when it receives an interrupt request from /INT0 or /INT1.



Refer to your Dynamic C manuals for instructions on writing ISRs.



When interrupts 0 and 1 are enabled “ON,” a logic level of “0” on inputs IN-06 and IN-07 halts the processor and performs the specified ISRs. All other activities are stopped until the ISR is complete. Enabling interrupts 0 or 1 can severely affect the interrupt latency of other types of interrupts, such as those generated by PRTs, the DMA, and UARTs. Contact Z-World Technical Support at (530)757-3737 for assistance.



Interrupts 0 and 1 can only be used when the PK2300’s inputs are in the factory-default “pulled up” configuration. Enabling the interrupts when the inputs are “pulled down” causes the PK2300 to malfunction.



Refer to Zilog’s ***Z80180/Z180 MPU User’s Manual*** for complete details on Z180 interrupts.

Resistance Measurement Input

The following RMI functions are located in the Dynamic C `EZIOPK23.LIB` library.



See Chapter 3, “Input/Output Configuration,” for directions on configuring the PK2300’s jumpers to enable Resistance Measurement Input.

- **void _eioSetupAI1st()**

Initializes the hardware for resistance-sensor measurements.

This function must be called before calling `eioBrdAI`, or the return value of `eioBrdAI` will be undefined. The resistance-sensor measurement routines require exclusive use of the Z180’s programmable timer PRT0.

- **int eioBrdACalib(int chanNum, int d1, int d2, float f1, float f2)**

Sets up the calibration constants needed by `eioBrdAI` when called with `chanNum` equal to 0.

The function computes the calibration coefficients and stores them in reserved locations in nonvolatile memory. The function `eioBrdInit` loads these constants from nonvolatile memory.

To calibrate the resistance measurement input, apply two known precision resistance’s to the resistance measurement input.

PARAMETERS: `chanNum` is always 0 because the PK2300 has only one Resistance Measurement Input.

`d1` is the raw, digital reading corresponding to the applied analog resistance `f1`.

`d2` is the raw, digital reading corresponding to the applied analog resistance `f2`.

- **float eioBrdAI(unsigned chanNum)**

Reads the resistance measurement input.

PARAMETER: **chanNum** must be either 0 or 1.

- **chanNum** is 0 if you want to read the calibrated and converted value (using the calibration constants stored in the PK2300).
- **chanNum** is 1 if you want to read the input's "raw" value. This function returns a number between 0 and 65535 when the raw value is read (i.e., Channel 1).
- If **chanNum** is out of range, the global variable **eioErrorCode** has the **EIO_NODEV** flag bit set. In this case, the expression **(eioErrorCode & EIO_NODEV)** evaluates to non-zero, indicating an error has happened.

The following sample program from the Dynamic C **SAMPLES\PK23XX** subdirectory prints the "raw" value of the resistance measurement.

AI1.C

```
#use eziopk23.lib // pk2300 specific defns
main() {
    eioBrdInit(0); // initialize general I/O
    _eioSetupAI1st(); // initialize RMI
    while (1) { // do this forever
        printf("reading %f\n",eioBrdAI(1));
        hitwd(); // hit the watchdog
    }
}
```



Refer to the Dynamic C **SAMPLES\PK23XX** subdirectory for additional sample programs.

PWM Outputs

The following PWM functions are located in the Dynamic C **EZIOPK23.LIB** library.

- **void _eioSetupAO1st()**
Initializes the PWM hardware.
_eioSetupAO1st must be called before using **eioBrdAO**.
- **int eioBrdAO(unsigned chanNum, unsigned state)**
Specifies the duty cycle for a particular output channel.
PARAMETERS: **chanNum** is a number ranging from 0 (for OUT-01) to 3 (for OUT-04).
state is a placeholder for a number ranging from 0 (to turn off the channel) to 256 (to turn on the channel, 100% duty cycle). The duty cycle is **state/256** (e.g., 128 for 50% duty cycle, 64 for 25% duty cycle).
- **int _eioBrdAORf()**
Refreshes the DMA counter and address pointer.
Your program must call **_eioBrdAORf** every 25 ms (or more frequently) after **_eioSetupAO1st** is called.
RETURN VALUE: The function returns -1 if the DMA count is zero (PWM has stopped) and returns 0 otherwise. If the function returns -1, it means the driver is either not initialized (by calling **_eioSetupAO1st**), or **_eioBrdAORf** is called less frequently than every 25 ms.



While the PWM functions are simple to use, you must be aware of their side effects. The functions use the Z180's built-in DMA hardware. Using the DMA-driven PWM functions limits the communication speed of the Z180's Serial Port 1 to 4800 bps. The Z180 also effectively runs at least 8% slower.

Your application must call **_eioBrdAORf** at least every 25 ms to refresh the drivers' period.

Contact Z-World Technical Support at (530)757-3737 for further assistance.

The following sample program in the Dynamic C **SAMPLES\PK23XX** subdirectory ramps the PWM output from OUT-01 up and down.

A01.C

```
#use eziopk23.lib // PK2300 specific defs
#define OUT_CHAN 0 // define output channel
main() {
    auto unsigned dutyCycle;
    auto int sign;
    eioBrdInit(0); // initialize general I/O
    _eioSetupA01st(); // initialize PWM
    dutyCycle = 0; // duty cycle starts at 0
    sign = 1; // ramp up
    while (1) { // do this forever
        eioBrdAO(OUT_CHAN,dutyCycle); // change cycle
        if (_eioBrdaORf()==-1) break; // refresh OK?
        if (dutyCycle == 256) sign = -1; // reverse
        else if (dutyCycle == 0) sign = 1; // reverse
        dutyCycle = dutyCycle + sign; // ramp
        hitwd(); // hit watchdog
    }
    printf("AO refresh failed\n");
}
```



Refer to the Dynamic C **SAMPLES\PK23XX** subdirectory for additional sample programs.

Additional Software Feature References



- For **Real-Time Clock** information, refer to descriptions of functions `tm_rd` and `tm_wr` in your Dynamic C manuals.
- For **Communication Ports** information, refer to descriptions of the **AASC** libraries in your Dynamic C manuals. For RS-485 driver switching, refer to descriptions of `on_485` and `off_485` in your Dynamic C manuals.
- For **Watchdog** information, refer to descriptions of the function `hitwd` in your Dynamic C manuals.
- For **EEPROM** information, refer to descriptions of the functions `ee_rd` and `ee_wr` in your Dynamic C manuals.
- For **Power Fail Flag** information, refer to the descriptions of the function `_sysIsPwrFail` and `sysIsPwrFail` in your Dynamic C manuals.
- For **Resetting the Board** information, refer to descriptions of the functions `sysForceSupRst`, `sysIsSuperReset`, `_sysIsSuperReset`, `sysForceReset`, `_sysIsWDTO`, and `sysIsWDTO` in your Dynamic C manuals.

Advanced Input/Output Programming

This section is intended for application developers who need to optimize their code.



The topics discussed in this section require a detailed understanding of the Z180 and its peripheral architecture.

Z180 technical manuals are available. For details, contact your Z-World representative at (530)757-3737.

Table 5-3 provides the concise input/output map of the PK2300 devices. The following sections provide details on using the devices.

Table 5-3. PK2300 Device I/O Map

I/O Address	Description	Comments
0x40c0 (read)	/TRIGGER of 555	Read to start charging and turns /OUTPUT of 555 high
0x4140 (write)	RS-485 Driver	Bit 0 indicates on/off, 1 to turn on, 0 to turn off
0x4141 (write)	LED D1	Same as above
0x4142 (write)	LED D2	Same as above
0x4143 (write)	CMOS OUT-09	Same as above
0x4144 (write)	CMOS OUT-10	Same as above
0x4145 (write)	CMOS OUT-11	Same as above
0x4146 (write)	CMOS OUT-12	Same as above
0x4147 (write)	OUT-08	Same as above
0x4160 (read)	IN-01 through IN-08	Bit x for state of IN_{x+1}
0x4161 (read)	IN-09 through IN-16	Bit x for state of IN_{x+9}
0x4160 (write)	OUT-01 through OUT-07	Bit 0, 1 and 2 indicates which high-current driver, bit 7 indicates on/off, 1 to turn-on, 0 to turn-off
0xa000 (read)	Power fail	Bit 0 indicates whether power is failing, 1 if power is failing, 0 if not

Digital Input Addressing Detail

Read I/O address 0x4160 for the states of inputs IN-01 through IN-08. In this I/O reading, bit 0 corresponds to IN-01, and bit 7 corresponds to IN-08. Read I/O address 0x4161 for states of inputs IN-09 through IN-16. In this I/O reading, bit 0 corresponds to IN-09 while bit 7 corresponds to IN-16. Table 5-4 lists the address structure of the digital inputs.

Table 5-4. Digital Input States

0x4160							
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
IN-08	IN-07	IN-06	IN-05	IN-04	IN-03	IN-02	IN-01
0x4161							
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
IN-16	IN-15	IN-14	IN-13	IN-12	IN-11	IN-10	IN-09

Table 5-5 provides the Dynamic C commands to read each individual input.

Table 5-5. Digital Input States

Channel	Function
IN-01	IBIT (0x4160,0)
IN-02	IBIT (0x4160,1)
IN-03	IBIT (0x4160,2)
IN-04	IBIT (0x4160,3)
IN-05	IBIT (0x4160,4)
IN-06	IBIT (0x4160,5)
IN-07	IBIT (0x4160,6)
IN-08	IBIT (0x4160,7)
IN-09	IBIT (0x4161,0)
IN-10	IBIT (0x4161,1)
IN-11	IBIT (0x4161,2)
IN-12	IBIT (0x4161,3)
IN-13	IBIT (0x4161,4)
IN-14	IBIT (0x4161,5)
IN-15	IBIT (0x4161,6)
IN-16	IBIT (0x4161,7)

Digital Output Addressing Detail

Write to I/O address 0x4160 to set the output states of OUT-01 through OUT-07. The lowest three bits is a number that specifies one of the 7 channels (0 for OUT-01). The most significant bit indicates whether the designated channel is enabled “ON” (if the bit is a 1) or disabled “OFF” (if the bit is a 0).

Table 5-6. Digital Output States

Channel	ON	OFF
OUT-01	1xxxx000	0xxxx000
OUT-02	1xxxx001	0xxxx001
OUT-03	1xxxx010	0xxxx010
OUT-04	1xxxx011	0xxxx011
OUT-05	1xxxx100	0xxxx100
OUT-06	1xxxx101	0xxxx101
OUT-07	1xxxx110	0xxxx110

In other words, if you want to turn OUT-05 on, the lowest three bits should be binary 100, and the most significant bit should be a 1, making the byte to write (binary) 1xxxx100 (xxxx means the values of these bit does not matter). 0x84 is one of the many variant representations of binary 1xxxx100.

