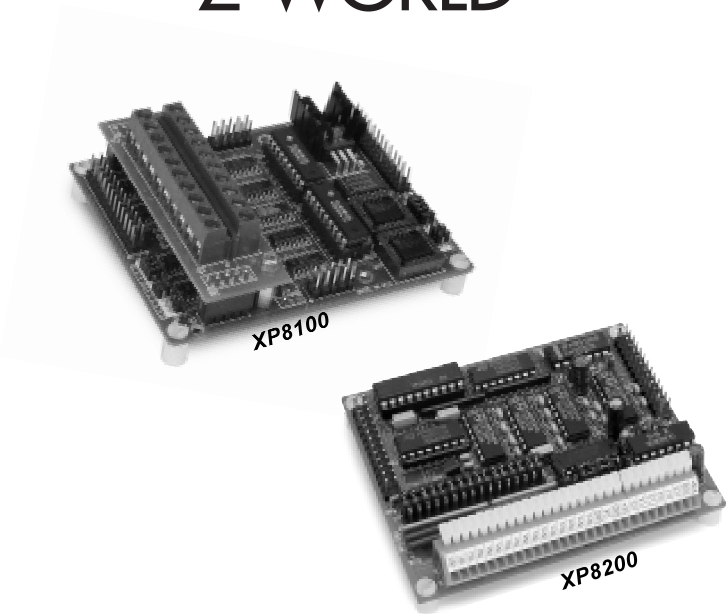
Z·WORLD



XP8100 and XP8200

Input/Output Expansion Boards

User's Manual

Revision D

XP8100 and XP8200 User's Manual

Part Number 019-0045 • Revision D
Last revised on February 7, 2000 • Printed in U.S.A.

Copyright

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

Trademarks

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

Notice to Users

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

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

All Z-World products are 100 percent functionally tested. Additional testing may include visual quality control inspections or mechanical defects analyzer inspections. Specifications are based on characterization of tested sample units rather than testing over temperature and voltage of each unit. Z-World may qualify components to operate within a range of parameters that is different 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

XP8200

ABOUT THIS MANUAL

This manual provides instructions for installing, testing, configuring, and interconnecting the Z-World XP8100 and XP8200 input/output expansion boards. Instructions are also provided for using Dynamic C® functions.

Assumptions

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

- Ability to design and engineer the target system that is controlled by a controller with analog-to-digital conversion expansion boards.
- Understanding of the basics of operating a software program and editing files under Windows on a PC.
- Knowledge of the basics of C programming.

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

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

- Knowledge of basic Z80 assembly language and architecture for controllers with a Z180 microprocessor.

  For documentation from Zilog, refer to the following texts.

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

- Knowledge of basic Intel assembly language and architecture for controllers with an Intel™386 EX processor.

  For documentation from Intel, refer to the following texts.

  Intel™386 EX Embedded Microprocessor User's Manual
  Intel™386 SX Microprocessor Programmer's Reference 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	Electronically 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
FD	Factory Default		

Conventions

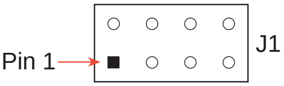
Table 3 lists and defines the typographical conventions that may be used in this manual.

Table 3. Typographical 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.
Italics	Indicates that something should be typed instead of the italicized words (e.g., in place of *filename*, 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

Blank

CHAPTER 1: OVERVIEW

Chapter 1 provides an overview description and board layout for the XP8100 Series input/output expansion boards.

XP8100 Series Overview

The XP8100 Series consists of compact input/output (I/O) expansion boards that connect to any Z-World controller supporting a Z-World PLCBus expansion port. The XP8100 Series expansion boards can more than double the digital I/O channels of a Z-World controller.

The XP8100's 32 I/O channels are configured as 16 inputs and 16 outputs. Other versions of the board are available, as indicated in Table 1-1, for added flexibility. Up to eight XP8100 boards may be linked together to provide 256 additional I/O lines.

Table 1-1. XP8100 Series Features

Model	Features
XP8100	16 protected digital inputs and 16 output drivers
XP8110	32 protected digital inputs
XP8120	32 output drivers

Because of the similarities, this manual refers to the functionality of all three XP8100 Series boards. References to all three boards will be made by referring to them as the XP8100 Series. Individual reference will be made where needed.

Connector Terminals

Three field wiring terminals (FWT) make it easy to plug and unplug wiring connections. Table 1-2 lists the FWT available for the XP8100 Series. Any of the boards in the XP8100 Series can support two FWT of any type.

Table 1-2. XP8100 Series Options

Option	Description
FWT50	Field wiring terminal with twenty 5 mm screw terminal connectors in two banks of 10 terminals each
FWT38	Field wiring terminal with 0.15 inch (3.81 mm) quick-release connectors in two banks of 10 terminals each
FWT-Opto	Field wiring terminal for inputs only, has optical isolation, uses 0.15 inch (3.81 mm) quick-release connectors in two banks of 10 terminals each

Refer to Appendix E, "field Wiring Terminals," for more information on how to use the FWT.

Outputs

The high-current outputs are capable of providing up to 500 mA, which is sufficient to drive inductive loads, relays, and other circuit-driven devices. The output drivers are socketed to allow a sourcing driver or TTL/CMOS parts to be added.

Inputs

The TTL/CMOS-compatible inputs can handle input signals between -19 and +20 volts. Input bias resistors may be user-configured to be pull-up or pull-down. Each input line is protected against transient voltages of -48 to +48 volts. A low-pass filter also blocks incoming voltage spikes.

Additional protection is possible by adding a field wiring terminal with optical isolation. See Table 1-2.

Factory Configurations

The XP8100 Series is available from the factory in three standard configurations, as listed in Table 1-1. Depending on the version, the board will have 32 channels of inputs, outputs or a combination of the two. It is not possible to change inputs to be outputs, or vice versa.

☎ For ordering information, call your Z-World Sales Representative at (530) 757-3737.

XP8100 Series Default Board Layouts

The default layouts for the XP8100, XP8110 and XP8120 expansion boards are shown in Figures 1-1, 1-2, and 1-3 for the boards as they are shipped from the factory. An outline around a particular component indicates the presence of the part in the default configuration of the board.

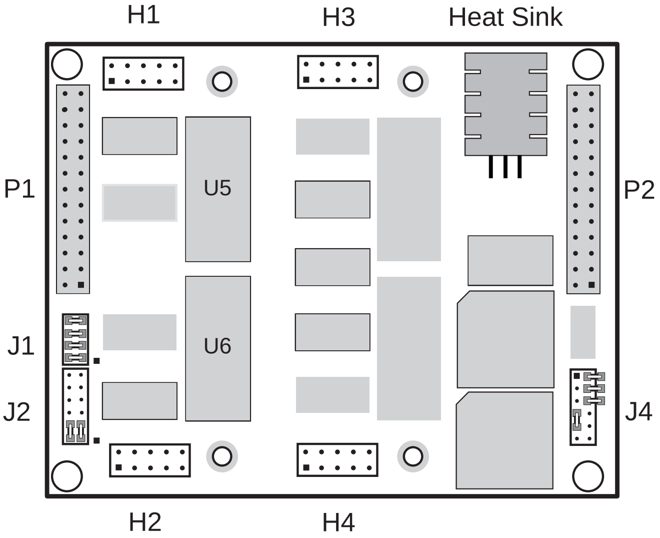


Figure 1-1. XP8100 Default Board Layout

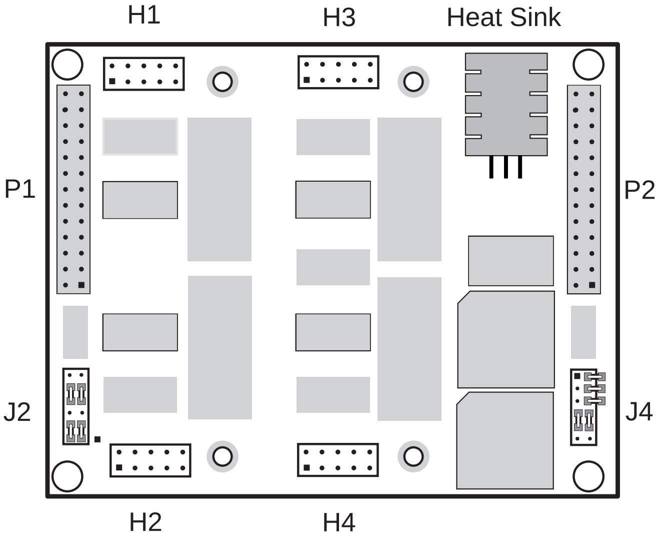


Figure 1-2. XP8110 Default Board Layout

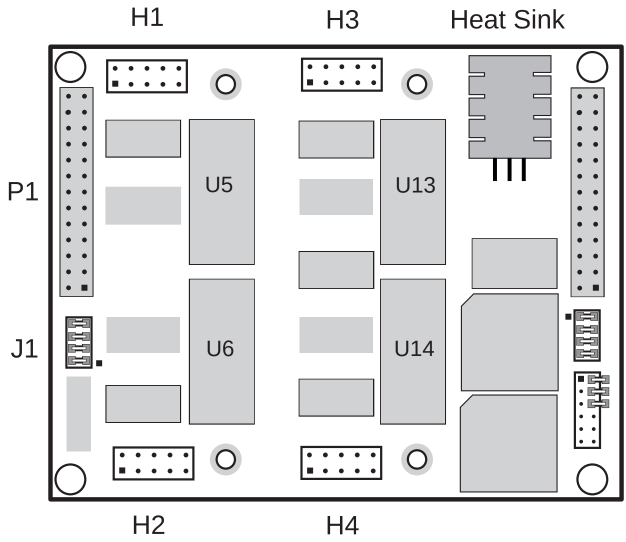


Figure 1-3. XP8120 Default Board Layout

Figure 1-4 shows the locations of the various components.

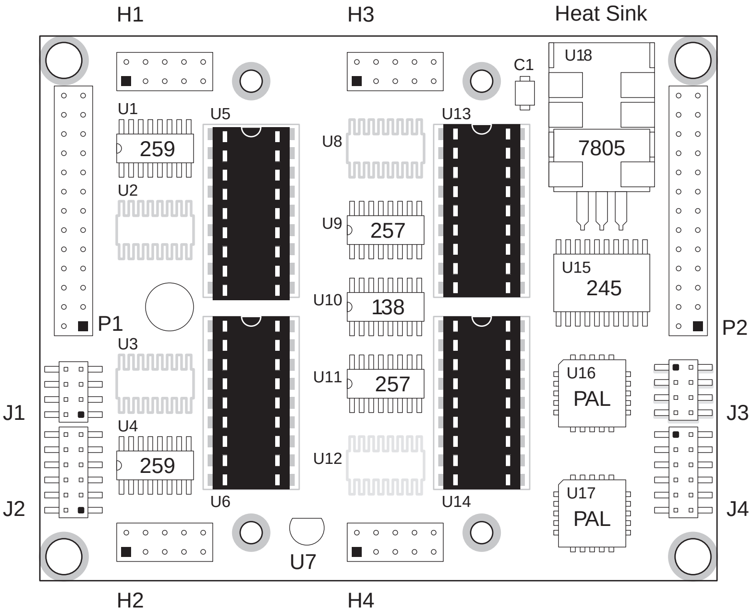


Figure 1-4. XP8100 Series Component Layout

Blank

CHAPTER 2: GETTING STARTED

Chapter 2 provides instructions for connecting XP8100 Series expansion boards to a Z-World controller. The following sections are included.

- Expansion Board Components
- Connecting Expansion Boards to a Z-World Controller
- Confirming Communications

XP8100 Series Components

The XP8100 Series boards offer protected digital inputs and high-current driver outputs. Figure 2-1 illustrates the basic layout and orientation of the expansion boards, including headers and other components. Some headers and other devices may not be present, depending on the specific board (XP8100, XP8110, or XP8120).