OUT-08 is a special case. OUT-08 resides at the I/O address 0x4147. Write 1 to I/O address 0x4147 to enable OUT-08; write 0 to I/O address 0x4147 to disable OUT-08.

LED Addressing Detail

LED D1 (User) corresponds to the I/O address 0x4141; write 1 to turn on the LED, write 0 to turn off the LED.

- Enable “ON” LED D1 (User)
~ `outport(0x4141, 1)`
- Disable “OFF” LED D1 (User)
~ `outport(0x4141, 0)`

LED D2 corresponds to the I/O address 0x4142, write 1 to turn on the LED, write 0 to turn off the LED.

- Enable “ON” LED D2 (Run)
~ `outport(0x4142, 1)`
- Disable “OFF” LED D2 (Run)
~ `outport(0x4142, 0)`

RS-485 Driver IC Addressing Detail

The RS-485 driver IC corresponds to the I/O address 0x4140. Write 1 to enable the RS-485 driver IC, write 0 to disable the RS-485 driver IC.

- Enable “ON” RS-485 driver
~ `outport(0x4140, 1)`
- Disable “OFF” RS-485 driver
~ `outport(0x4140, 0)`



See Chapter 3, “Input/Output Configuration,” for directions on setting up the PK2300’s jumpers to enable the RS-485 channels inputs/outputs.

Resistance Measurement Input Addressing Detail

The PK2300 uses a 555 timer IC to measure the resistance of an external resistance sensor.



See Chapter 3, “Input/Output Configuration,” for directions on setting up the PK2300’s jumpers to enable the Resistance Measurement Input.

The 555-based resistance measurement input’s sequence of operations is listed below.

1. Start a measurement cycle by triggering the 555 timer IC and enabling your “timing mechanism.”
2. Wait for the PK2300’s timing capacitor (C10 and C11), if installed, to charge to a threshold, causing a transition at the output of the 555 timer IC, ending the measurement cycle.
3. Convert the count accumulated by the timing mechanism during the measurement cycle to the appropriate sensor reading using calibration constants derived from test readings.

The software supporting **eioBrdAI** (interface described earlier in this chapter’s simple-function section) uses the Z180’s on-chip programmable timer PRT0 as a timing mechanism. The Z180’s on-chip DMA1 is used as a capturing mechanism to stop and record the time elapsed. The steps involved are listed below.

1. The software sets up the Z180’s on-chip DMA1 controller to transfer the least-significant byte of the counter PRT0 to a RAM location. In addition, DMA1 is set up to interrupt the processor when the byte is transferred.
2. The software also sets up the PRT0 counter to have the maximum reload value of 0xffff (T). PRT0 is set up to cause an interrupt when the counter reaches 0.
3. PRT0 and DMA1 are enabled immediately after the 555 timer IC is triggered (by reading I/O address 0x40c0).

Two interrupts may happen after the 555 timer IC and PRT0 are enabled.

- 4a. If the 555 timer IC output transitions low (because its associated timing capacitor(s), C10 (and C11 if installed), has charged up to the threshold) before PRT0 counts to zero, DMA1 transfers the least significant byte of the PRT0's counter to a RAM location and causes an interrupt. Note that when the least significant byte of PRT0 counter is read by DMA1, the most significant byte of PRT0 counter is also captured and stored internally. The DMA1 interrupt routine then copies the captured most significant byte of the PRT0 counter (locked when the least significant byte was read by DMA1) to another location in RAM.

We now have a snapshot of the PRT0 counter T' . Based on the difference $T - T'$ (recall that T was the reload value, 0xffff), we can compute the actual delay (the PRT0 counter decrements every 20 states) in ticks of PRT0. Each tick of PRT0 is 20 clocks, which is $20/9.216 \text{ MHz} = 2.17 \mu\text{s}$. The delay between trigger and output on the 555 timer is approximately $1.1 \times RC$ seconds, where R is the resistive sensor's resistance (in ohms), and C is the capacitance of C10 (and C11 if installed), 470 nF (940 nF if C11 is also installed).

The DMA1 interrupt routine should also reset PRT0, DMA1 and the 555 timer IC (go back to step 1), letting the measurement cycle repeat itself.

- 4b. If PRT0 counts to zero before the 555 timer IC's output transitions low, the PRT0 interrupt occurs first. This interrupt signifies that the 555 timer IC is taking too long to reach its threshold—in other words, the resistive-sensor's current resistance is out of range. In this case, the PRT0 interrupt routine disables itself, and records T' as 0.

PWM Addressing Detail

The driver of the PWM on the PK2300 is fairly complicated. This is because it uses the clock output from communication port 1 (CKA1) to drive the request line DMA Channel 0 in edge detection mode. The simple interface previously described (`eiobrdAO`) provides PWM support for OUT-01 to OUT-04. If the application requires more PWM channels, or requires specific frequencies or precision, the application engineer may need to make trade-offs.

This section describes how PWM channels are driven, as well as how to customize PWM resource allocation to compromise number of modulated channels, frequency and resolution.

- **Determine number of channels, frequency, resolution.**

A pulse-width modulated waveform has a frequency and a resolution. The frequency states how many times the pattern repeats itself in a second (Hz). The resolution states how many divisions within one waveform can be resolved (distinguished). As a collection, the PWM driver also needs to know the total number of channels pulse-width modulated. We assume all channels have the same frequency and resolution.

The clock output from communication port 1 (CKA1) must have a frequency $f_1 = N_{ch} \times f_w \times R_w$, in which f_1 is the frequency of CKA1, N_{ch} is the number of pulse-width-modulated channels, f_w is the frequency of each channel, and R_w is the resolution in number of divisions per wave.

For example, the driver interface `_eiosetupAO1st` makes the following assumptions: $N_{ch} = 4$, $f_1 = 76800$ Hz, $R_w = 256$. Consequently, $f_w = 76,800$ Hz / $(4 \times 256) = 75$ Hz.

- **Declare storage for the WPB (waveform pattern buffer).**

Memory must be allocated to store the waveform pattern.

- **Set up the waveform.**

The PWM functions use the Z180's built-in DMA mechanism to transfer PWM "edges" from memory to the high-current ports at specific time intervals. Each edge is a byte whose least-significant three bits select one of the high-current outputs, OUT-01 through OUT-07. The most significant bit is a 1 to turn the specified port on (rising PWM "edge") or a 0 to turn the specified port off (falling PWM "edge"). Edges for the channels being pulse-width modulated are then grouped into composite transitions.

Each composite transition is a series of edges, each representing one possible transition for an individual channel. For example, if OUT-01 and OUT-02 are the only pulse-width modulated channels, a composite transition consists of two bytes, one to specify a possible transition for channel OUT-01, one to specify a possible transition for channel OUT-02.

Let us assume the first byte in the composite transition corresponds to OUT-01, and the second byte corresponds to OUT-02.

The composite PWM waveform is a series of composite transitions (CTs) that specify the duty cycle of the PWM channels. For example, if we wish OUT-01 to be at 0.375 duty cycle, OUT-02 to be at 0.75 duty cycle, and a resolution of 8 divisions per cycle, a simple waveform would be as follows.

- CT1: turn on OUT-01, turn-on OUT-02.
- CT2: do nothing.
- CT3: do nothing.
- CT4: turn off OUT-01.
- CT5: do nothing.
- CT6: do nothing.
- CT7: turn off OUT-02.
- CT8: do nothing.

go back to CT1.

Outputting the byte 0x80 turns on OUT-01, 0x00 turns off OUT-01, 0x81 turns on OUT-02, and 0x01 turns off OUT-02. The byte 0x07 is an “noop” and does nothing.

This is because OUT-08 corresponds to I/O address 0x4147, not 0x4160. Consequently, the composite transitions (with noops) can be translated into the following byte sequence to be sent to the I/O address 0x4160 as follows.

- CT1: 0x80, 0x81
- CT2: 0x07, 0x07
- CT3: 0x07, 0x07
- CT4: 0x00, 0x07
- CT5: 0x07, 0x07
- CT6: 0x07, 0x07
- CT7: 0x07, 0x01
- CT8: 0x07, 0x07

go back to CT1

The equivalent byte stream (contents in the waveform pattern buffer) is a repeating pattern of

```
0x80, 0x81, 0x07, 0x07, 0x07, 0x07, 0x00, 0x07,  
0x07, 0x07, 0x07, 0x07, 0x07, 0x01, 0x07, 0x07
```

The driver library provides a function, `dmawpwmSetBuf`, that allows the application engineer to modify the content of the waveform pattern buffer.

See the section on “PWM Advanced Programming Functions” later in this chapter for details on `dmawpwmSetBuf`.

- **Set up the clock.**

The DMA device transfer from memory to I/O port address 0x4160 is driven by falling edges on signal /DREQ0. Since /DREQ0 is connected to CKA1 (the clock output of communication channel 1), the communication speed of communication channel 1 determines how frequently the DMA device transfer memory to I/O. Each transfer corresponds to one edge in the previous section.

Refer to a Zilog or a Hitachi user’s manual for the Z180 or 64180 MPU for more details on how to set up the frequency of CKA1.

The driver includes a function, `dmawpwmInit`, that sets up the frequency of CKA1.

The PWM interface sets up CKA1 to clock at 76,800 Hz in the call `_eioSetupA01st()`.

See the section on “PWM Advanced Programming Functions” later in this chapter for details on `dmawpwmInit`.

- **Refresh the DMA counter and source address.**

The DMA device does not automatically reload the counter and source address registers when the specified amount of bytes is transferred. When the DMA device finishes transferring the specified amount of bytes, it stops and optionally causes an interrupt. In other words, the PWM waveform is abruptly ended when the DMA finishes.

To overcome this limitation, the application must periodically “refresh” the counter and source address registers of the DMA device. The refresh should check whether the counter is less than a critical number. If so, both the counter and the source address registers must be “rebound” to a previous state (a larger counter value and a corresponding lower source address).

Note that in refreshing the registers, the PWM waveforms cannot be disrupted. In other words, the previous state to which the refresh routine restores must be phase synchronized with the PWM waveforms at the moment.

The driver library provides a refresh routine, `_eioBrdAORf`, to refresh the DMA counter and source address registers. `_eioBrdAORf ()` can be called from a preemptive task or from the main program. The refresh routine must be called frequently enough so that the DMA counter never reaches 0. The following inequality states the requirement.

$$f_r \geq f_1 / (l_{\text{wpb}} / 2)$$

where f_r is the refresh frequency, f_1 is the frequency of CKA1, and l_{wpb} is the total length of the waveform pattern buffer.