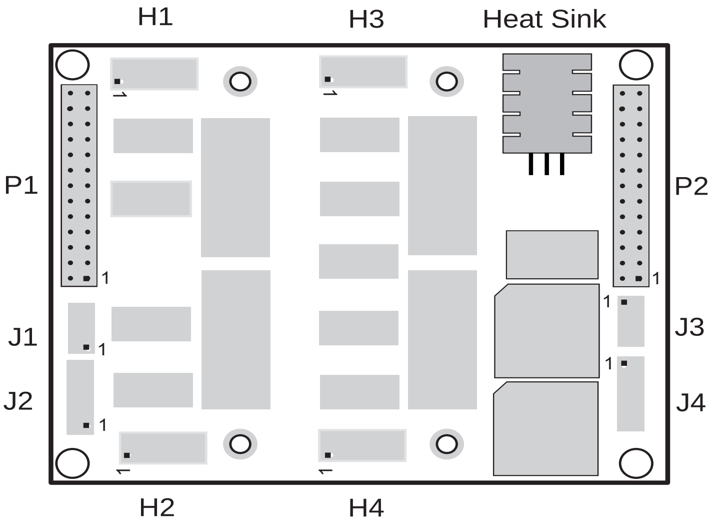


Figure 2-1. XP8100 Series Board Layout

Pay particular attention to the location of pin 1 of headers J1–J4, as indicated by a small squares in Figure 2-1. The layout orientation of J1 and J2 is opposite that of J3 and J4, so the pin 1 locations are rotated 180 degrees. Figure 2-1 is referenced throughout the manual.

See Chapter 1, “Overview,” for the exact layouts of the XP8100, XP8110 and XP8120 expansion boards.

Be careful to orient H1, H3, and the heat sink to the top, as shown in Figure 2-1, when referring to jumper and header locations.

Connecting Expansion Boards to a Z-World Controller

Use the 26-conductor ribbon cable supplied with an expansion board to connect the expansion board to the PLCBus on a Z-World controller. See Figure 2-2. The expansion board's two 26-pin PLCBus connectors, P1 and P2, are used with the ribbon cable. Z-World recommends using the cable supplied to avoid any connection problems.

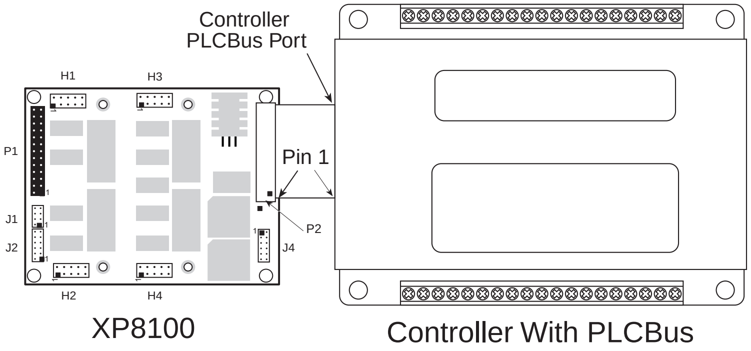


Figure 2-2. Connecting XP8100 Expansion Board to Controller PLCBus

> ⚠ Be sure power to the controller is disconnected before adding any expansion board to the PLCBus.

Follow these steps to connect an expansion board to a Z-World controller.

1. Attach the 26-pin ribbon cable to the expansion board's **P2** PLCBus header.
2. Connect the other end of the ribbon cable to the PLCBus port of the controller.

> ⚠ Be sure pin 1 of the connector cable matches up with pin 1 of both the controller and the expansion board(s).

3. If additional expansion boards are to be added, connect header **P2** on the new board to header **P1** of the board that is already connected. Lay the expansion boards side by side with headers P1 and P2 on adjacent boards close together, and make sure that all expansion boards are facing right side up.

> See Appendix C, "Connecting and Mounting Multiple Boards," for more information on connecting multiple expansion boards.

Setting Board Addresses

Z-World has established an addressing scheme for the PLCBus on its controllers to allow multiple expansion boards to be connected to the controller.

Every XP8100 Series board is shipped from the factory with a default address of 7. An XP8100 Series board may be assigned any address between 0 and 7. Jumpers are placed on the pins of header J4 to configure the board address. Figure 2-3 shows the jumper settings to set addresses 0-7.

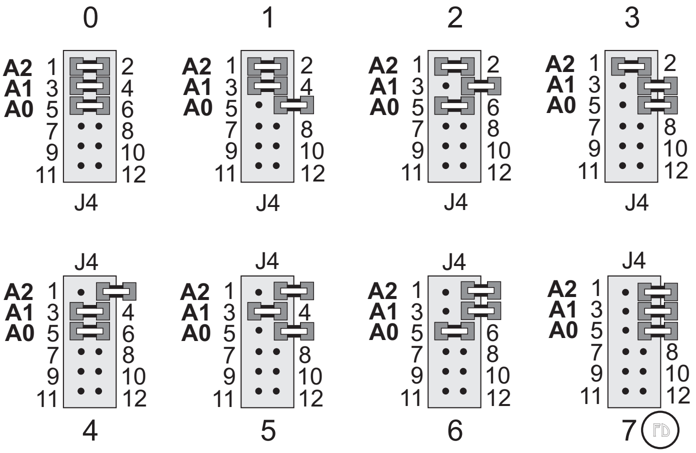


Figure 2-3. J4 Jumper Settings for XP8100 Series PLCBus Addresses

Only the first six pins of the 12-pin header J4 on the XP8100 Series are used to set the board address.

Remember that each expansion board must have a unique PLCBus address if multiple boards are to be connected. If two boards have the same address, communication problems will occur that may go undetected by the controller. A maximum of eight XP8100 boards may be addressed by a controller at one time.

Power

Z-World's expansion boards receive power from the controller over the +24 V line of the PLCBus. An onboard regulator converts this to the +5 V used by the expansion boards. The expansion boards draw about 110 mA, which means a power requirement of 1.3 W for a 12 V controller and 2.6 W for a 24 V controller.

Power may be applied to the controller once the controller and the expansion boards are properly connected using the PLCBus ribbon cable.

Confirming Communications

Run the following test program once the XP8100 Series expansion board is connected to a controller and power is applied. The sample program will confirm whether the controller and expansion board are communicating properly.

See the ***Dynamic C Technical Reference*** manual for more detailed instructions.

XP81ID.C

```
#use vdriver.lib
#use eziocmmn.lib
#use eziopbdv.lib
// uncomment #use ezioplc.lib line for PK2100(Rugged
// Giant), PK2200(Little Star), BL1200(Little PLC),
// BL1600(Little G)
//#use ezioplc.lib
// uncomment #use eziomgpl.lib line for BL1400(Micro-G)
// or BL1500(Micro-G2)
//#use eziomgpl.lib
char TITLE[] = {“XP81xx Board Detection”};
main(){
   int i;
   VdInit();
   printf(“%s\n\n”, TITLE);
   eioResetPlcBus();      // reset the PLCBus
   eioPlcRstWait();       // delay ensures the PLCBus
                          // boards reset
   // locate all possible jumper-set addresses
   // from 0 to 7 and display status
   for (i = 0; i <= 7; ++i) {
      // read to locate the board
      if (plcXP81In(i*32)==-1)
         printf(“Board %d is not located\n”,i);
      else
         printf(“Board %d is located\n”,i);
   }
}
```

Use the following steps to run the sample program.

1. Open the sample program **`XP81ID.C`** located in the Dynamic C **`samples\plcbus`** subdirectory. This program is designed to locate and display the address numbers of XP8100 Series boards connected on the PLCBus.

2. Be sure to “uncomment” the appropriate library at the top of the sample program for the particular controller being used. Do this by removing the forward slashes (**`//`**) in front of the appropriate **`#use`** library.

3. Compile the program by pressing **F3** or by choosing **Compile** from the **COMPILE** menu. Dynamic C compiles and downloads the program into the controller's memory. During compilation, Dynamic C rapidly displays several messages in the compiling window, which is normal.
4. Run the program by pressing **F9** or by choosing **Run** from the **RUN** menu.
5. The **STDIO** window will display a message once the program is running. If communication between the XP8100 Series expansion board and the controller is ok, the message will be **Board (#) is located.** If a problem exists with communications, the message will be **Board (#) is not located.** Remember that the default address is 7 for XP8100 Series expansion boards.
6. To halt the program, press **<CTRL Z>**.
7. To restart the program, press **F9**.

Check the board jumpers, PLCBus connections, and the PC/controller communications if an error message appears.

CHAPTER 3: I/O CONFIGURATIONS

Chapter 3 describes the built-in flexibility of the XP8100 Series expansion boards, and describes how to configure the available inputs/outputs. The following sections are included.

- Input/Output Pin Assignments
- Inputs
- Outputs
- Making Input/Output Connections

XP8100 Series Input/Output Pin Assignments

There are two “banks” of inputs/outputs that total up to 32 inputs/outputs for the XP8100 Series expansion boards. Bank A consists of headers H1 and H2, Bank B consists of headers H3 and H4. Figure 3-1 shows an outline of input/output Banks A and B.

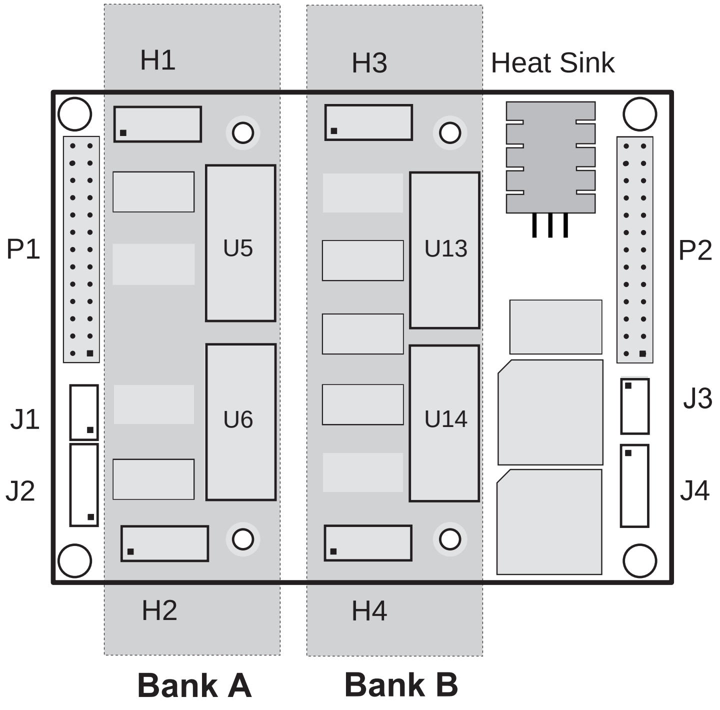


Figure 3-1. Outline of Input/Output Banks A and B

Banks A and B each have 16 input/output channels. The pins on headers H1 through H4 will function either as inputs or as outputs, depending on the specific XP8100 Series model.

Each header (H1–H4) contains a group of 10 pins. The 10 pins on each individual header function similarly to one another.

Some headers and other devices may or may not be present, depending on the specific XP8100 Series expansion board. See Chapter 1, “Overview,” for the exact board layouts.

Table 3-1 lists the functionality of the header pins for the XP8100 Series expansion boards.

Table 3-1. Header I/O Designations

	I/O Bank A		I/O Bank B	
	H1	H2	H3	H4
XP8100	8 outputs	8 outputs	8 inputs	8 inputs
XP8110	8 inputs	8 inputs	8 inputs	8 inputs
XP8120	8 outputs	8 outputs	8 outputs	8 outputs

The pin locations are different for the optional field wiring terminal (FWT) blocks described in Table 1-2. Refer to Appendix F, "Using FWT Boards," for the correct header and pin locations in these circumstances.