For example, `_eioSetupA01st ()` sets up $f_1 = 76,800$ Hz, and $l_{\text{wpb}} = 4096$. As a result, the application engineer must ensure $f_r \geq 37.5$ Hz.

- **Changing duty cycles.**

Once the PWM waveforms are up and running, the application may need to change the duty cycles for the channel(s). This poses two problems. First, the change should only be done to the channel that needs a change of duty cycle; all other channels should remain the same. Second, the change must become effectively phase synchronized with the current waveform.

The solution to the first problem depends on how the edges are represented. In particular, it depends on whether the “noop” edges are used. If the noop edges are used, changing duty cycle is a matter of moving the edges that are not “noop.” For example, in the “*Set up the waveform*” step, if we wish to change the duty cycle of OUT-01 to 0.25, we change the waveform from

```
0x80, 0x81, 0x07, 0x07, 0x07, 0x07, 0x00, 0x07,
0x07, 0x07, 0x07, 0x07, 0x07, 0x01, 0x07, 0x07
```

to

```
0x80, 0x81, 0x07, 0x07, 0x00, 0x07, 0x07, 0x07,
0x07, 0x07, 0x07, 0x07, 0x07, 0x01, 0x07, 0x07 .
```

The underlined edges are the only ones affected.

Of course, the pattern may be repeated many times in the waveform pattern buffer. Each occurrence of the pattern in the buffer must be modified in the same manner.

However, although the use of “noop” edges seems to be compute-time inexpensive, it does require the application to maintain the location of the non-noop edges. In other words, the application must maintain a duty cycle variable for each channel in addition to a variable for the waveform pattern buffer.

Recall that the second problem of changing the duty cycle is the requirement for the change to be phase synchronized to the current waveform. Many of the involved issues are similar to those of refreshing the DMA counter and pointer. The driver software library provides the function `dmawpwmSwBuf` to switch waveform pattern buffers.



The following sample programs in the Dynamic C `SAMPLES\PK23XX` subdirectory provide more detailed examples.

`DMAPWM1 . C`
`DMAPWM2 . C`
`DMAPWM3 . C`

PWM Advanced Programming Functions

- `void dmapwmSetBuf(char *pBufStart,
char bufLength256,
unsigned step,
char outChar)`

Formats part of the waveform pattern buffer for DMA-driven PWM.

PARAMETERS: `pBufStart` points to the first byte to be formatted. Note that `pBufStart` does not always have to point to a 256-byte aligned address.

`bufLength256` is the length of the buffer, including the overflow area.

`step` is the number of bytes to skip between outputting `outChar`.

`outChar` is the actual bytes to send to the I/O address.



The following sample programs in the Dynamic C `SAMPLES\PK23XX` subdirectory provide more detailed examples.

```
DMAPWM1.C  
DMAPWM2.C  
DMAPWM3.C
```

- `void dmapwmSwBuf(unsigned newBuf256)`

In order to facilitate all-or-none duty-cycle transitions, you should use two buffers. While one buffer is being used by the DMA mechanism to generate the PWM output, modify the other buffer for the new PWM pattern. When the new buffer is ready, this function should be called to switch to use the buffer at the address pointed to by `newBuf256` in 256-byte units.

- `char *dmapwmBufBeg(char *bufPtr)`

The buffer used by the PWM mechanism starts at 256-byte boundaries. Normal data definition declarations such as

```
char buffer[0x2000]
```

start at byte boundaries.

RETURN VALUE: a character pointer that points to the first 256-byte aligned root address larger than or equal to the parameter `bufPtr`.

- `void dmapwmInit(unsigned phyBuffer256,
 unsigned bufSize256,
 unsigned resSize256,
 unsigned ioAddr,
 char cka1rate)`

Initializes the DMA PWM mechanism.

When the function returns, CKA1 of communication port 1 generates clock pulses at `cka1rate` \approx 19.2 kHz to /DREQ0. DMA Channel 0 would then perform memory-to-I/O transfer for each clock pulse falling edge.

PARAMETERS: `phyBuffer256` is the 256-byte aligned physical address of the buffer in 256-byte units. In general, if the buffer is defined as an array in root memory (i.e., of type `(char *)`), the following expression should be passed to this parameter.

`(unsigned) ((xaddr(buffer)+255)>>8)`

in which `buffer` is a pointer of type `(char *)` to the array.

- `bufsize256` is the size of the buffer, in 256-byte units. This size should not include the overflow area.
- `resSize256` is the size of the overflow area in 256-byte units.
- `ioAddr` is the port to which the DMA should transfer memory content.
- `cka1rate` is the clock rate generated by CKA1 in multiples of 19.2 kHz. Allowed numbers are 2, 4 and 8.

Blank



APPENDIX A: TROUBLESHOOTING

Appendix A provides procedures for troubleshooting system hardware and software. The following sections are included.

- Out of the Box
- Dynamic C Will Not Start
- Dynamic C Loses Serial Link
- PK2300 Repeatedly Resets
- Common Programming Errors

Out of the Box

Check the items mentioned in this section before starting development.

- Do not connect any RS-485 equipment or I/O devices until the PK2300 has been verified to run standalone.
- Verify that the entire host system has good, low-impedance, separate grounds for analog and digital signals. Often the PK2300 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 PK2300.
- Verify that the host PC's COM port works by connecting a 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 Z-World power supply that comes with the developer's kit. If another power supply must be used, verify that it has enough capacity and filtering to support the PK2300.
- Use the Z-World cables that come with the developer's kit. The most common fault of user-made cables is failure to properly assert CTS at the RS-232 port of the PK2300. Without CTS being asserted, the PK2300'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 PK2300's RTS.
- The PK2300 developer's kit comes with an RJ-12 cable. This cable looks very much like a standard U.S. telephone cable, except it uses 6-pin RJ-12 connectors, not 4-pin RJ-11 connectors.



A regular U.S. telephone RJ-11 connector will plug in but *will not work*.

- Experiment with each peripheral device connected to the PK2300 to determine how it appears to the PK2300 when powered up, powered down, and/or when its connecting wiring is open or shorted.

Dynamic C Will Not Start

If Dynamic C will not start, an error message such as **Target Not Responding** or **Communication Error** appears on the Dynamic C screen. The following situations and their possible resolutions are usually behind these error messages.

- *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 one 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 PK2300's jumper is set for standalone operation. Reconfigure the board for programming mode.
- If all else fails, connect the serial cable to the PK2300 after powerup. 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 powerup to eliminate this problem.
- To reset the PK2300, recycle power or momentarily short pins 5 and 6 on header H1.

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 the application program. Make sure that interrupts are not disabled for a period greater than 50 ms.

PK2300 Repeatedly Resets

The PK2300 resets every second 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

- 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 type mismatches 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.

- Values for constants or variables out of range. Table A-1 lists acceptable ranges for variables and constants.
- Confusing a function's definition with an instance of its use in a listing.
- Not ending statements with semicolons.

Table A-1. Ranges of Dynamic C Function Types

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

- Not inserting commas as required in function parameter lists.
- Leaving out ASCII space character between characters forming a different legal—but unwanted—operator.
- Confusing similar-looking operators such as `&&` with `&`, `==` with `=`, and `//` with `/`.
- Inadvertently inserting ASCII nonprinting characters into a source-code file.
- The program seems to lock up when the PK2300 is configured for Resistive Measurement Input, and causes a watchdog reset after `_eioSetupAI1st`. This may be because the resistance being measured is too small. To resolve the problem, try adding a constant resistor in series with the sensor being measured. If the total resistance is at least 1.5 k Ω , the interval between DMA1 interrupts will be about 0.8 ms, which is enough time for the processor to perform computations other than handling DMA1 interrupts.
- The DMA-driven PWM function drives the output correctly for a while, then some channels remain ON, while others remain OFF. The most likely cause is `_eioBrdaORf` is not called frequently enough. Either increase the frequency with which this function is called, increase the size of the waveform platform buffer, or slow down the clock `CKA1`.



APPENDIX B: SPECIFICATIONS

Appendix B provides comprehensive PK2300 physical, electronic and environmental specifications. The following sections are included.

- Electrical and Mechanical Specifications
- Factory Default Jumper Positions
- Protected Digital Inputs
- High-Current Drivers

Electrical and Mechanical Specifications

Table B-1 lists electrical, mechanical, and environmental specifications for the PK2300.

Table B-1. PK2300 Specifications

Parameter	Specification
Enclosure Size	2.96"W × 5.00"L × 1.81"H (75 mm × 127 mm × 46 mm)
Board Size	2.82"W × 3.75"L × 1.50"H (71.2 mm × 95.2 mm × 38.1 mm)
Operating Temperature	-40°C to 60°C with enclosure in preferred orientation
Humidity	5% to 95%, noncondensing
Input Voltage and Current	9 V to 24 V DC, 147 mA
Configurable I/O	6 inputs, 5 outputs
Digital Inputs	11 protected, -20 V to +24 V DC
Digital Outputs	<ul style="list-style-type: none"> • 8 high-current sinking, 75 mA/channel at 48 V (all channels ON) at 60°C, 1 channel can sink up to 500 mA continuously at 25°C • (optional) 8 high-current sourcing, 75 mA/channel at 30 V at 60°C • (optional) CMOS-level outputs
Analog Inputs	See Resistance Measurement Input
Analog Outputs	Up to 7 PWM digital outputs
Resistive Measurement Input	One, 0 Ω to 275 kΩ
Processor	Z180
Clock	9.216 MHz
SRAM	32K standard
EEPROM	Flash memory may be used as EEPROM
Flash EPROM	128K standard
Serial Ports	2 RS-232 or 1 RS-232 with RTS/CTS and 1 RS-485
Serial Rate	Up to 57,600 bps
Watchdog/Supervisor	Yes
Time/Date Clock	Yes
Backup Battery	BR2325-1HG or equivalent, 3-year shelf life, 10-year life in use

PK2300 External Dimensions

Figure B-1 illustrates external dimensions for the PK2300.