XP8100 Series Inputs

Protected Digital Inputs

The XP8100 and XP8110 boards are equipped with protected digital inputs designed as logical data inputs, returning a 1 or 0. The inputs accept voltages between -20 V and +24 V DC, with a logic threshold of 2.5 V DC. This means that an input returns a 0 if the input voltage is below 2.5 V, and a 1 if the input voltage is above 2.5 V DC.

A low-pass filter on each input channel has a time constant of:

T_{RC} = 220 µs at 4.5 kHz.

If the XP8100 Series board has inputs, they may be configured as "pull-up" or "pull-down" in groups of fours and eights. The configuration of each input should be determined by normal operating conditions, powerdown mode, and possible failure modes, including open or shorted conditions. These factors will influence decisions about whether to configure the inputs as "pull-up" or "pull-down."

The factory default is for inputs to be pulled up to +5 V.

Inputs may be pulled up to +5 V or pulled down to ground by configuring jumpers on headers J2 and J4.

See Figure 3-1 to help locate headers J2 and J4.

The jumpers on headers J2 and J4 configure the inputs on Bank A (H1 and H2) and Bank B (H3 and H4) as pull-up or pull-down To pull down an input from the factory default (pull-up), place a jumper across the appropriate two pins of J2 and/or J4, as shown in Figures 3-2 and 3-3 for the XP8100 and XP8110 expansion boards.

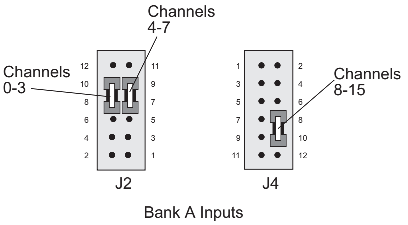
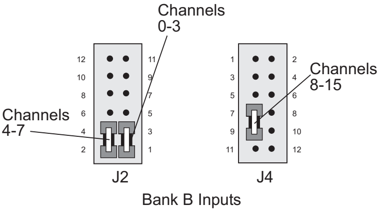
Pull-Up Configurations	Function
 Channels 4-7 Channels 0-3 12 10 8 6 4 2 11 9 7 5 3 1 1 3 5 7 9 11 2 4 6 8 10 12 Channels 8-15 J2 J4 Bank A Inputs	Note: Other jumpers may be present on J2 and J4. The J2 and J4 jumper configurations relate to Bank A inputs 0-15. FD
 Channels 0-3 12 10 8 6 4 2 11 9 7 5 3 1 Channels 4-7 1 3 5 7 9 11 2 4 6 8 10 12 Channels 8-15 J2 J4 Bank B Inputs	Note: Other jumpers may be present on J2 and J4. The J2 and J4 jumper configurations relate to Bank B inputs 0-15. FD

Figure 3-2. XP8100 Series Jumper Pull-Up Configurations

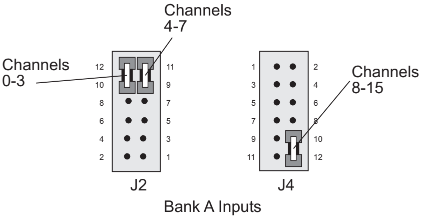
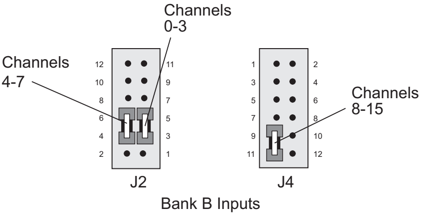
Pull-Down Configurations	Function
 Channels 4-7 Channels 0-3 12 10 8 6 4 2 11 9 7 5 3 1 1 3 5 7 9 11 2 4 6 8 10 12 Channels 8-15 J2 J4 Bank A Inputs	Note: Other jumpers may be present on J2 and J4. The J2 and J4 jumper configurations relate to Bank A inputs 0-15.
 Channels 0-3 Channels 4-7 12 10 8 6 4 2 11 9 7 5 3 1 1 3 5 7 9 11 2 4 6 8 10 12 Channels 8-15 J2 J4 Bank B Inputs	Note: Other jumpers may be present on J2 and J4. The J2 and J4 jumper configurations relate to Bank B inputs 0-15.

Figure 3-3. XP8100 Series Jumper Pull-Down Configurations

Input lines connected to opto-isolator devices must be configured as “pull-up.” Otherwise, the expansion board may be damaged.

Note that board address jumpers occupy the top three rows of header J4 (pins 1–6) as seen relative to the heat sink being at the top of the board.

XP8100 Series Outputs

The XP8100 Series expansion boards are shipped from the factory with the outputs configured with “sinking” drivers. The sinking drivers are rated up to a maximum output voltage of 48 V and a maximum current of 500 mA per individual output when only one output in a particular bank is active at once.

When all outputs are on simultaneously, thermal limits restrict the current to less than 100 mA per output. Similarly, if multiple outputs are turned on at the same time, the driver current should not exceed 350 mA per output. If the temperature exceeds 50°C, derate power dissipation by 55°C/W.

Jumpers across headers J1 and J3 define the sinking or sourcing configuration of the outputs. For the default sinking setting, the jumpers are placed horizontally across headers J1 and J3, as shown in Figure 3-4. The XP8100 uses only header J1 and the XP8120 uses both headers J1 and J3.

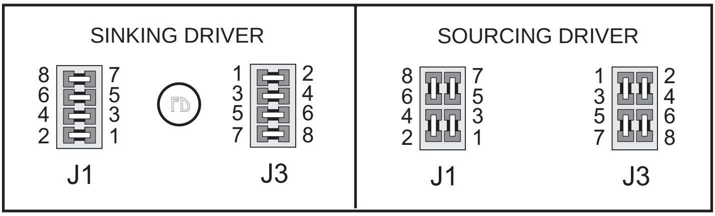


Figure 3-4. Jumper Configurations for Sinking and Sourcing Outputs

The factory default is for outputs to be configured with sinking drivers (ULN2803).

See Appendix D, “Sinking and Sourcing Drivers,” for details on using sourcing outputs.

Making XP8100 Series I/O Connections

The four 10-pin headers (H1–H4) accept either ribbon-cable connectors or up to two XP8100 Series FWT blocks for input/output connections. Input and output lines are wired to the 10-pin headers directly using a custom-built cable and connector, or by using the FWT connectors available from Z-World.

The hardware pin assignments for each header are referenced in Figure 3-5. Note that the first pin, indicated by the square, is labeled zero.

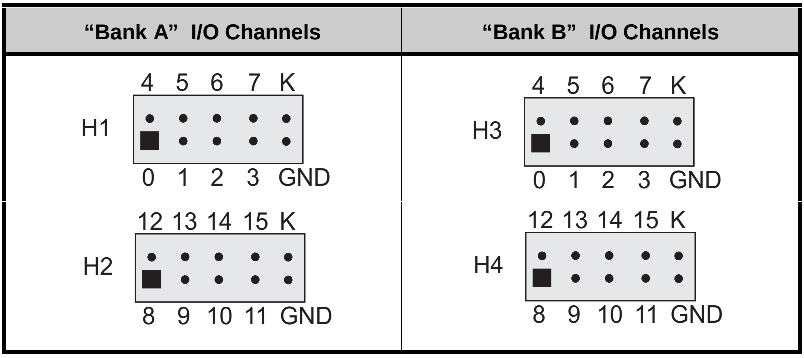


Figure 3-5. XP8100 Series Header H1–H4 Pin Assignments

Note that the hardware pin assignments for Bank B (H3 and H4) do not match up with the Bank B software input/output assignments. Both hardware and software assignments are cross-referenced in Table 4-2 in Chapter 4, "Software Reference."

Inputs/outputs may be connected with discrete wires instead of a ribbon cable. Refer to Appendix E, "Field Wiring Terminals," for information on the optional FWT connectors.

Pay close attention to the locations of pins on the header when connecting inputs/outputs.

I/O Jumper Configurations

There are four headers for jumper blocks. Depending on the specific XP8100 Series expansion board, not all the four headers may be installed on a particular board. Headers J1 and J3 are used to configure outputs, while headers J2 and J4 are used to configure inputs. Header J4 is present on all XP8100 Series expansion boards, and is used to configure inputs and address settings.

Table 3-2 lists the headers that are installed specifically for each XP8100 Series expansion board and provides a reference to the jumper configurations.

Table 3-2. XP8100 Series I/O Jumper Configurations

Header	Pins Connected	Configures
XP8100—16 inputs and 16 outputs		
J1	Sinking or sourcing drivers: see Figure 3-4	"Bank A" Output Channels 0–15
J2	Pull-up or pull-down inputs: see Figures 3-2 and 3-3	"Bank B" Input Channels 0–7
J4		"Bank B" Input Channels 8–15 and board address
XP8110—32 inputs		
J2	Pull-up or pull-down inputs: see Figures 3-2 and 3-3	"Bank A" Input Channels 0–7 and "Bank B" Input Channels 0–7
J4		"Bank A" Input Channels 8–15, "Bank B" Input Channels 8–15, and board address
XP8120—32 outputs		
J1	Sinking or sourcing drivers: see Figure 3-4	"Bank A" Output Channels 0–15
J3		"Bank B" Output Channels 0–15
J4	—	Board address only

See Figure 2-3 in Chapter 2, "Getting Started," for the jumper configurations to set board addresses.

Blank

CHAPTER 4: SOFTWARE REFERENCE

Chapter 4 describes the Dynamic C functions that initialize the XP8100 Series expansion boards, and perform input/output operations. The following major sections are included.

- Software Input/Output Channel Assignments
- Software Overview
- Digital Inputs/Outputs
- Advanced Input/Output Programming

XP8100 Series Software Input/Output Channel Assignments

Together, the four headers of Banks A and B provide a total of 32 inputs/outputs. In hardware, the input/output channels are numbered 0–15 for Bank A and are also numbered 0–15 for Bank B. However, the channels must have unique software numbers, and so the inputs/outputs for Bank A retain their numbering of 0–15, but the inputs/outputs for Bank B are numbered 16–31.

Therefore, header H1 consists of software I/O channels 0–7, header H2 consists of software I/O channels 8-15, header H3 consists of software I/O channels 16–23, and header H4 consists of software I/O channels 24–31.

See Chapter 1, "Overview," for the board layouts showing the exact locations of the headers.

Table 4-1 summarizes the software I/O assignments for each header.

Table 4-1. I/O Channel Assignments for XP8100 Series Headers

Header	Software I/O Channels
H1	0–7
H2	8–15
H3	16–23
H4	24–31

Table 4-2 lists the software I/O channel assignments for each header pin. The table details the software function number assigned to the actual hardware pin for headers H1–H4.   Refer to this table when planning which channel to activate or read during program development.

Table 4-2. I/O Channel Assignments forXP8100 Header Pins

Hardware Headers H1–H4 Pin Channel Assignment	Bank A		Bank B	
	H1 Software Channel Assignment	H2 Software Channel Assignment	H3 Software Channel Assignment	H4 Software Channel Assignment
0	0	–	16	–
1	1	–	17	–
2	2	–	18	–
3	3	–	19	–
4	4	–	20	–
5	5	–	21	–
6	6	–	22	–
7	7	–	23	–
8	–	8	–	24
9	–	9	–	25
10	–	10	–	26
11	–	11	–	27
12	–	12	–	28
13	–	13	–	29
14	–	14	–	30
15	–	15	–	31

Software Overview

This section describes a set of simple software functions to use when controlling the XP8100 Series expansion board inputs/outputs.

See the section “Advanced Programming” later in this chapter to get more information on developing applications to meet tight timing requirements.

Dynamic C Libraries

Several Dynamic C function libraries need to be used with the routines defined in this chapter. There are three common libraries used by all Z-World controllers and specific libraries designed for certain controllers. The chart in Table 4-3 identifies which libraries must be used with particular Z-World controllers.