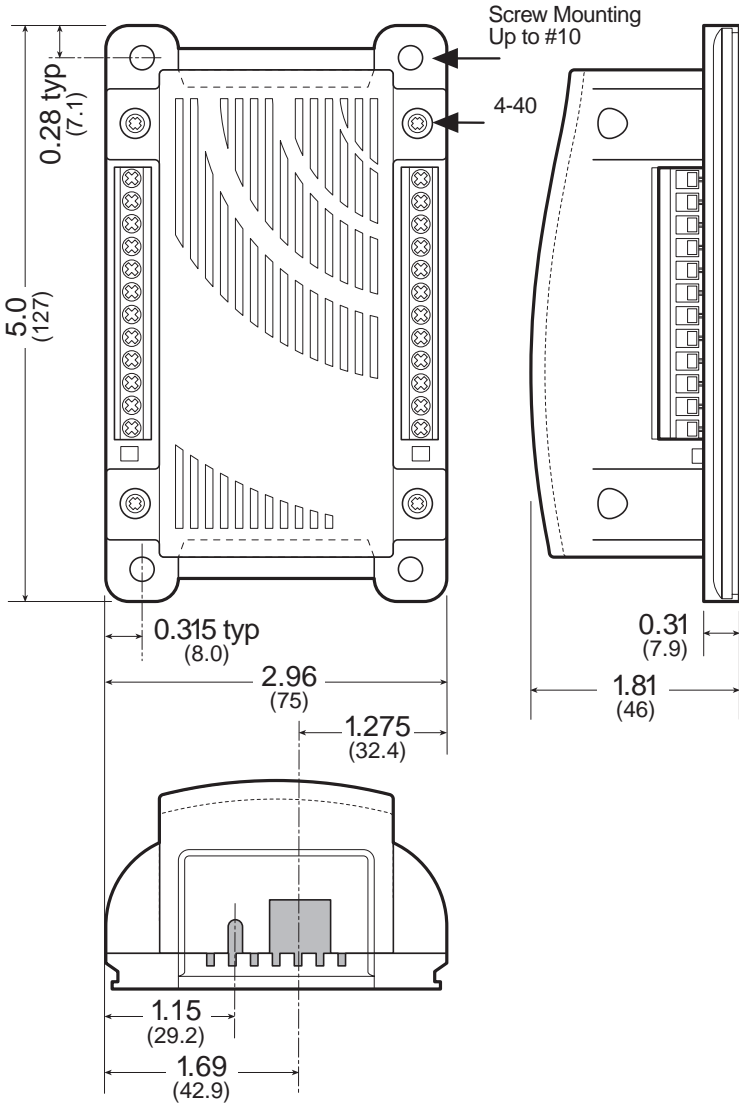


Figure B-1. PK2300 External Dimensions

Factory Default Jumper Positions

Figure B-2 illustrates the of header locations on the PK2300 main board. The jumpers configurations for these headers are listed in Table B-2.

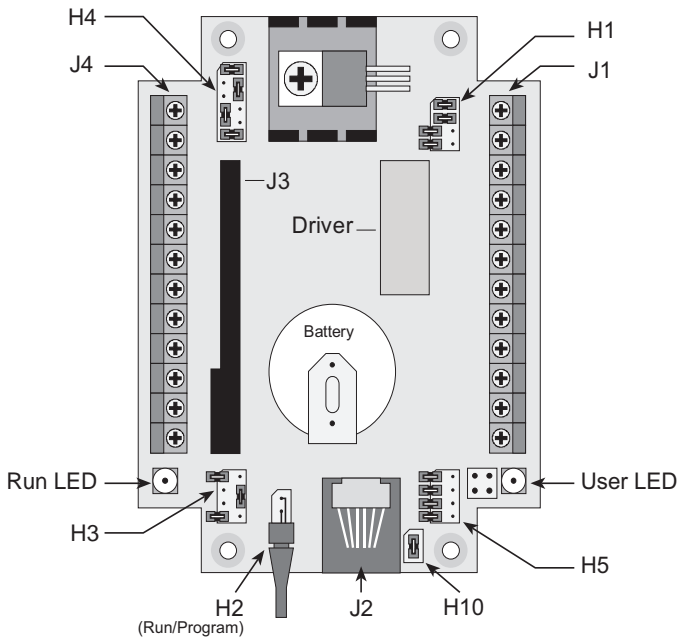


Figure B-2. Locations of PK2300 Headers

Table B-2. PK2300 Jumper Connections

Header	Pins	Description	Factory Default
H1	1-4	Connect pins 1-2 and pins 3-4 for sinking driver Connect pins 1-3 and pins 2-4 for sourcing driver	Pins 1-2 and pins 3-4 connected
	5-6	Connect to reset processor, not connected allows Run or Program mode	Not connected
	7-8	Connect to enable pin 7, header J1, as IN-16	Not connected; J1 pin 7 is high-current output OUT-04

continued...

Table B-2. PK2300 Jumper Connections (concluded)

Header	Pins	Description	Factory Default
H2	1–2	Run/program jumper: connect for Program mode, disconnect for Run mode	Connected
H3	1–2 7–8	Connect for 3-wire RS-232 on Z0 and 3-wire RS-232 on Z1	Not connected
	1–3 5–7	Connect for 5-wire RS-232 on Z0 and RS-485 on Z1	Not connected
	4–6	Connect for 3-wire RS-232 on Z0 and RS-485 on Z1	Connected
H4	1–6	Connect pins 1–2 and pins 4–6 to configure J4 pins 9 and 10 as protected digital inputs IN-08 and IN-09	Pins 1–2 and pins 4–6 connected to enable protected digital inputs IN-08 and IN-09 on J4 pins 9 and 10
		Connect pins 1–3 and pins 5–6 to configure J4 pins 9 and 10 as resistive measurement inputs RMI+ and RMI–	
	7–12	Connect pins 7–9 and pins 11–12 to configure J4 pins 11 and 12 as RS-485	Pins 7–9 and pins 11–12 connected to enable RS-485 on J4 pins 11 and 12
		Connect pins 7–8 and pins 10–12 to configure J4 pins 11 and 12 as protected digital inputs IN-10 and IN-11	
H5	1–2	Connect to enable J1 pin 11 as protected digital input IN-12	Not connected; J1 pin 11 is high-current output OUT-08
	3–4	Connect to enable J1 pin 10 as protected digital input IN-13	Not connected; J1 pin 10 is high-current output OUT-07
	5–6	Connect to enable J1 pin 9 as protected digital input IN-14	Not connected; J1 pin 9 is high-current output OUT-06
	7–8	Connect to enable J1 pin 8 as protected digital input IN-15	Not connected; J1 pin 8 is high-current output OUT-05
H10	1–2	Connect to enable 120 Ω termination resistor for multidrop network	Connected



See Chapter 3, “Input/Output Configuration,” for more details on the jumper configurations.

Protected Digital Inputs

Table B-3 lists the specifications for the protected digital inputs.

Table B-3. Protected Digital Input Specifications

Protected Digital Inputs	Maximum Rating
Digital Input Voltage	-20 V to +24 V
Digital Input Current	15 mA
Frequency Response (worst case)	Faster than 656 Hz, not longer than 1.52 ms (input 5 V DC)

Frequency Response for IN-01 to IN-05, and IN-08 to IN-16

The protection network consists of a low-pass filter with a 3 dB downpoint of 723 Hz. For example, if the driving source of a protected input is a step function, that step becomes available 1.38 ms later as a valid +5 V DC CMOS input to the PK2300's data bus.

The following formula shows how R_{IN} and C affect the frequency response of protected inputs IN-01 through IN-05 and IN-08 through IN-16.

$$f_c = [2\pi R_{IN} C]^{-1} = [(2\pi)(22 \times 10^3)(10 \times 10^{-9})]^{-1} = 723 \text{ Hz}$$

$$t = [f_c]^{-1} = 1.38 \text{ ms (at 0.707 of full input value)}$$

Figure B-3 shows the circuit for protected inputs IN-01 through IN-05, and IN-08 through IN-16 in the factory-default pulled-up configuration.

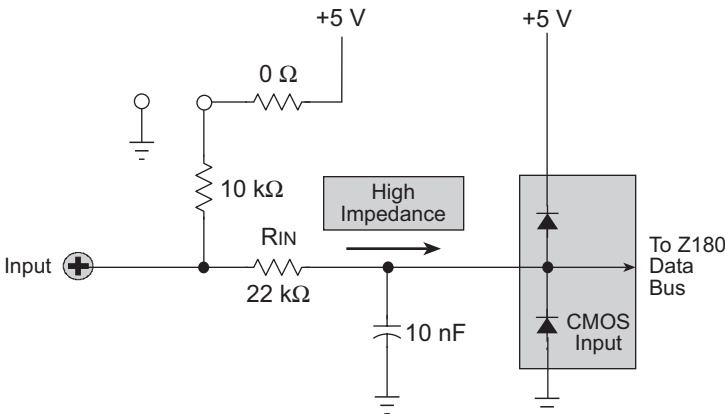


Figure B-3. Protected Digital Inputs IN-01 to IN-05, and IN-08 to IN-16

Frequency Response for IN-06 and IN-07

These two interrupt-ready inputs also have a protection network that consists of a low-pass filter. The 3 dB downpoint is at 7.23 kHz. For example, if the driving source of these two protected inputs is a step function, then that step voltage registers approximately 138 ms later as a valid +5 V DC CMOS input to the PK2300's data bus.

Figure B-4 shows the circuit for IN-06 and IN-07 in the factory-default pulled-up configuration.

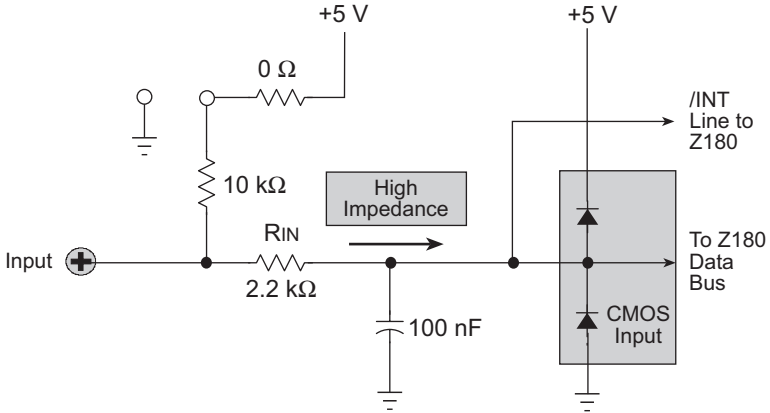


Figure B-4. Level-Sensitive Inputs IN-06 and IN-07

Customization

Frequency Response and Input Range

A faster frequency response is possible by replacing R_{IN} with a smaller resistor. For example, if the digital input is being driven by a +5 V DC CMOS-compatible driver, R_{IN} may be replaced with a 0 Ω valued 0805 resistor.



A 0 Ω resistor for R_{IN} will adversely affect the PK2300's noise immunity.



The PK2300 may be ordered in quantity with a customer-specified resistor installed by the factory at R_{IN} . For more details, contact your Z-World Sales Representative at (530)757-3737.

Default Pull-Up Assignments

All protected digital inputs are factory-configured with +5 V DC pull-up resistors. Since there are two banks (or groups) of inputs, each bank can be uniquely pulled up to +5 V DC or pulled down to ground by two separate permanent surface mount 0 Ω resistors.

- Bank **ONE** includes the dedicated protected digital inputs IN-01 to IN-04. Bank **ONE** is assigned via JP9 (0 Ω resistor).
- Bank **TWO** includes all the other inputs (IN-05 through IN-16), and is configured with JP10 (0 Ω resistor).

High-Current Drivers

Table B-4 lists the high-current driver characteristics when sinking drivers or sourcing drivers are used.

Table B-4. High-Current Driver Characteristics


Characteristic	Sinking Driver	Sourcing Driver
		
IC Model Number	2803	2985
Channels	8	8
Max. Current per Channel (all channels ON)	75 mA @ 60°C 125 mA @ 50°C	75 mA @ 60°C 125 mA @ 50°C
Voltage Source Range	2 V to 48 V DC	15 V to 30 V DC
Package Power Dissipation	2.2 W	2.2 W
Max. Current (all channels ON)	1.38 A	1.38 A
Max. Collector-Emitter Voltage (V_{CE})	1.6 V	1.6 V
Derating	18 mW/°C (55°C/W)	18 mW/°C (55°C/W)
Output Flyback Diode (K)	Yes	Yes
Max. Diode-Drop Voltage (K)	2 V DC	2 V DC

Table B-5 lists safe operating conditions for the PK2300 high-current drivers.

Table B-5. High-Current Driver Safe Operating Range at 60°C

No. Outputs ON	Current/Channel	Duty Cycle
8	75 mA	100%
4	150 mA	100%
2	300 mA	100%
8	150 mA	50%
4	300 mA	50%
2	500 mA	50%



See Appendix F, “Enclosure Mounting,” for more details on mounting and heat dissipation.

Function of “K”

The common supply for all eight output channels is called “K,” and is so labeled on the PK2300’s terminals. “K” must be powered up to allow proper operation of the PK2300.

The “K” connection performs two functions for the high-current drivers.

1. “K” supplies power to the driver circuits inside the driver chips.
2. “K” also allow a diode internal to the driver chips to “snub” voltage transients produced during inductive kick (associated with switching inductive loads). Relays, solenoids, and speakers are examples of inductive loads.

Long leads may present enough induction to also produce large, potentially damaging voltage transients. Because the anodes of the protection diodes for each bank of channels are common, only one supply voltage can be used for each bank of high-current driver loads.

Figure B-5 and Figure B-6 illustrate the use of the “K” connections for sinking and sourcing configurations. Note that the load’s supply must have a common ground with all other supplies in your system, and all loads must use the same supply voltage.

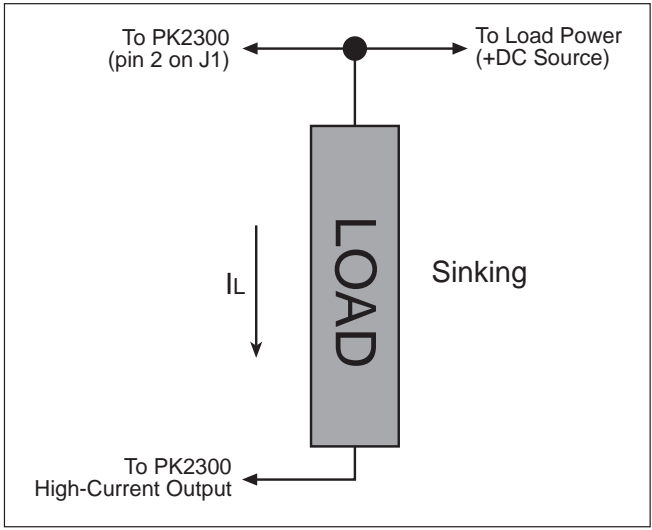


Figure B-5. PK2300 “K” Connection (Sinking Configuration)

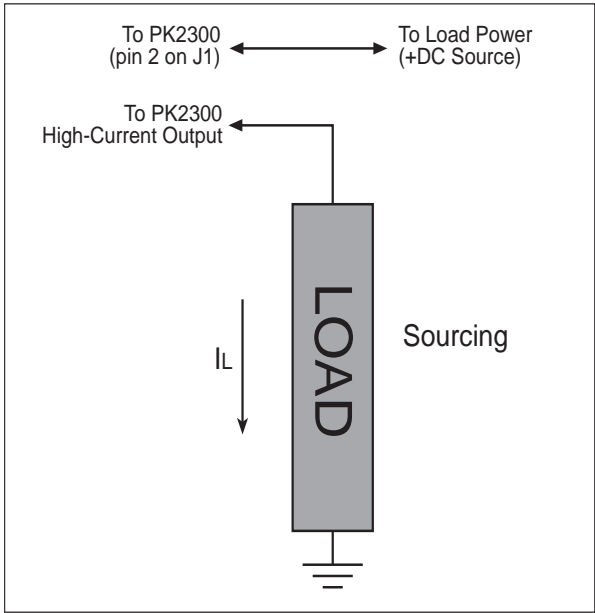


Figure B-6. PK2300 “K” Connection (Sourcing Configuration)



“K” must be connected to your load’s power supply as shown in Figure B-5 or Figure B-6.



APPENDIX C: **SERIAL INTERFACE BOARD 2**

Appendix C provides technical details and baud rate configuration data for Z-World's Serial Interface Board 2. The following sections are included.

- Introduction
- External Dimensions

Introduction

The Serial Interface Board 2 (SIB2) is an optional interface adapter used to program the PK2300. The SIB2 is contained in an ABS plastic enclosure, making it rugged and reliable. The SIB2 enables the PK2300 to communicate with Dynamic C via the Z180's clocked serial I/O (CSIO) port, freeing all PK2300 serial ports for use by the application during programming and debugging.

The SIB2's 8-pin cable plugs into header JP1 of the PK2300's CM7200 core module, and a 6-conductor RJ-12 phone cable connects the SIB2 to the host PC. The SIB2 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 SIB2 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 PK2300 (which also resets the SIB2), then select **Reset Target** from Dynamic C.

The SIB2 receives power and resets from the target PK2300 via the 8-pin connector. Therefore, do not unplug the SIB2 from the target PK2300 while power is applied. To do so could damage both the PK2300 and the SIB2; additionally, the target may reset.

The SIB2 consumes approximately 60 mA from the +5 V supply. Target-system current consumption therefore increases by this amount while the SIB2 is connected to the PK2300.



Never connect or disconnect the SIB2 with power applied to the controller.

External Dimensions

Figure C-1 illustrates the external dimensions for the Serial Interface Board 2.

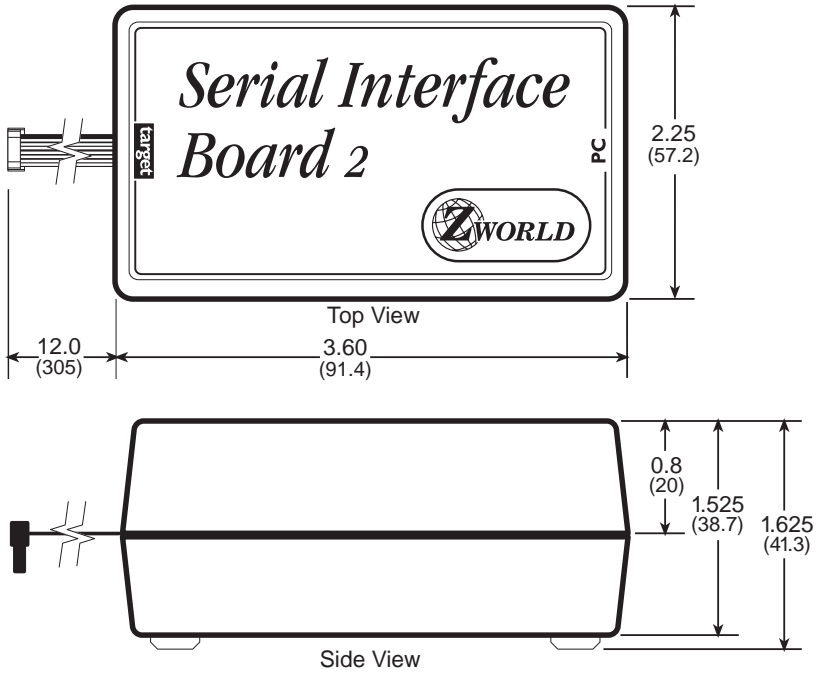


Figure C-1. Serial Interface Board 2 External Dimensions

Blank



APPENDIX D:
SINKING VS. SOURCING DRIVERS

Appendix D provides detailed information about sinking and sourcing high-current drivers.

The PK2300 has up to eight high-current driver outputs that can be configured as “sourcing” or “sinking.” The configuration is selected by installing the appropriate driver IC and by setting header H1. The factory-installed driver chip and default jumper settings are for “sinking” control (ULN2803).

Figure D-1 shows the locations of the driver ICs at U1 and U2. Figure D-2 shows the jumper configurations on header H1.

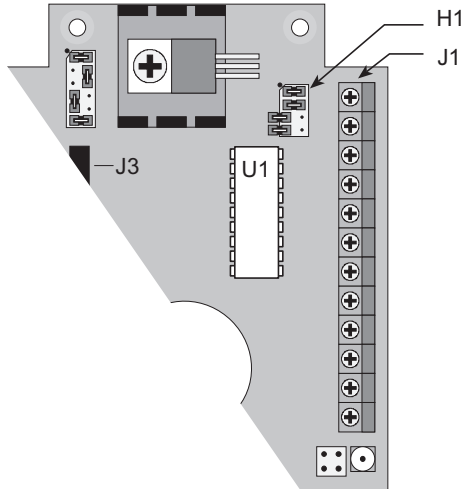


Figure D-1. PK2300 High-Current Output Driver Chip at U1

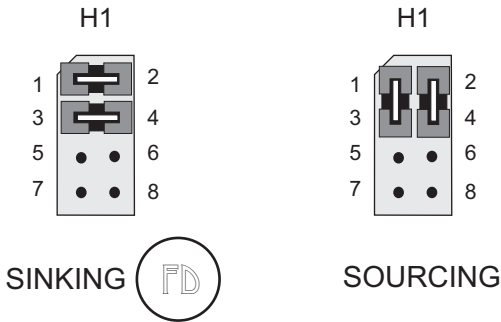


Figure D-2. PK2300 Header H7 Jumper Configurations

Selecting Sourcing or Sinking Drivers

The ULN2985 sourcing driver is included in the Developer's Kit. Since the driver IC is socketed, the factory-default ULN2803 sinking driver can be easily substituted.