Table 4-3. Dynamic C Libraries Required by Z-World Controllers

Library Needed	Controller
`VDRIVER.LIB`	All controllers
`EZIOCMMN.LIB`	All controllers
`EZIOPBDV.LIB`	All controllers
`EZIOTGPL.LIB`	BL1000
`EZIOLGPL.LIB`	BL1100
`EZIOMGPL.LIB`	BL1400, BL1500
`EZIOPLC.LIB`	BL1200, BL1600, PK2100, PK2200
`EZIOPLC2.LIB`	BL1700

Before using one of these libraries in an application, first include the library name in a **`#use`** command. For example, to use functions in the library **`EZIOPLC.LIB`**, be sure there is a line at the beginning of the program in the following format.

```
#use ezioplc.lib
```

Supplied Software

These Dynamic C functions are used to initialize the PLCBus. Call these functions in a program before any code to read inputs or set outputs.

- **`VdInit()`**

  Initializes the timer mechanism.

  LIBRARY: **`VDRIVER.LIB`**

- **`void eioResetPlcBus()`**

  Resets all expansion boards connected to the PLCBus.

  When using this function, initialize timers with **`VdInit()`** before resetting the PLCBus. All PLCBus devices must reset before performing any subsequent operations.

  LIBRARY: **`EZIOPLC.LIB`**

- **`void eioPlcRstWait()`**

  Provides a delay long enough for the PLCBus to reset.

  This function provides a delay of 1–2 seconds to ensure devices on the PLCBus reset. This function should be called after resetting the PLCBus.

  LIBRARY: **`EZIOPBDV.LIB`**

- **`long int eioErrorCode`**

  Represents a global bit-mapped variable whose flags reflect error occurrences.

  This register for this variable is initially set to 0. If the application tries to access an invalid channel, the flag **`EIO_NODEV`** (the first bit flag) is set in this register. Note that the other bits in **`EIO_NODEV`** deal with networked controllers.

Digital Inputs/Outputs

The following functions provide an easy way to read inputs and activate outputs. The digital input and output functions are located in the Dynamic C **`EZIOPBDV.LIB`** library.

Setting Inputs

- **`int plcXP81In( unsigned eioAddr )`**

  Reads the state of an XP8100 Series input channel.

  PARAMETER: **`eioAddr`** specifies the board address and the input pin to be read. Use the following formula in the function's argument to determine **`eioAddr`**.

  `32*brdNum+pin`

  The variable **`brdNum`** is the board address (the default address is 7, as explained in Chapter 2) and the variable **`pin`** is the input being read (software pin assignment **`0`**–**`31`**).

  RETURN VALUE:

  - 0 if the board is found and the input channel reads low.
  - 1 if the board is found and the input channel reads high.
  - Sets the flag **`EIO_NODEV`** in **`eioErrorCode`** and returns –1 if the channel does not exist (that is, if **`eioAddr`** is greater than **`31`**).

For the XP8100 board, **`eioAddr`** is a number ranging from 16 through 31. For the XP8110 board, **`eioAddr`** is a number ranging from 0 through 31.

Program 4-1 demonstrates how to read the status of a digital input.

Program 4-1. Input Demonstration Program

```
#use vdriver.lib
#use eziocmmn.lib
#use eziopbdv.lib
// uncomment #use ezioplc.lib line below for
// PK2100(Rugged Giant), PK2200(Little Star),
// BL1200(Little PLC) and BL1600(Little G)
// #use ezioplc.lib
// uncomment #use eziomgpl.lib line below for
// BL1400(Micro-G) or BL1500(Micro-G2)
// #use eziomgpl.lib
char TITLE[] = {“XP81xx Digital Input”};
main() {
   int channum;
   int i, j;
   VdInit();
   printf(“%s\n\n”, TITLE);
   eioResetPlcBus();     // reset the PLCBus
   eioPlcRstWait();      // delay ensures the
                         // PLCBus boards reset
                         // locate all possible
                         // jumper-set addresses
                         // from 0 to 7 and
                         // display status
   for (i = 0; i <= 7; ++i){
      if (plcXP81In(i*32)==-1) {   // do a read to
                                   // locate the board
         printf(“Board %d is not located\n\n”,i);
      }
      else {
         printf(“Board %d is located\n”,i);
         //read each channel from 0 to 31 and display status
         printf(“Reading all 32 positions\n”);
         for (channum = 0; channum <= 31; ++channum) {
            j = plcXP81In(i*32+channum);
            // read the input of the channel
            printf(“HV%d reads %d\n”, channum, j);
         }
      printf(“\nPress a key to continue...\n”);
      while (!kbhit());
      getchar();
      }
   }
}
```

Setting Outputs

- **`int plcXP81Out(unsigned eioAddr, int state);`**

  Writes to an output channel.

  PARAMETERS: **`eioAddr`** specifies both the board address and the output to turn on or off. Use the following formula in the function's argument to determine **`eioAddr`**.

  ```
      32*brdNum+pin
  ```

  The variable **`brdNum`** is the board address (the default address is 7, as explained in Chapter 2) and the variable **`pin`** is the output being set (software pin assignment **`0–31`**).

  `state` is 0 if the corresponding output is to be disabled or turned "OFF," **`state`** to 1 if the corresponding output is to be enabled or turned "ON."

  RETURNVALUE:

  - 0 if the output is within range.
  - Sets the flag **`EIO_NODEV`** in **`eioErrorCode`** and returns a -1 if and only if the channel does not exist (that is, if **`eioAddr`** is greater than **`31`**).

Program 4-2 demonstrates how to set the status of a digital output.

Program 4-2. Output Demonstration Program

```
#use vdriver.lib
#use vdriver.lib
#use eziocmmn.lib
#use eziopbdv.lib
#use ezioplc.lib
// uncomment #use ezioplc.lib line below for
// PK2100(Rugged Giant), PK2200(Little Star),
// BL1200(Little PLC) and BL1600(Little G)
// #use ezioplc.lib
// uncomment #use eziomgpl.lib line below for
// BL1400(Micro-G) or BL1500(Micro-G2)
// #use eziomgpl.lib
char TITLE[] = {“XP81xx Digital Output”};
main() {
   int channum;
   int i;
   VdInit();
   printf(“%s\n\n”, TITLE);
   eioResetPlcBus();     // reset the PLCBus
   eioPlcRstWait();      // delay ensures the
                         // PLCBus boards reset
                         // locate all possible
                         // jumper-set addresses
                         // from 0-7 and display
                         // status
   for (i = 0; i <= 7; ++i) {
      if (plcXP81In(i*32)==-1) { //read to locate board
         printf(“Board %d is not located\n\n”,i);
      }
      else {
         printf(“Board %d is located\n”,i);
                         // enable each chan from 0-31
         printf(“Enabling all 32 positions\n”);
         or (channum = 0; channum <= 31; ++channum)
         plcXP81Out(i*32+channum,1); //wr state to out chan
         printf(“Press a key to continue...\n”);
         while (!kbhit());
         getchar();      // disable each chan from 0-31
         printf(“Disabling all 32 positions\n”);
         for (channum = 0; channum <= 31; ++channum)
         plcXP81Out(i*32+channum,0); //wr state to out chan
         printf(“Press a key to continue...\n”);
         while (!kbhit());
         getchar();
      }
      printf(“\n”);
   }
}
```

Advanced Programming

While the functions described in the last four pages are easy to use to read and set input/output channels, they may not be able to meet the requirements of critical, real-time applications. This section discusses how to access the inputs/outputs on the XP8100 Series expansion boards more efficiently. To this, the reader must be familiar with binary arithmetic, C programming, and low-level PLCBus operations.

Functions for PLCBus Cycles, Reading and Writing

The PLCBus functions described in this section for the XP8100 Series expansion boards will make a program more abstract and portable. Dynamic C's **`inport`** and **`outport`** statements or **`in`** and **`out`** assembly instructions may still be used for controllers that support the PLCBus directly. However, the expansion boards still have to be reset and a delay has to be provided to ensure that all resets have occurred.

The following functions are located in **`EZIOPBDV.LIB`**.

- **`unsigned _eioPlcXP81Addr(char BrdAddr)`**

  Converts the logical address into a 12-bit physical address.

  PARAMETER: **`BrdAddr`** is the jumper-configured board address, which ranges from 0 to 7. The logical address of the XP8100 Series expansion boards is **`0000 0pqr`**, where **`pqr`** is the binary representation for a board address of 0 to 7.

  The function converts the logical address into a 12-bit physical address, **`r000 01pq 0001`**.

  RETURN VALUE: The bit-mingled XP8100 Series physical address.

The following functions are located in **`EZIOPLC.LIB`** and can be used to simplify the multiple writes and reads on the PLCBus.

- **`void eioPlcAdr12(unsigned addr)`**

  Specifies an address on the PLCBus using the BUSADR0, BUSADR1, and BUSADR2 cycles. **`addr`** is broken into three nibbles, and one nibble is written during each BUSADRx cycle, with BUSADR0 the first bus cycle.

  `addr` contains the PLCBus cycle addresses. BUSADR0 contains the least significant four bits as shown below.

`addr`:	**`0000`**	**`rxyz`**	**`01pq`**	**`0001`**
BUSxxxx:.		ADR2	ADR1	ADR0

- **`void eioPlcAdr4(unsigned addr)`**

  Writes to PLCBus register BUSADR2.

  `addr` is the most significant four bits, **`rxyz`**. Here **`xyz`** is represented as a group number. This function writes **`rxyz`** only in register BUSADR2. Table 4-4 on the next page lists the **`rxyz`** addresses.

- **`char _eioReadD0()`**

  This function reads the BUSRD0 register and returns the four data bits D3–D0 read off the PLCBus.

- **`char _eioReadD1()`**

  This function reads the BUSRD1 register and returns the four data bits D3–D0 read off the PLCBus.

- **`void _eioWriteWR(char ch)`**

  This function writes to the BUSWR register.

  `ch` is the four data bits, D3–D0, written in the BUSWR register.

Address Calculation

Addressing an XP8100 Series expansion board first involves explicitly determining each bit of the board's address and then arranging those bits in a particular order. This form of addressing is more complex than the simple formula presented in the preceding section.

Let **`p`**, **`q`**, and **`r`** represent the most significant to least significant bits of the jumper-set address of an XP8100 Series expansion board. The "logical" address of each board in binary notation is then **`0000 0000 0pqr`**. The default address of any XP8100 Series expansion board is 7, as explained in Chapter 2.

The actual address that is passed to advanced PLCBus functions, however, must be rearranged to a physical address, **`rxyz 01pq 0001`**, where **`xyz`** corresponds to either the board identification address or a group number for the input/output data. The physical address is passed during a PLCBus cycle by presenting the least-significant nibble, **`0001`**, to the BUSADR0 register, the middle nibble **`01pq`** to the BUSADR1 register, and the most-significant nibble **`rxyz`** to the BUSADR2 register. Table 4-4 on the next page lists the **`rxyz`** addresses.

For convenience, the function **`_eioPlcXP81Addr`** described in the previous section is available to transform the logical address into the physical address **`r000 01pq 0001`** required by the PLCBus.

Table 4-4 lists the software input/output group numbers and the corresponding register BUSADR2 values to use when accessing the XP8100's input channels via PLCBus registers BUSRD0 and BUSRD1. The bit positions of all 32 channels are also included. The input/output channels are shown as channels 00 through 31.

Table 4-4. Software Input Registers

			n (I/O Channels)			
Group Number	**BUSADR2 rxyz**	**PLCBus Register**	**D3**	**D2**	**D1**	**D0**
0	r000	BUSRD0	X	X	X	0
0	r100	BUSRD0	03	02	01	00
		BUSRD1	07	06	05	04
1	r101	BUSRD0	11	10	09	08
		BUSRD1	15	14	13	12
2	r110	BUSRD0	19	18	17	16
		BUSRD1	23	22	21	20
3	r111	BUSRD0	27	26	25	24
		BUSRD1	31	30	29	28

Checking for Presence of XP8100 Using Dynamic C Functions

It is possible to verify whether an XP8100 Series expansion board with a given bus address is actually responding. If the program addresses an XP8100 with the lowest three bits of the highest nibble cleared, then the XP8100 at that address will enter an "ID mode." A correctly identified board in the ID mode responds with a nibble that has the least significant bit cleared.