The CM7200 Core Module needs to be removed to access the high-current output driver. The CM7200 Core Module is plugged onto header J3 and held in place by one screw.



The PK2300 is available in quantity with a sourcing driver factory-installed. For more information, call your Z-World Sales Representative at (530) 757-3737.

Sinking Driver (Low-Side Drive)

The ULN2803 sinking driver can handle up to 500 mA for any channel, or an absolute maximum of 1.38 A, which represents an average of 75 mA per channel with all channels ON at 60°C. The absolute maximum power that the ULN2803 can dissipate is 2.2 W. The saturation voltage is a maximum of 1.6 V DC per channel. The sinking driver's source voltage must range from 2 V to 48 V.

Figure D-3 illustrates the sinking driver configuration.

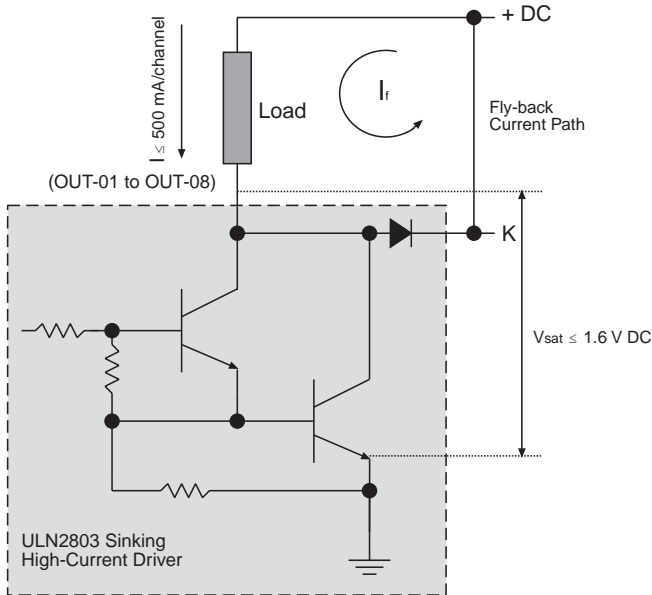


Figure D-3. ULN2803 Sinking Driver

Sourcing Driver (High-Side Drive)

The ULN2985 sourcing driver can handle up to 500 mA for any channel, or an absolute maximum of 1.38 A, which represents an average of 75 mA per channel with all channels ON at 60°C. The absolute maximum power that the ULN2985 can dissipate is 2.2 W. The saturation voltage is a maximum of 1.6 V DC per channel. The sourcing driver's source voltage must range from 15 V to 30 V.

Figure D-4 illustrates the sourcing driver configuration.

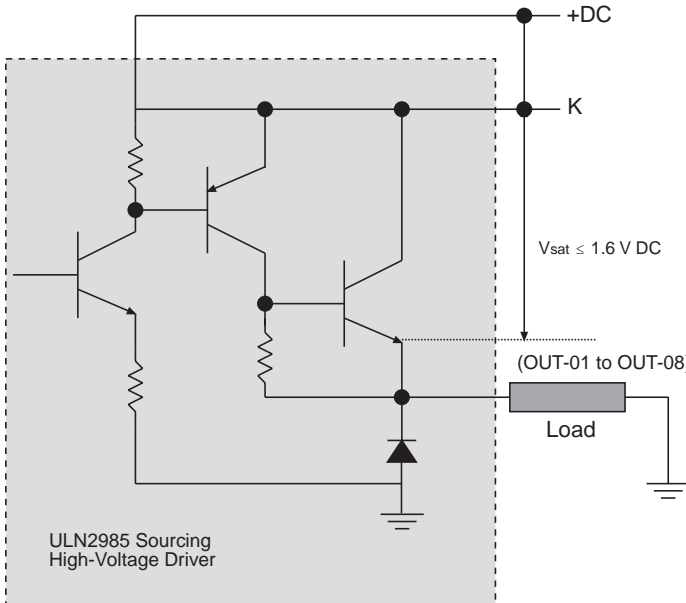


Figure D-4. ULN2985 Sourcing Driver



See the Motorola (DL128) or Allegro (AMS 502Z) **Linear Data Books** for more information on sinking and sourcing high-current drivers.



APPENDIX E: POWER MANAGEMENT

Appendix E provides detailed information about the power systems and how the PK2300 handles power failures.

Power-Failure Detection Circuitry

Figure E-1 shows the PK2300's power-failure detection circuit. Note that the 691 trips at 1.3 V DC.

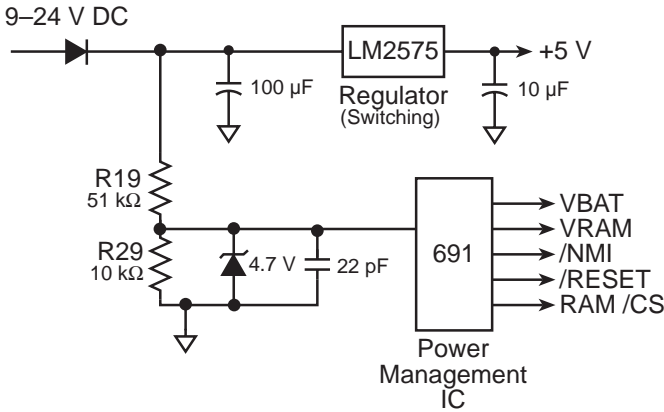


Figure E-1. PK2300 Power-failure Detection Circuitry

Power Failure Sequence of Events

Figure E-2 summarizes the events that occur as the input power fails.

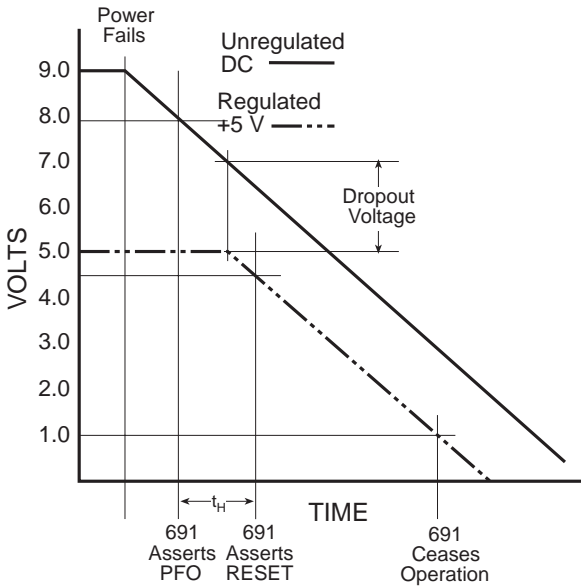


Figure E-2. Power Failure Sequence

First, the 691 power-management IC triggers a power-failure /NMI (nonmaskable interrupt) when the *unregulated* DC input voltage (9 V to 24 V DC) falls in the range $8.02\text{ V} \leq +\text{DC} \leq 8.63\text{ V}$ (as determined by the voltage divider).

At some point, the raw input voltage level will not exceed the required regulated voltage level by the regulator's *dropout voltage* (see Figure E-2). The regulated output voltage then begins to drop. The 691 triggers a system reset when the *regulated* +5 V supply is in the range from 4.50 V to 4.75 V DC, allowing the power-failure routine a “holdup” interval, t_{H} , to store important state data. The 691 forces the chip select (/CS) of the SRAM to high (standby mode).

Tip To increase the “holdup” interval t_{H} , use a power supply with a large capacitance. This will provide additional time for the PK2300 to execute a safe shutdown.

The time/date clock and SRAM switch to the lithium backup battery when the regulated voltage falls below the battery voltage of approximately 3 V.

The 691 keeps the reset (/RESET) enabled until the +5 V regulated voltage drops below 1 V. The 691 ceases operating at this point. By this time, the portion of the circuitry not battery-backed should have long since ceased functioning.

The ratio of the power supply's output capacitance to a circuit's current draw determines the actual duration of the holdup time interval, t_{H} .

A situation similar to a continuous low input (“brownout”) can occur if the power supply is overloaded. For example, when a high-current device such as a relay turns ON, the raw voltage supplied to the PK2300 may dip below 7.9 V. The interrupt routine does a shutdown. This shutdown turns the LED off, clearing the problem. However, if the cause of the overload persists, the system oscillates, alternately experiencing an overload and then resetting. To correct this situation, get a larger, “stiffer” power supply.

If the power cable is removed abruptly from the PK2300, then only the capacitors on the board are available, reducing computing time to a few microseconds. These times can vary considerably, depending on system's configuration and loads on the +5 V regulated and the 9 V to 24 V unregulated supplies.

The interval between power-failure detection and the entry to the power-failure interrupt routine is approximately 100 μs , or less if the Dynamic C /NMI communication is not in use.

Blank



APPENDIX F: ENCLOSURE MOUNTING

Appendix F provides technical details for the PK2300 enclosure, and includes mounting suggestions for the enclosure.

Enclosure Mounting Considerations

The enclosure orientation affects the ability of the PK2300 to dissipate or remove heat from its circuitry to the outside ambient environment. Heat removal is crucial to reliable operation.

The preferred orientations of the PK2300 enclosure are shown in Figures F-1 to F-4 in order of recommended to least desirable.