Use the following procedure and sample program with the Dynamic C functions to check whether a board actually exists on the PLCBus.

1. Calculate the physical PLCBus address of the board using the function

   `_eioPlcXP81Addr`.

   The address will automatically be in "ID mode" in the form **`r000 01pq 0001`**. Remember that **`pqr`** is the jumper-configured board address, as explained in Chapter 2.

2. Send the physical address to the PLCBus using the function

   `eioPlcAdr12`.

3. Read back the nibble D3-D0 using the function

   `eioReadD0`.

4. Determine whether a board exists on the PLCBus by checking if the least significant bit D0 is cleared or contains a zero. Refer to Table 4-4 to help determine D0.

Program 4-3, **`XP81IDX.C`**, shows how to detect XP8100 expansion boards connected on the PLCBus using Dynamic C functions. Compile and run this program from the Dynamic C **`SAMPLES\PLCBUS`** subdirectory.

Program 4-3. Board Detection Program

XP81IDX.C

```
#use vdriver.lib
#use eziocmmn.lib
#use eziopbdv.lib
#use ezioplc.lib
   // for PK2100(Rugged Giant), PK2200(Little Star),
   // BL1200(Little PLC), BL1600(Little G)
   //#use eziomgpl.lib
   // for BL1400(Micro-G) or BL1500(Micro-G2)
main(){
   int i;
   int brdAdr;
   VdInit();              // auto hit watch dog
   eioResetPlcBus();      // reset PLCBus
   eioPlcRstWait();  // delay ensures PLCBus boards reset
// locate all possible jumper-set board addresses from 0 to 7
   for (i = 0; i <= 7; ++i) {
      brdAdr = _eioPlcXP81Addr(i);
                          // convert to PLCBus format
   eioPlcAdr12(brdAdr);   // send board address
   if (_eioReadD0()&1)    // read nibble, mask bit 0
      printf(“Board %d is not located\n”,i);
   else
      printf(“Board %d is located\n”,i);
   }
}
```

Checking for Presence of XP8100 Without Using Dynamic C Functions

The following steps may be used to check whether a board is connected to the PLCBus without using Dynamic C functions. The procedure requires accessing the BUSADR0 and BUSADR1 registers during a PLCBus cycle. The procedure essentially checks if a board with a specific address exists.

1. The physical address is always in the form `rxyz 01pq 0001.`

   The letters `pqr` stand for the board address, which is 0 to 7, in binary notation. The letters `xyz` will always be `000` for "ID mode." Thus, the string becomes `0000 0101 0001` for a board address of 2 since the binary notation for 2 is `010`.

2. First write the nibble `0001` in the BUSADR0 register during a PLCBus cycle.
3. Write `01pq` in the BUSADR1 register.
4. Write `r000` in the BUSADR2 register (remember `xyz` is always `000` for ID mode).
5. Read back the data bits D3–D0 from the BUSRD0 register as `n`.
6. Determine if the least significant bit 0 (D0) of `n` is cleared. One method of checking bit 0 is to mask `n` by performing a "logical and 1" of `n`. If the result is zero, the XP8100 board is present.
7. At this point, repeat Steps 3-6 to check for another board only if the BUSADR0 register has not been accessed, and use an address number that is different from the one just checked. Then, change `pqr` to identify the next board address.

See Appendix G, "PLCBus States," for detailed states and transitions for the PLCBus. These will be useful for advanced programming.

Reading an Input State Using Dynamic C Functions

The specific XP8100 Series expansion board will determine how many inputs are available, if any.

See the board layouts in Chapter 1, “Overview,” to determine which XP8100 Series expansion board is actually being used.

The XP8100 Series of expansion boards is a 4-bit PLCBus device. Each read register can return up to four bits during a cycle. There are two read registers, BUSRD0 and BUSRD1.

The XP8100 Series input channels are organized into four groups, and each group has eight individual channels. Group 0 corresponds to I/O channels 0–7, Group 1 corresponds to channels 8–15, Group 2 corresponds to channels 16–23, and Group 3 corresponds to channels 24–31.

Use the following procedure when reading an input state to first select the proper group of inputs and then read the state of that group’s inputs.

1. Use the function **`eioPlcXP81Addr`** to calculate the physical PLCBus board address (**`r000 01pq 0001`**).
2. Use the function **`eioPlcAdr12`** to send the physical address to the PLCBus.
3. Use the function **`eioPlcAdr4`** to send the selected group number **`rxyz`**. Table 4-4 provides the group numbers for the I/O channels.
4. Use either function **`_eioReadD0`** or **`eioReadD1`**, depending on the I/O channel number, to read the nibble D3–D0.
5. Determine if a board exists on the PLCBus by checking if the I/O channel number and corresponding bit position contains a one. Refer to Table 4-4 for corresponding bit positions D3–D0.
6. At this point, the program may do one of the following.
    - Go to Step 1 to select another board
    - Go to Step 3 to select another group on the same board
    - Go to Step 4 to read from the same channel group

The sample program **`XP81INX.C`** demonstrates how to read inputs using the Dynamic C functions supplied. Compile and run this program from the Dynamic C **`SAMPLES\PLCBUS`** subdirectory.

Reading an Input State Without Using Dynamic C Functions

The following steps demonstrate how to operate the PLCBus to read an input without using the supplied Dynamic C functions.

See Appendix G, “PLCBus States,” for detailed states and transitions for the PLCBus. These will be useful for advanced programming.

1. Refer to Table 4-4 for the register and channel assignments.
2. The physical address must be in the format

   `rxyz 01pq 0001`.

   The board’s address is represented in binary notation as **`pqr`**. The group number is **`xyz`**.
3. First write the nibble **`0001`** in register BUSADR0 during a PLCBus read cycle.
4. Write **`01pq`** in register BUSADR1.
5. Write **`rxyz`** in register BUSADR2.
6. Read back the data bits from the proper register (BUSRD0 and BUSRD1) as **`n`**.
7. Determine if a board exists on the PLCBus by checking if the channel number and corresponding bit position contain a one.
8. The program may now do one of the following if the BUSADR0 read cycle has not been accessed using a **`0001`**.
   - Go to Step 3 to select another board
   - Go to Step 4 to select another group on same board
   - Go to Step 5 to read from the same group.

Controlling Outputs Using Dynamic C Functions

Controlling outputs using Dynamic C functions is similar to the procedure for reading an input's state. The procedure for writing an output also considers the XP8100 Series expansion board to have four groups of input/output channels, with each group having eight channels.

However, the output write procedure deals with only one channel for each PLCBus cycle, unlike the input procedure which handles four input channels during each PLCBus cycle.

Table 4-5 lists which PLCBus address to use when accessing a group of eight channels via the PLCBus BUSWR register.

Table 4-5. Software Output Registers

					BUSWR			
BUSADR2					**n**			
Group Number	**0**	**1**	**2**	**3**	**Channel Data**			**state 0=off, 1=on**
`rxyz`	`r100`	`r101`	`r110`	`r111`	**D3**	**D2**	**D1**	**D0**
Output Channel	00	08	16	24	0	0	0	0
					0	0	0	1
	01	09	17	25	0	0	1	0
					0	0	1	1
	02	10	18	26	0	1	0	0
					0	1	0	1
	03	11	19	27	0	1	1	0
					0	1	1	1
	04	12	20	28	1	0	0	0
					1	0	0	1
	05	13	21	29	1	0	1	0
					1	0	1	1
	06	14	22	30	1	1	0	0
					1	1	0	1
	07	15	23	31	1	1	1	0
					1	1	1	1

The following procedure first selects the proper group of outputs and then writes the state to the group's output channel.

1. Use the function `_eioPlcXP81Addr` to calculate the physical address `r000 01pq 0001`.
2. Use the function `eioPlcAdr12` to send the physical address to the PLCBus.
3. Use the function `eioPlcAdr4` to send the selected group number `rxyz`. Table 4-5 lists the output registers for the I/O channel group numbers.
4. Use `_eioWriteWr` to send the output state D3–D0. Table 4-5 lists the output registers for the corresponding bit positions D3–D0 and channel numbers.
5. At this point, the program may do one of the following.
    - Go to Step 1 to select another board
    - Go to Step 3 to select another group on same board
    - Go to Step 4 to write to the same group.

Controlling Outputs Without Using Dynamic C Functions

The following steps demonstrate how to perform the PLCBus operation of setting an output without using the supplied Dynamic C functions. Refer to Table 4-5 for the register, channel and group number assignments.

See Appendix G, "PLCBus States," for detailed states and transitions for the PLCBus. These will be useful for advanced programming.

1. The physical address must be in the format `rxyz 01pq 0001`. The board's address is `pqr` and the group number is `xyz`.
2. First write the nibble `0001` in register BUSADR0 during a PLCBus cycle.
3. Write `01pq` in register BUSADR1.
4. Write `rxyz` in register BUSADR2.
5. To turn the output channel on, write the data bits D3–D0 to the BUSWR register. Refer to Table 4-5 to find the corresponding bit positions D3–D0.
6. The program may do one of the following if the BUSADR0 cycle has not been accessed using a `0001`.
    - Go to Step 3 to select another board
    - Go to Step 4 to select another group on the same board
    - Go to Step 5 to write to an output of the same group.

XP8200

Blank

CHAPTER 5: OVERVIEW

Chapter 5 provides overview description and board layout for the XP8200 input/output expansion boards.

XP8200 Universal Input/Output Board

The XP8200 expansion board has 16 "universal" inputs and six high-current digital outputs. The "universal" inputs are configurable to function in three different ways.

1. Digital inputs with a single, fixed threshold.
2. Digital inputs with two or more adjustable thresholds.
3. Comparator (analog) inputs.

An external voltage source is needed if the "universal" inputs are configured as digital inputs with two or more thresholds, or as comparator inputs.

The high-current digital outputs are suitable for driving relays or small actuators. Diodes protect the drivers against spikes.

The XP8200 can be connected to the PLCBus with other expansion boards. Up to 32 XP8200s can be addressed on a single bus.

XP8200 Board Layout

Figure 5-1 shows the layout for the XP8200 expansion board.

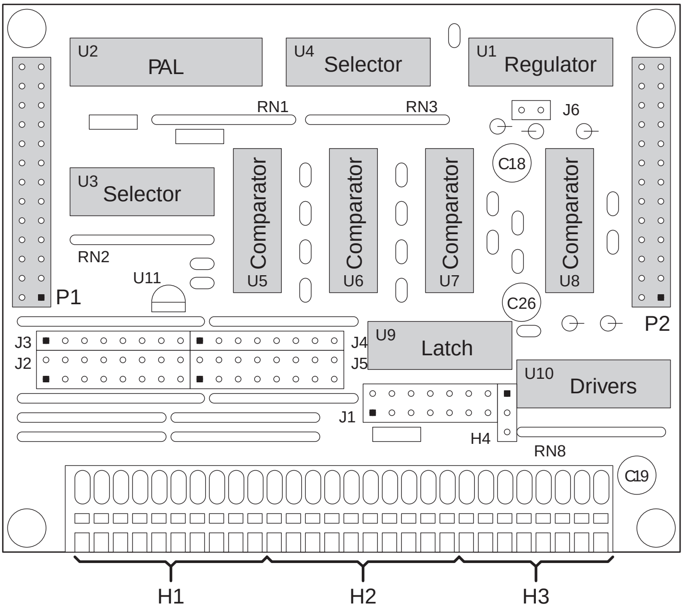


Figure 5-1. XP8200 Board Layout

Chapter 6: Getting Started

Chapter 2 provides instructions for connecting XP8200 expansion boards to a Z-World controller. The following sections are included.