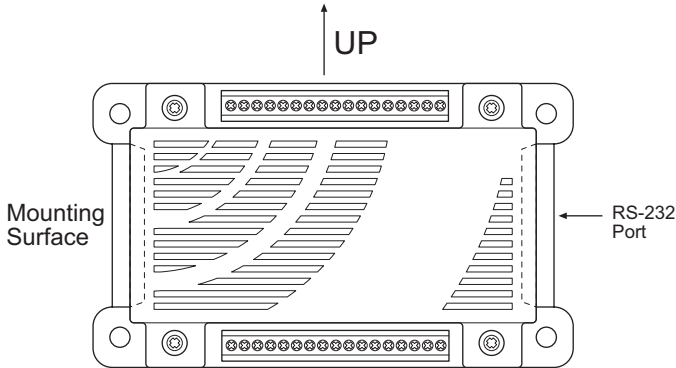
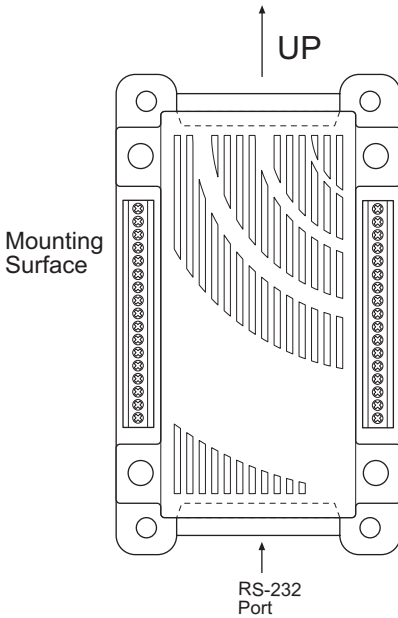


Figure F-1. Preferred PK2300 Enclosure Mounting



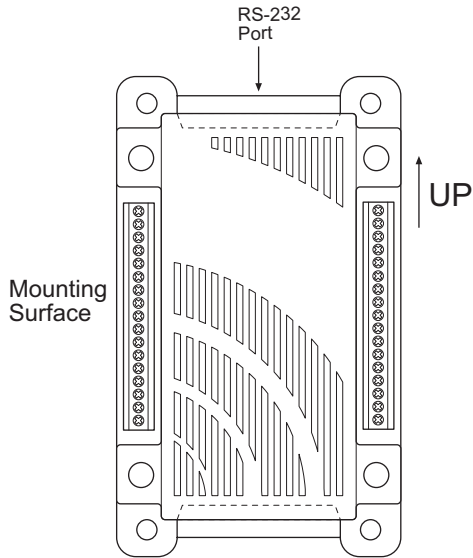


Figure F-3. Acceptable PK2300 Enclosure Mounting

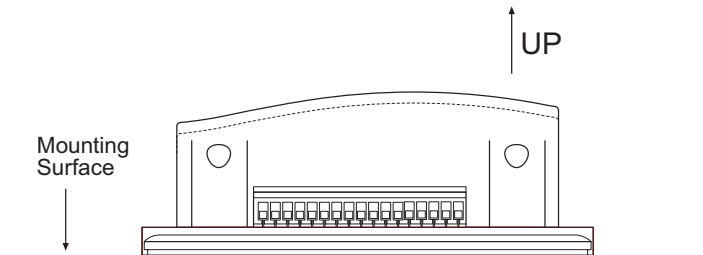
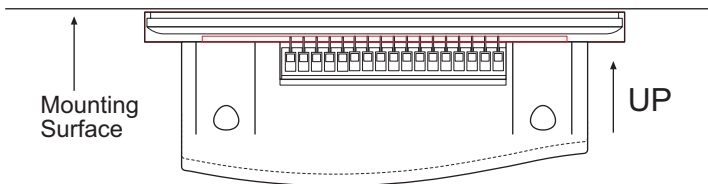


Figure F-4. Acceptable PK2300 Enclosure Mounting



**Figure F-5. PK2300 Enclosure Mounting
(not recommended)**



The mounting configuration shown in Figure F-5 is not recommended. Components on the PK2300 may overheat, leading to failure or reduced operating life.

Mounting Controllers on DIN Rails

The PK2300 can be installed in modular plastic circuit-board holders attached to a DIN rail, a widely used mounting system, as shown in Figure F-6.

The circuit-board holders are 77 mm wide and come in lengths of 11.25 mm, 22.5 mm, and 45 mm. The holders, available from Z-World, snap together to form a tray of almost any length. Z-World's PK2300 enclosure is 75 mm wide and fits directly in these circuit-board holders.

The PK2300 can also be mounted with plastic standoffs to any flat surface that accepts screws. The mounting holes are 0.315 inches (8 mm) in from the edge of a board, and can accept up to #10 mounting screws.

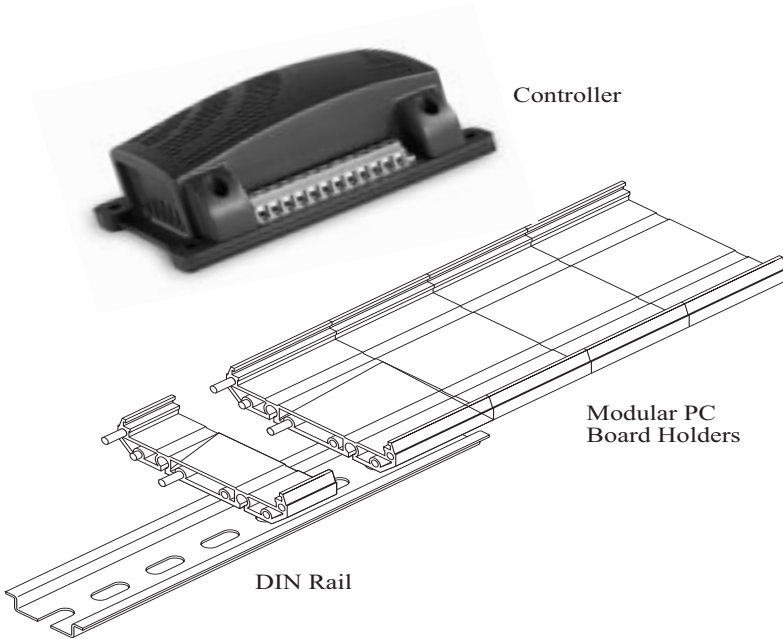


Figure F-6. Mounting PK2300 on DIN Rail



For information on ordering DIN rail mounts, call your Z-World Sales Representative at (530) 757-3737.



*APPENDIX G: **NONVOLATILE STORAGE***

Appendix G provides information about the flash EPROM memory used in the PK2300.

Flash memory acts as EEPROM. Unlike many older Z-World controllers, the PK2300 does not have an EEPROM for nonvolatile storage of important system parameters. However, the PK2300 flash EPROM simulates EEPROM, and uses the same Dynamic C function calls as other Z-World controllers to read and write nonvolatile data.

The flash EPROM constants listed in Table G-1 apply to the PK2300.

Table G-1. PK2300 Flash EPROM Constants

Address	Definition
0	Startup mode. If 1, enter program mode. If 8, execute loaded program at startup.
1	Programming baud rate in multiples of 1200 bps. The factory value is 16 (19,200 bps).
2	RMI calibration constant.
3	RMI calibration constant.
4	RMI calibration constant.
5	RMI calibration constant.
6	RMI calibration constant.
7	RMI calibration constant.



APPENDIX H: ***I/O MAP AND INTERRUPT VECTORS***

Appendix H provides information on the PK2300's memory mapping, which is based on the CM7200 core.

CM7200 Input/Output Map

The internal registers for the input/output devices built into the Z180 processor occupy the first 40 (hex) addresses of the I/O space. Table H-1 lists the addresses of these internal registers.

Table H-1. CM7200 Z180 Internal I/O Registers

Address	Name	Description
0x00	CNTLA0	Control Register A, Serial Channel 0
0x01	CNTLA1	Control Register A, Serial Channel 1
0x02	CNTLB0	Control Register B, Serial Channel 0
0x03	CNTLB1	Control Register B, Serial Channel 1
0x04	STAT0	Status Register, Serial Channel 0
0x05	STAT1	Status Register, Serial Channel 1
0x06	TDR0	Transmit Data Register, Serial Channel 0
0x07	TDR1	Transmit Data Register, Serial Channel 1
0x08	RDR0	Receive Data Register, Serial Channel 0
0x09	RDR1	Receive Data Register, Serial Channel 1
0x0A	CNTR	Clocked Serial Control Register
0x0B	TRDR	Clocked Serial Data Register
0x0C	TMDR0L	Timer Data Register, Channel 0, low
0x0D	TMDR0H	Timer Data Register, Channel 0, high
0x0E	RLDR0L	Timer Reload Register, Channel 0, low
0x0F	RLDR0H	Timer Reload Register, Channel 0, high
0x10	TCR	Timer Control Register
0x11–13	—	<i>Reserved</i>
0x14	TMDR1L	Timer Data Register, Channel 1, low
0x15	TMDR1H	Timer Data Register, Channel 1, high
0x16	RLDR1L	Timer Reload Register, Channel 1, low
0x17	RLDR1H	Timer Reload Register, Channel 1, high
0x18	FRC	Free-Running Counter
0x19–1F	—	<i>Reserved</i>
0x20	SAR0L	DMA Source Address, Channel 0, low
0x21	SAR0H	DMA Source Address, Channel 0, high
0x22	SAR0B	DMA Source Address, Channel 0, extra bits

continued...

Table H-1. CM7200 Z180 Internal I/O Registers (concluded)

Address	Name	Description
0x23	DAR0L	DMA Destination Address Channel 0, low
0x24	DAR0H	DMA Destination Address Channel 0, high
0x25	DAR0B	DMA Destination Address Channel 0, extra bits
0x26	BCR0L	DMA Byte Count Register, Channel 0, low
0x27	BCR0H	DMA Byte Count Register, Channel 0, high
0x28	MAR1L	DMA Memory Address Register, Channel 1, low
0x29	MAR1H	DMA Memory Address Register, Channel 1, high
0x2A	MAR1B	DMA Memory Address Register, Channel 1, extra bits
0x2B	IAR1L	DMA I/O Address Register, Channel 1, low
0x2C	IAR1H	DMA I/O Address Register, Channel 1, high
0x2D	—	<i>Reserved</i>
0x2E	BCR1L	DMA Byte Count Register, Channel 1, low
0x2F	BCR1H	DMA Byte Count Register, Channel 1, high
0x30	DSTAT	DMA Status Register
0x31	DMODE	DMA Mode Register
0x32	DCNTL	DMA/WAIT Control Register
0x33	IL	Interrupt Vector Low Register
0x34	ITC	Interrupt/Trap Control Register
0x35	—	<i>Reserved</i>
0x36	RCR	Refresh Control Register
0x37	—	<i>Reserved</i>
0x38	CBR	MMU Common Base Register
0x39	BBR	MMU Bank Base Reg
0x3A	CBAR	MMU Common/Bank Area Register
0x3B–3D	—	<i>Reserved</i>
0x3E	OMCR	Operation Mode Control Register
0x3F	ICR	I/O Control Register

Real-Time Clock Registers

Table H-2 provides the real time-clock IC's internal registers.

Table H-2. Real-Time Clock Internal Registers

Address	Data Bits	Symbol	Meaning	Range
0x4180	D0–D7	SEC1	seconds	0–9
0x4181	D0–D7	SEC10	10 seconds	0–5
0x4182	D0–D7	MIN1	minutes	0–9
0x4183	D0–D7	MIN10	10 minutes	0–5
0x4184	D0–D7	HOUR1	hours	0–9
0x4185	D0–D7	HOUR10	10 hours	0–2
0x4186	D0–D7	DAY1	days	0–9
0x4187	D0–D7	DAY10	10 days	0–3
0x4188	D0–D7	MON1	months	0–9
0x4189	D0–D7	MON10	10 months	0–1
0x418A	D0–D7	YEAR1	years	0–9
0x418B	D0–D7	YEAR10	10 years	0–9
0x418C	D0–D7	WEEK	week days.	0–6
0x418D	D0–D7	TREGD	Register D	—
0x418E	D0–D7	TREGE	Register E	—
0x418F	D0–D7	TREGF	Register F	—

Other Input/Output Addresses

The I/O addresses listed in Table H-3 control the I/O devices that are external to the Z180 processor.

Table H-3. I/O Addresses for Devices External to Z180

Address	Description	Function
0x40C0	/TRG to 555	Read to start charging and turn /OUTPUT of 555 high
0x4140	RS-485 driver	Bit 0 indicates on/off: 1 to turn on, 0 to turn off
0x4141	LED D1 (RUN LED)	As above
0x4142	LED D2 (USER LED)	As above
0x4143	CMOS OUT9	As above
0x4144	CMOS OUT10	As above
0x4145	CMOS OUT11	As above
0x4146	CMOS OUT12	As above
0x4147	OUT8	As above
0x4160 (read)	IN-01 to IN-08	Bit x for state of Inx+1
0x4161 (write)	OUT-01 to OUT-07	Bits 0, 1, 2 indicate which high-current driver, bit 7 indicates on/off: 1 to turn on, 0 to turn off
0x4161	IN-09 to IN-16	Bit x for state of Inx+1
0xA000	Power failure	Bit 0 indicates whether power is failing: 1 power is failing, 0 if not

Interrupt Vectors

Table H-4 presents a suggested interrupt vector map. Most of these interrupt vectors can be altered under program control. The addresses are given here in hexadecimal, relative to the start of the interrupt vector page, as determined by the contents of the I-register. These are the default interrupt vectors set by the boot code in the Dynamic C EPROM.

Table H-4. Interrupt Vectors for Z180 Internal Devices

Address	Name	Description
0x00	INT1_VEC	Expansion bus attention INT1 vector.
0x02	INT2_VEC	INT2 vector.
0x04	PRT0_VEC	PRT Channel 0
0x06	PRT1_VEC	PRT Channel 1
0x08	DMA0_VEC	DMA Channel 0
0x0A	DMA1_VEC	DMA Channel 1
0x0C	CSI/O_VEC	Clocked Serial I/O
0x0E	SER0_VEC	Asynchronous Serial Port Channel 0
0x10	SER1_VEC	Asynchronous Serial Port Channel 1

A directive such as the following is used to “vector” an interrupt to a user function in Dynamic C.

```
#INT_VEC 0x10 myfunction
```

This statement causes the interrupt at offset 0x10 (Serial Port 1 of the Z180) to invoke the function `myfunction()`. The function must be declared with the `interrupt` keyword as follows.

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

Interrupt Priorities

Table H-5 lists the interrupt priorities from highest to lowest.

Table H-5. Interrupt Priorities

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

Blank



*APPENDIX I: **BATTERY***

Appendix I provides information about the onboard lithium battery.

Storage Conditions and Shelf Life

The lithium battery will provide approximately 9,000 hours of backup time for the onboard real-time clock and static RAM. However, backup time longevity is affected by many factors, including the amount of time the controller is unpowered and the static RAM size. The controller should be stored at room temperature in the factory packaging until field installation. Take care that the controller is not exposed to extreme temperature, humidity, and/or contaminants such as dust and chemicals.

To ensure maximum battery shelf life, follow proper storage procedures. Replacement batteries should be kept sealed in the factory packaging at room temperature until installation. 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 bottom side of the printed circuit board that secure it 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 BR2325-1HG 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 Behandlein 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. Deshagase 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)**

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

Blank

Symbols

#INT_VEC 126
 =(assignment)
 use 92

A

advanced I/O programming 75
AO1.C 57

B

battery
 cautions 131
 replacing 130
 baud rate
 programming 20, 22, 36, 91, 120
 board layout 13
 brownout 60, 113
bufLength256 86
bufPtr 86
bufSize256 87

C

cautions
 H1 31
 H5 31
 simultaneous I/O 31
 CE compliance 16
 changing duty cycles 84
chanNum 40, 46, 65
 definition 65
ckalrate 87
 clock
 setup 83
 CM7200 core module 26
 CMOS outputs 14, 34
 common problems
 cables 90
 COM port 91

common problems (continued)
 programming errors 91
 repeated resets 91
 communication
 ports 74
Compile 38
 icon 22
 program 22
 components 26
 configuring
 inputs and outputs 30
 serial communication 29
 connect PK2300 to PC 18
 customization 15

D

Darlington transistors 42
 default communication rate 20
 default jumper positions 96
 Developer's Kit 16
 device I/O map 75
 digital I/O 65
 addressing 76, 77
 output states 77
 using 39
 digital input 14
 dimensions
 PK2300 95
 SIB2 105
 DIN rails 118
 DMA counter 83
dmapwmBufBeg 86
dmapwmInit 87
dmapwmSetBuf 83, 86
dmapwmSwBuf 86
 duty cycle 56
 calculating 72
 changing 84
 Dynamic C 14, 16
 error messages 20
 serial options 20

E	
EEPROM	
simulated	120
EIO_NODEV	42, 46, 64
use	65, 71
eioBrdACalib	70
eioBrdAI	46, 70, 71
eioBrdAO	56, 72
eioBrdAORf	72
eioBrdDI	40, 65
eioBrdDO	42, 66
eioBrdInit	64, 70
eioErrorCode ...	42, 46, 64, 66
use	65, 71
eioSetupAI1st	64, 70, 92
eioSetupAO1st	64, 72
enclosure	
mounting	116
EPROM	126
flash	58
error messages	20, 22
establishing communication	20
EZIOPK23.LIB ...	64, 65, 70, 72
F	
F3	22
F9	22
features	14
flash EPROM	58
constants	120
writes	
lifetime	38
flash LEDs	22
flexibility	15
flexible I/O	27
float	
use	91
function of K	101
H	
H1	31
standard configuration	96
H10	52, 53
H2	
standard configuration	96
H3	
standard configuration	96
H4	33
standard configuration	96
H5	31
caution	31
standard configuration	96
halt program	22
header	
H1	31
H4	33
H5	31
H10	52, 53
high-current driver	
outputs	14, 42
specifications	100
I	
I/O	
addresses external to Z180 ..	125
advanced programming	75
combinations	14
configuration	29–31, 34
device I/O map	75
digital	14
flexible	27
functions	27
memory map	122
types	26
inputs	
demonstration program ...	40, 47
pull-up	40
int	
type specifier, use	91
interrupts	126
interrupt routine	113
interrupt service routine	69
interrupt vectors	126
priorities	127
ioAddr	87

J		O	
J2 pin assignments	50	offChar	86
jumpers		operating modes	36
default positions	96	changing	37
standard configuration	96	permissible activities	37
K		output	
K	101	channel assignments	66
L		CMOS	14, 34
LED	58	demonstration program ...	42, 43
addressing	78	high current	42
run	36	high-current driver	14
specifications	58	overload	113
user	36	P	
level-sensitive interrupts	68	pBufStart	86
literal (C term)		permissible activities	37
use	91	phyBuffer256	87
lithium backup battery	113, 130	PK2300	
M		components	26
memory	58	connect to PC	18
mode		device I/O map	75
program	36	establish communication	20
run	36	external dimensions	95
standalone	36	features	14
models	12	models	12
mounting		network	52
DIN rails	118	operating modes	36
end caps	118	overview	12
multidrop network	52	power-on reset	60
N		subsystems	62
network	52	PK2310	12
newBuf256	86	PK23FLSH.C	22, 23
NMI. <i>See</i> nonmaskable interrupt		ports	
nonmaskable interrupt	60	serial	16
nonvolatile storage	120	power	
		fail flag	74
		SIB2	104
		supervisor	60
		supply	16
		power failure	
		detection circuitry	112
		sequence of events	112

power supply	18	RS-232	14, 29
program		communication	50
mode	36	connector pinouts	50
output demonstration	43	RS-485	14, 29
programming	16	communication	51
programming cable	18	driver addressing	78
protected digital inputs 14, 26, 39		port	50, 51
specifications	98	termination resistors	51–53
PRT0	79	run	
PWM	14	icon	22
addressing	81	mode	36
advanced programming	86	program	22
baud rates	57	program standalone	38
how to use	54	sample program	22
outputs	54, 72	Run/Program jumper	37
specifications	54		
square waves	56		

R

real-time clock	12, 59, 74	sample program	
functions	59	AI1.C	71
internal registers	124	AO1.C	57, 73
refresh DMA counter	83	DIO1.C	67
regulated input voltage	113	input demonstration	40, 47
reset		output demonstration	42
board	74	relay	67
power-on	60	resistive measurement	71
processor	61	PK23FLSH.C	22
threshold	60	RMI	46, 71
resistive measurement input. <i>See</i>		selecting	
RMI		IN-08 and IN-09	30
resSize256	87	IN-12 through IN-16	31
return to program mode	38	OUT-04 through OUT-08	31
RJ-12 connector		RMI	30
SIB2	20, 104	RS-232	29
RJ-12 jack	14	RS-485	29
RMI	12, 14, 45, 64, 70	serial communication	50
circuit	26	channels	26
demonstration program	46	configuring	29
how to use	46	ports	14, 16, 29
sample applications	45	Serial Interface Board 2	20, 104
theory of operation	49	baud rate	104
		development	20
		dimensions	105
		power	104

S

shutdown	113
simulated EEPROM	74
sinking drivers	107
low-side drive	109
software	
feature reference	74
output channel assignments ...	66
supplied	64
source (C term)	
use	92
sourcing drivers	107
high-side drive	110
specifications	
Developers Kit	16
electrical and mechanical	94
high-current driver	100
LEDs	58
protected digital inputs	98
PWM	54
SRAM	58
standalone	
mode	36
run program	38
storage	
nonvolatile	120
subsystems	62
supplied software	64

T

termination resistor	51–53
tm_rd	59
tm_wr	59
troubleshooting	
baud rate	91
cables	90
COM port	90, 91
communication mode	91
expansion boards	90
grounds	90
operating mode	91
power supply	90
repeated resets	91

U

unregulated input voltage	113
user-programmable LEDs	58

V

voltage divider	113
-----------------------	-----

W

watchdog	74
waveform setup	81

Z

Z180	
Serial Port 1	126

Blank



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

Part No. 019-0040
Revision E