- Expansion Board Components
- Connecting Expansion Boards to a Z-World Controller
- Confirming Communications

XP8200 Components

The XP8200 expansion board offers 16 "universal" inputs and six digital outputs. Figure 6-1 shows the basic layout and orientation of components, jumper blocks, and connectors.

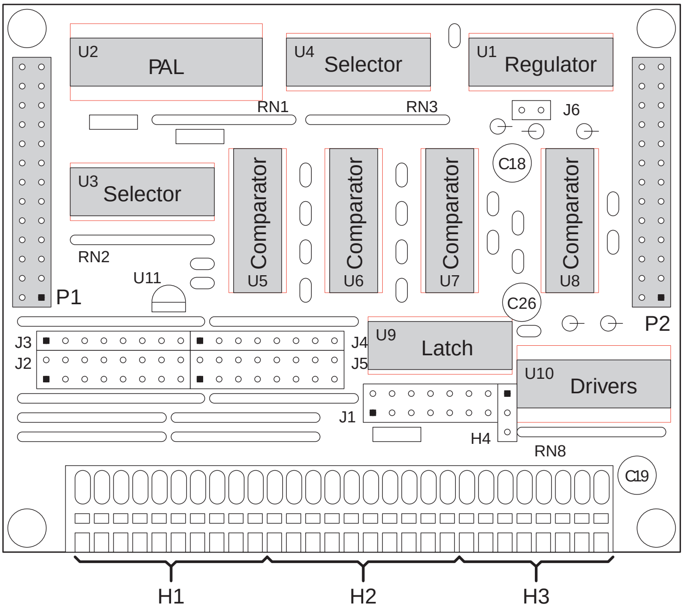


Figure 6-1. XP8200 Board Layout

Connecting Expansion Boards to a Z-World Controller

Use the 26-conductor ribbon cable supplied with an expansion board to connect the expansion board to the PLCBus on a Z-World controller. See Figure 6-2. The expansion board's two 26-pin PLCBus connectors, P1 and P2, are used with the ribbon cable. Z-World recommends using the cable supplied to avoid any connection problems.

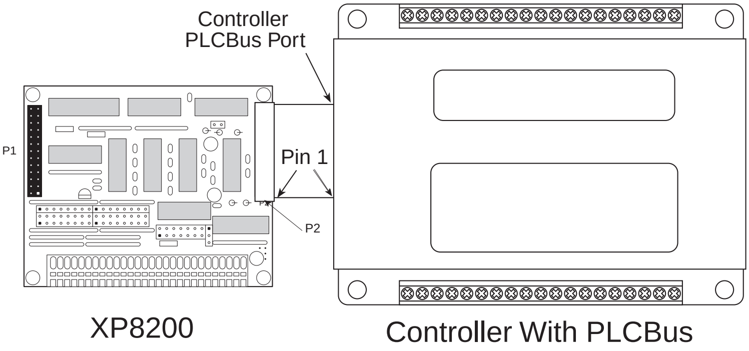


Figure 6-2. Connecting XP8100 Expansion Board to Controller PLCBus

> ⚠ Be sure power to the controller is disconnected before adding any expansion board to the PLCBus.

Follow these steps to connect an expansion board to a Z-World controller.

1. Attach the 26-pin ribbon cable to the expansion board's **P2** PLCBus header.
2. Connect the other end of the ribbon cable to the PLCBus port of the controller.

> ⚠ Be sure pin 1 of the connector cable matches up with pin 1 of both the controller and the expansion board(s).

3. If additional expansion boards are to be added, connect header **P2** on the new board to header **P1** of the board that is already connected. Lay the expansion boards side by side with headers P1 and P2 on adjacent boards close together, and make sure that all expansion boards are facing right side up.

See Appendix C, "Connecting and Mounting Multiple Boards," for more information on connecting multiple expansion boards.

Setting Board Addresses

XP8200 expansion boards are shipped from the factory with pins 1 and 2 on header J1 connected. Each XP8200 can have one of 16 PALs. Therefore, up to 32 addresses are possible, depending on whether pins 1 and 2 are connected.

Remember that each expansion board must have a unique PLCBus address if multiple boards are to be connected. If two boards have the same address, communication problems will occur that may go undetected by the controller.

Power

Z-World's expansion boards receive power from the controller over the +24 V line of the PLCBus. An onboard regulator converts this to the +5 V used by the expansion boards. The expansion boards draw about 110 mA, which means a power requirement of 1.3 W for a 12 V controller and 2.6 W for a 24 V controller.

Power may be applied to the controller once the controller and the expansion boards are properly connected using the PLCBus ribbon cable.

XP8200 expansion boards are normally configured for 24 V, with the jumper on header J6 not connected. A factory modification to header J6 configures the XP8200 for use with 12 V controllers.

☎ XP8200 expansion boards are available from the factory with header J6 configured for 12 V. For ordering information, call your Z-World Sales Representative at (530) 757-3737.

Confirming Communications

Run a sample program once the XP8200 expansion board is connected to a controller and power is applied. The sample program will confirm whether the controller and expansion board are communicating properly.

The sample programs **`UIEXPIO.C`** and **`UIVDVR.C`**, found in the Dynamic C **`SAMPLES\PLCBUS`** subdirectory, are functionally equivalent. However, **`UIVDVR.C`** uses the virtual driver, while **`UIEXPIO.C`** uses direct drivers. When these sample programs are used, the analog and universal inputs work only with the PK2100 controller.

The program prints a message, then waits for a character typed at the PC. Depending on the character, the program turns on all the drivers, turns them off, reads the inputs as digital inputs or reads the inputs as analog inputs.

The following instructions apply to either program.

1. Power up the controller and make sure it is working properly. If there are any problems, consult the controller's technical reference manual. Now disconnect power from the controller.
2. Connect the XP8200 to the controller using the PLCBus ribbon cable.
3. If a PK2100 controller is used, connect the UEXP terminal on the PK2100 to the DA terminal on the XP8200.
4. Check jumper J1 on the XP8200. Make sure pins 1-2 are connected.
5. Power up the controller and start Dynamic C on the PC. Open and run the sample program.
6. Press **F4** to enable the keyboard input. Type "a," "b," "c," or "d" a few times. Observe the result.
7. Now, change one or more of the input jumpers on the large header comprising J2–J5. (These connect inputs either to the excitation voltage or to ground.) Observe the changes when "c" or "d" is keyed.
8. Last, change the jumper setting on H4. (This switches the negative input to the comparators between VL and the external DAC source.) Observe the changes (if any) when "d" is keyed.

Check the board jumpers, PLCBus connections, and the PC/controller communications if an error message appears.

See the ***Dynamic C Technical Reference*** manual for more detailed instructions.

Blank

Chapter 7: I/O Configurations

Chapter 7 describes the built-in flexibility of the XP8200 expansion boards, and describes how to configure the available inputs/outputs. The following sections are included.

- Input/Output Pin Assignments
- Inputs
- Outputs
- Making Input/Output Connections

XP8200 Input/Output Pin Assignments

External Connections

The universal inputs (IN0–IN15) appear on Wago connectors H1 and H2. Driver outputs (/L0–/L5) appear on connector H3. The three connectors look like one long header, as shown in Figure 7-1.

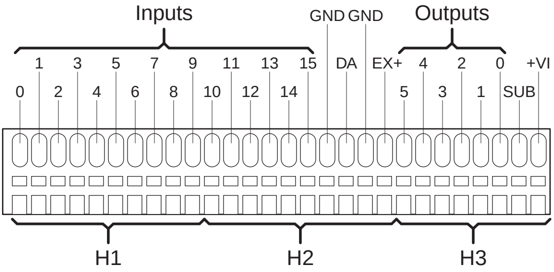

Inputs
GND GND
Outputs
1 3 5 7 9 11 13 15 DA EX+ 4 2 0 +VI
0 2 4 6 8 10 12 14 5 3 1 SUB
H1
H2
H3

Figure 3-6. XP8200 I/O Headers

Table 7-1 summarizes the functions of the connectors on these terminals.

Table 7-1. Description of I/O Pins on Headers H1, H2, and H3

Pins	Description
IN0–IN15	Universal inputs
GND	Board ground
DA	External voltage source (presumed to be DAC)
EX+	External excitation voltage
/L0–/L5	High voltage driver outputs
VIN+	Load supply voltage for devices driven by /L0 through /L5
SUB	Load supply ground

Jumpers

The XP8200 uses seven headers, J1 through J6, and H4. Pins 1 and 2 of header J1, coupled with the encoding of the PAL chip U2, determine the board's PLCBus address, as explained in Chapter 8 and in Appendix D, "Board Layouts." Figure 7-2 shows the signals on header J1.

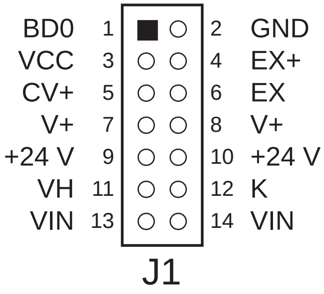


Figure 7-2. Header J1 Pinout

Table 7-2 lists the jumper settings for header J1.

Table 7-2. Jumper Settings for Header J1

Pins	Description
1–2	Least significant bit of board address (1 if not connected)
3–5	Comparator voltage CV+ connected to VCC (+5 V)
5–7	Comparator voltage CV+ connected to V+ (default)
4–6	Reference voltage connected to external excitation voltage (EX+)
6–8	Reference voltage connected to V+ (default)
9–11	Input voltage to regulator (VH) is +24 V from PLCBus (default)
11–13	Input voltage to regulator (VH) is VIN from external supply
10–12	Common protective diode K connected to +24 V from PLCBus
12–14	Common protective diode K connected to VIN from external supply (default)

When the jumper on header J6 is connected (it has only two pins), V+ is 7 V. Otherwise, V+ is 14 V.

Use the connections shown in Figure 7-3 for header H4 to select whether the inputs are digital or "universal."

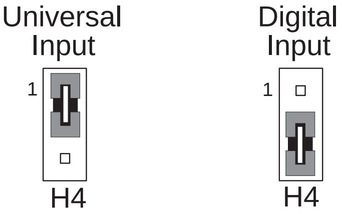


Figure 7-3. Header H4 Connections

Headers J2, J3, J4 and J5 comprise one large header for connecting the universal inputs. There are 16 columns on this header, one for each of the 16 input channels. There are three pins in each column. Connecting the upper two pins pulls a channel's input to ground. Connecting the bottom two pins in a column pulls that channel's input up to the excitation voltage, as shown in Figure 7-4.

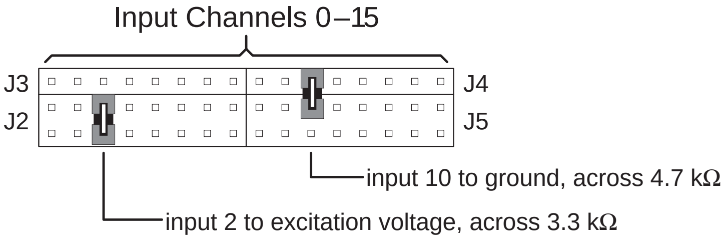

Input Channels 0–15
J3
J2
J4
J5
input 10 to ground, across 4.7 kΩ
input 2 to excitation voltage, across 3.3 kΩ

Figure 7-4. XP8200 I/O Connections on Headers J2, J3, J4, and J5

XP8200 Inputs

"Universal" Inputs

The XP8200's 16 inputs (IN0–IN15 at the connector) are fed to the positive side of 16 comparators, as shown in Figure 7-5. An external analog voltage source (DA) or a fixed threshold VL drives the negative side of the comparators.

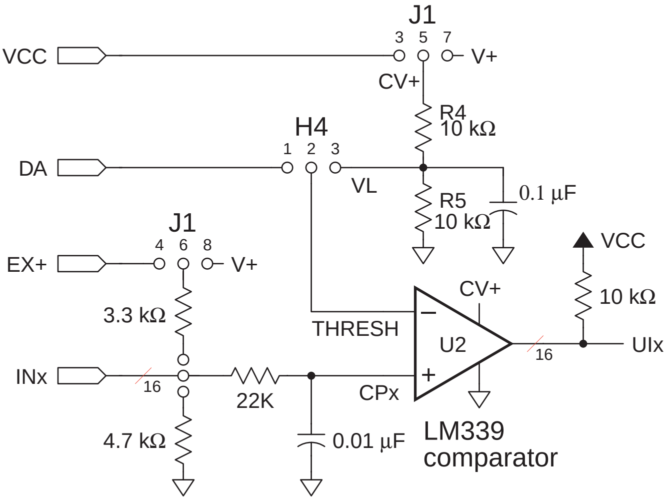


Figure 7-5. XP8200 Inputs

Each input channel can be pulled high to an excitation voltage (which can be external or V+) or pulled to ground, depending on the jumper settings on headers J2, J3, J4 and J5. The input feeds to the positive side of a comparator. The negative side of the comparator is connected to an external voltage source (DA) or to VL from the onboard regulator by setting header H4.

Digital Inputs

When the negative side of the comparator is connected to VL, the inputs function as digital inputs: the channel reads “1” when the input voltage is more positive than its threshold voltage, and “0” otherwise. The threshold voltage is given by Equation (7-1).

$$V_{THRESHOLD} = (CV +) \times \left(\frac{R_5}{R_4 + R_5} \right) \tag{7-1}$$

Digital Inputs with Two Thresholds

The input channels can be used as analog channels or as digital channels with two (or more) thresholds specified in software. The negative side of the comparator is connected to an external analog voltage source (such as the PK2100’s DAC output) on the controller.

To use the channels as digital inputs, Z-World recommends using the ***virtual driver*** to sample the inputs periodically. When an input voltage is higher than its upper threshold, the channel becomes “1,” and when it goes below its lower threshold, the channel becomes “0.” The digital result does not change (in software) when the input is between these two thresholds. This provides stability for some kinds of inputs.

Analog Inputs

The input channels can also be used as analog inputs when the negative side of the comparator is connected to an external DAC. It is even possible to take an analog reading by successive approximation because the comparator threshold can be varied. The accuracy of such a reading depends, in part, on the accuracy of the DAC output.

Call the function **`_uiAnalog(channel)`** when using the inputs as analog inputs. This function handles all the details.

Analog inputs are supported only on the PK2100 controller.

XP8200 Outputs

Six high-current transistor outputs (/LO–/L5) drive external loads such as relays or small actuators. Figure 7-6 shows an XP8200 output.

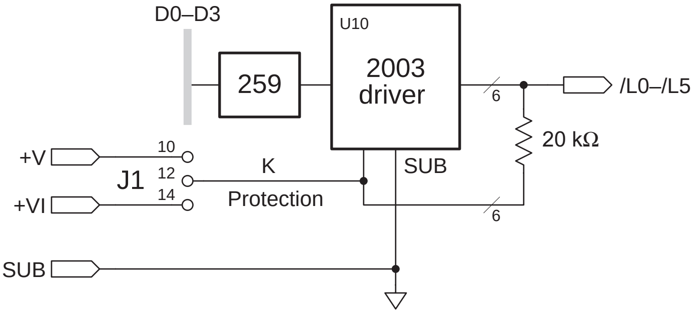


Figure 7-6. XP8200 Outputs

The outputs can either be supplied +24 V from the PLCBus or an external voltage (VIN) from header H3. Connect SUB, also on H3, to external ground when using an external voltage. Diodes in the 2003 driver protect the driving transistors against spikes.

The driver chip dissipates up to 1.25 W at 60°C. With six channels on at the same time, the maximum current is ~160 mA per channel. The maximum current per channel is 500 mA.

If incandescent lights are being driven by the outputs, place a resistor in series with the load to limit the initial inrush of current.

Blank

CHAPTER 8: SOFTWARE REFERENCE

Chapter 8 describes the Dynamic C functions that initialize the XP8200 expansion boards, and perform input/output operations. The following major sections are included.

- Software Input/Output Channel Assignments
- Software Overview
- Digital Inputs/Outputs
- Advanced Input/Output Programming
- Operating the XP8200

Software Overview

This section describes a set of simple software functions to use when controlling the XP8200 expansion board inputs/outputs.

See the section “Advanced Programming” later in this chapter to get more information on developing applications to meet tight timing requirements.

Dynamic C Libraries

Several Dynamic C function libraries need to be used with the routines defined in this chapter. There are three common libraries used by all Z-World controllers and specific libraries designed for certain controllers. The chart in Table 8-1 identifies which libraries must be used with particular Z-World controllers.

Table 8-1. Dynamic C Libraries Required by Z-World Controllers

Library Needed	Controller
VDRIVER.LIB	All controllers
EZIOCMMN.LIB	All controllers
EZIOPBDV.LIB	All controllers
EZIOTGPL.LIB	BL1000
EZIOLGPL.LIB	BL1100
EZIOMGPL.LIB	BL1400, BL1500
EZIOPLC.LIB	BL1200, BL1600, PK2100, PK2200
EZIOPLC2.LIB	BL1700

Before using one of these libraries in an application, first include the library name in a **#use** command. For example, to use functions in the library **EZIOPLC.LIB**, be sure there is a line at the beginning of the program in the following format.

```
#use EZIOPLC.LIB
```

Supplied Software

These Dynamic C functions are used to initialize the PLCBus. Call these functions in a program before any code to read inputs or set outputs.

- **`VdInit()`**

  Initializes the timer mechanism.

  LIBRARY: **`VDRIVER.LIB`**

- **`void eioResetPlcBus()`**

  Resets all expansion boards connected to the PLCBus.

  When using this function, initialize timers with **`VdInit()`** before resetting the PLCBus. All PLCBus devices must reset before performing any subsequent operations.

  LIBRARY: **`EZIOPLC.LIB`**

- **`void eioPlcRstWait()`**

  Provides a delay long enough for the PLCBus to reset.

  This function provides a delay of 1–2 seconds to ensure devices on the PLCBus reset. This function should be called after resetting the PLCBus.

  LIBRARY: **`EZIOPBDV.LIB`**

- **`long int eioErrorCode`**

  Represents a global bit-mapped variable whose flags reflect error occurrences.

  This register for this variable is initially set to 0. If the application tries to access an invalid channel, the flag **`EIO_NODEV`** (the first bit flag) is set in this register. Note that the other bits in **`EIO_NODEV`** deal with networked controllers.

Digital Inputs/Outputs

The following functions provide an easy way to read inputs and activate outputs. The digital input and output functions are located in the Dynamic C **`EZIOPBDV.LIB`** library.

Setting Inputs

- **`int plcXP82In(unsigned address)`**

  Reads the status of a protected XP8200 input pin.

  PARAMETER: **`address`**, the address of the input pin, is **`16*board_address+pin_number`**, where **`board_address`** is **`PAL#*2+JUMPER`**, the XP8200 expansion board's logical address (0 if the address jumper is installed, 1 if the address jumper is not installed). **`pin_number`** ranges from 0 (for IN0) to 15 (for IN15).

  RETURN VALUE: 0 if the input pin reads low, 1 if the input pin reads high, and –1 if the input pin (the XP8200 board) does not exist. If –1 is returned, the flag **`EIO_NODEV`** is set in **`eioErrorCode`**.

Program 8-1 demonstrates how to read the status of a digital input.

Program 8-1. Input Demonstration Program

```
#use vdriver.lib
#use eziocmmn.lib
#use eziopbdv.lib
// uncomment #use ezioplc.lib line below for
// PK2100(Rugged Giant), PK2200(Little Star),
// BL1200(Little PLC) and BL1600(Little G)
// #use ezioplc.lib
// uncomment #use eziomgpl.lib line below for
// BL1400(Micro-G) or BL1500(Micro-G2)
// #use eziomgpl.lib
char TITLE[] = {“XP8200 Digital Input”};
main() {
   int channum;
   int i, j;
   VdInit();
   printf(“%s\n\n”, TITLE);
   eioResetPlcBus();     // reset the PLCBus
   eioPlcRstWait();      // delay ensures the
                         // PLCBus boards reset
                         // locate all possible
                         // jumper-set addresses
                         // from 0 to 1 and
                         // display status
   for (i = 0; i <= 2; ++i){
      if (plcXP82In(i*16)==-1) {    // do a read to
                                    // locate the board
         printf(“Board %d is not located\n\n”,i);
      }
      else {
         printf(“Board %d is located\n”,i);
         //read each channel from 0 to 15 and display status
         printf(“Reading all 16 positions\n”);
         for (channum = 0; channum <= 15; ++channum) {
            j = plcXP82In(i*16+channum);
            // read the input of the channel
            printf(“HV%d reads %d\n”, channum, j);
         }
      printf(“\nPress a key to continue...\n”);
      while (!kbhit());
      getchar();
      }
   }
}
```

Setting Outputs

- **`int plcXP82Out(unsigned address, int state)`**

  Switches high-voltage position on XP8200 board.

  PARAMETERS: **`address`** is **`8*board#+pin#`**, with **`board#`** 0 or 1, depending on the address jumper on header J1. **`board#`** can range from 0 to 31 with special PALs. **`pin#`** ranges from 0 to 5.

  `state` indicates whether the output should be enabled: the specified pin is disabled if **`state`** is 0, otherwise the pin is enabled.

  RETURN VALUE: 0 if the XP8200 expansion board exists, otherwise –1. If –1 is returned, the flag **`EIO_NODEV`** is set in **`eioErrorCode`**.

Program 8-2 demonstrates how to set the status of a digital output.

Program 8-2. Output Demonstration Program

```
#use vdriver.lib
#use eziocmmn.lib
#use eziopbdv.lib
#use ezioplc.lib
// uncomment #use ezioplc.lib line below for
// PK2100(Rugged Giant), PK2200(Little Star),
// BL1200(Little PLC) and BL1600(Little G)
// #use ezioplc.lib
// uncomment #use eziomgpl.lib line below for
// BL1400(Micro-G) or BL1500(Micro-G2)
// #use eziomgpl.lib
char TITLE[] = {“XP8200 Digital Output”};
main() {
   int channum;
   int i;
   VdInit();
   printf(“%s\n\n”, TITLE);
   eioResetPlcBus();     // reset the PLCBus
   eioPlcRstWait();      // delay ensures the
                         // PLCBus boards reset
                         // locate all possible
                         // jumper-set addresses
                         // from 0-1 and display
                         // status
   for (i = 0; i <= 1; ++i) {
      if (plcXP82In(i*6)==-1) { //read to locate board
         printf(“Board %d is not located\n\n”,i);
      }
      else {
         printf(“Board %d is located\n”,i);
                         // enable each chan from 0-5
         printf(“Enabling all 6 positions\n”);
         or (channum = 0; channum <= 5; ++channum)
         plcXP82Out(i*6+channum,1); //wr state to out chan
         printf(“Press a key to continue...\n”);
         while (!kbhit());
         getchar();      // disable each chan from 0-5
         printf(“Disabling all 32 positions\n”);
         for (channum = 0; channum <= 5; ++channum)
         plcXP82Out(i*6+channum,0); //wr state to out chan
         printf(“Press a key to continue...\n”);
         while (!kbhit());
         getchar();
      }
      printf(“\n”);
   }
}
```

Advanced Programming

While the functions described in the last four pages are easy to use to read and set input/output channels, they may not be able to meet the requirements of critical, real-time applications. This section discusses how to access the inputs/outputs on the XP8200 expansion boards more efficiently. To this, the reader must be familiar with binary arithmetic, C programming, and low-level PLCBus operations.

Address Calculation

Each XP8200 can have one of 16 PALs. The jumper on header J1 (pins 1–2) can be set two ways. Thus, there can be 32 XP8200s on a single PLCBus. Each board's physical address has the format

```
qrsa 010p 0000
```

where

`pqrs` is determined by the PAL, and

`a` = **`1`** when J1 pins 1–2 are not connected.

BUSADR0 contains the least significant four bits as shown below.

address:	**`0000`**	**`qrsa`**	**`010p`**	**`0000`**
BUSxxxx:		ADR2	ADR1	ADR0

XP8200 addresses, therefore, range from 0x040 to 0x05F. The 12-bit address can be placed on the bus using the functions **`set12adr`**, **`read12data`**, and **`write12data`**.

Logical Addresses

Dynamic C recognizes 16 of the 32 possible XP8200s. It refers to the input channels and output drivers for these 16 boards directly by assigning them numbers: **`board#`** << **`4 + channel#`**, as shown in Table 8-2.

Table 8-2. XP8200 Logical Addresses

Board	Index	Input Channel Numbers	Output Channel Numbers
0x040	0x0	0x00–0x0F	0x00–0x05
0x041	0x1	0x10–0x1F	0x10–0x15
...	...	...	...
0x04E	0xE	0xE0–0xEF	0xE0–0xE5
0x04F	0xF	0xF0–0xFF	0xF0–0xF5

Many functions in the **`UIBOARD.LIB`** library use this numbering scheme.

Functions for PLCBus Cycles, Reading and Writing

The PLCBus functions described in this section for the XP8100 Series expansion boards will make a program more abstract and portable. Dynamic C's **inport** and **outport** statements or **in** and **out** assembly instructions may still be used for controllers that support the PLCBus directly. However, the expansion boards still have to be reset and a delay has to be provided to ensure that all resets have occurred.

The following functions are located in **EZIOPBDV.LIB**.

- **unsigned _eioPlcUIOAddr(char BrdAddr)**

  Converts the logical address into a 12-bit physical address.

  PARAMETER: **BrdAddr** is the board address, 0–31. The logical address of the XP8200 expansion boards (**2*PAL# + Jumper#**) has a format of **000p qrsa**, where **pqrs** is the PAL number and **a** is the jumper-selected board address (0 if jumper installed, otherwise 1).

  The function converts the logical address into a 12-bit physical address, **qrsa 010p 0000**.

  RETURN VALUE: The bit-mingled XP8200 physical address.

The following functions are located in **EZIOPLC.LIB** and can be used to simplify the multiple writes and reads on the PLCBus.

- **void eioPlcAdr12(unsigned addr)**

  Specifies an address on the PLCBus using the BUSADR0, BUSADR1, and BUSADR2 cycles. **addr** is broken into three nibbles, and one nibble is written during each BUSADRx cycle, with BUSADR0 the first bus cycle.

  addr contains the PLCBus cycle addresses. BUSADR0 contains the least significant four bits as shown below.

addr:	**0000**	**qrsa**	**010p**	**0000**
BUSxxxx:.		ADR2	ADR1	ADR0

- **void eioPlcAdr4(unsigned addr)**

  This function writes to PLCBus register BUSADR2.

  addr is the most significant four bits, **qrsa**. This function writes **qrsa** only in register BUSADR2.

- **char _eioReadD0()**

  This function reads the BUSRD0 register and returns the four data bits D3–D0 read off the PLCBus.

- **char _eioReadD1()**

  This function reads the BUSRD1 register and returns the four data bits D3–D0 read off the PLCBus.

- **`void _eioWriteWR(char ch)`**

  This function writes to the BUSWR register.

  `ch` is the four data bits, D3–D0, written in the BUSWR register.

Operating the XP8200

The general method to use an XP8200 expansion board is described below.

1. Send a reset command to all boards on the PLCBus.
2. Get the board's attention by placing its address on the PLCBus.
3. Read the inputs on the board. Set or clear the driver outputs.

The XP8200 has 16 universal inputs and 6 driver outputs. To access an input channel or an output channel, write to the board's latch using the BUSWR cycle, as shown in Table 8-3.

Table 8-3. Setting XP8200 Output Drivers

D1	D0
0	Set or clear /L0
1	Set or clear /L1
0	Set or clear /L2
1	Set or clear /L3
0	Set or clear /L4
1	Set or clear /L5
0	*Unused*
1	Set up a read. When D0 = 0, select even input channels. When D = 1, select odd input channels

Setting an output driver is straightforward. However, reading one of the input channels in digital mode is slightly more complicated. There are two different read cycles: BUSRD1 corresponds to input channels 0–7, and BUSRD2 corresponds to channels 8–15. To set up for a read, write 0xE to the latch to select the even input channels, and write 0xF to select the odd input channels. Thus, the BUSRD1 and BUSRD2 cycles read four channels at a time, one bit per channel.

The various XP8200 bus cycles are explained in Table 4-8.

Table 8-4. XP8200 Bus Cycles

Register	Address	Usage
BUSADR0	0xC8	First address byte.
BUSADR1	0xCA	Second address byte.
BUSADR2	0xCC	Third address byte.
BUSWR	0xCE	Write data to the latch. This selects an output or specifies odd or even input channels.
BUSRD0	0xC0	Determine the presence of an XP8200. Bit 0, when 0, indicates the presence of a properly addressed board.
BUSRD1	0xC2	Read input channels [0, 2, 4, 6] or [1, 3, 5, 7]. Which group is selected by a prior write to the latch.
BUSRD2	0xC4	Read input channels [8, 10, 12, 14] or [9, 11, 13, 15]. Which group is selected by a prior write to the latch.
BUSRESET	0xC6	Resets all expansion cards on the PLCBus.

Other Supported Software for the XP8200

Z-World recommends the functions described in the "Supplied Software" section because they can be used for all Z-World controllers that support a PLCBus or a simulated PLCBus. This section describes software that is supported by Z-World for the PK2100 and PK2200 controllers.

Low-level functions, such as **`set12adr`**, **`write12data`**, and **`read12data`**, are in **`DRIVERS.LIB`**. Functions specific to the XP8200 may be found in **`UIBOARD.LIB`**.

The **`UIBOARD.LIB`** library supports up to 16 XP8200s with addresses 0x040 to 0x04F. Call **`_uiLocateBrd()`** before using another function in this library to initialize the board and to determine which XP8200s, among the first 16, are present. The software in **`UIBOARD.LIB`** refers to input channels and output drivers directly.

Virtual Driver

The XP8200 can also be controlled by using the virtual driver. The virtual driver is a background task that periodically (every 25 ms) services each XP8200 on the PLCBus. It reads the input channels, storing the values in software variables and it sets the driver outputs according the value in software variables.

The programmer deals only with the variables and the background task—**`ui_drivers()`**—handles the XP8200.

A change in input has to persist for two “ticks” of the virtual driver before the variable recording the state of the input is changed.

Direct Software

- **`void _uiLocateBrd`**

  Determines the presence of the first 16 XP8200s. The function also initializes variables to be used with the virtual driver. See the next section.

  This function must be called before any others in **`UIBOARD.LIB`**.

- **`int _uiOut( int channel, int state )`**

  Sets the driver identified by **channel**. If **state** is 0, the driver is reset. Otherwise, the driver is set. The channel number for the first board can be 0x00–0x05, the channel for the second board can be 0x10–0x15, and so on.

  RETURN VALUE: 1 if the function is successful, and –1 otherwise.

- **`int _uiDigin( int channel )`**

  Reads a digital input identified by **`channel`**. The channel number for the first board can be 0x00–0x0F, the channel for the second board can be 0x10–0x1F, and so on.

  RETURN VALUE: 0 or 1, depending on the state of the input; –1 if unsuccessful.

- **`int _uiAnalog( int channel )`**

  Reads an analog input identified by **`channel`**. The channel number for the first board can be 0x00–0x0F, the channel for the second board can be 0x10–0x1F, and so on.

  RETURN VALUE: 0–10,000 (millivolts) if successful; –1 if unsuccessful.

  This function is only supported with the PK2100. It assumes that the UEXP output of the PK2100 is used as the external analog voltage source connected to the DA terminal of the XP8200.

- **`int _uiAdcal( int channel )`**

  Returns a calibrated analog value from the specified channel (0–15) on the XP8200 currently selected. The function returns **`0–10,000`** (millivolts) if it is successful.

  RETURN VALUE: 0–10,000 (millivolts) if successful; –1 if unsuccessful.

- **`int cplc_ui_adrd( int channel )`**

  Obtains a raw value (analog) from the specified channel on the currently selected XP8200.

- **`int _uiDocal( int raw_value )`**

  Returns a calibrated analog value from **`raw_value`**.

Virtual Driver Software

These constants defined in **`UIBOARD.LIB`** concern the virtual driver.

```
#define UIUNIVERSAL 1    //treat inputs as universal
#define UIDIGITAL     0  //treat inputs as digital
#define VIRTUAL_UIEXP 2        //number of UI boards
```

The virtual driver reads digital input when a board's input mode is UIDIGITAL and the line DAV is connected to VL. The inputs are "universal" when a board's input mode is UIUNIVERSAL and DAV is connected to the DA terminal (which must be connected to an analog voltage source.)

The constant **`VIRTUAL_UIEXP`** specifies the number of XP8200s that will be operated by the virtual driver. The default is 2.

An array (UI in the virtual driver) of **`VIRTUAL_UIEXP`** elements provides storage for the input and output variables. It has the following structure.

```
struct {
   byte OUT[6];            // for driver outputs
   byte inputmode;         // digital or universal
   byte DIG[16];           // digital input values
                           // universal values with
                           // hysteresis settings
   shared int UHIGH0, ULOW0;  byte UCUR0, UIN0;
   shared int UHIGH1, ULOW1;  byte UCUR1, UIN1;
   ...
   shared int UHIGH14,ULOW14; byte UCUR14,UIN14;
   shared int UHIGH15,ULOW15; byte UCUR15,UIN15;
} uiv_data;
```

- **`void _uiExpInit()`**

  Initializes the virtual driver variables.

- **`int _uiSetIMode( int index, int mode )`**

  Sets the input mode of a board identified by **`index`**. The index must, of course, be a value from **`0`** to **`VIRTUAL_UIEXP – 1`**. **`index`** = 0 identifies board 0x040. **`index`** = 15 identifies board 0x04F. The term **`mode`** can be digital (0) or universal (1).

When the input mode is digital, the variables UI[index].DIG[*i*] are set to 1 or 0, depending on the level of the input relative to the fixed threshold (VL) on the comparator's negative input.

When the input mode is universal—and the UEXP channel of the PK2100 is connected to DA—the variables UHIGH*xx*, ULOW*xx*, UCUR*xx*, and UIN*xx* are active. The hysteresis thresholds UI[index].UHIGH*xx* and UI[index].ULOW*xx* can be set. When an input channel (*xx*) has a voltage higher than its high threshold, UI[index].UIN*xx* becomes a "1," and when it has a voltage lower than its low threshold, UI[index].UIN*xx* becomes a "0." (The background task **`ui_drivers`** takes care of monitoring the inputs.) If the input is between the high and low values, UIN*xx* is not changed.

- **`int ui_drivers()`**

This is the function that services the variables representing XP8200 inputs and outputs. This function can be called periodically from a background function. It can be called by the real-time kernel.

For the PK2100, this function is called through the background routine that services the keypad and LCD. To make this background function invoke **`ui_drivers()`**, place the following definition in the program.

```
#define UIBOARD_VDVR
```

APPENDICES

Blank

Appendix A: PLCBus

Appendix A provides the pin assignments for the PLCBus, describes the registers, and lists the software drivers.

PLCBus Overview

The PLCBus is a general-purpose expansion bus for Z-World controllers. The PLCBus is available on the BL1200, BL1600, BL1700, PK2100, and PK2200 controllers. The BL1000, BL1100, BL1300, BL1400, and BL1500 controllers support the XP8300, XP8400, XP8600, and XP8900 expansion boards using the controller's parallel input/output port. The BL1400 and BL1500 also support the XP8200 and XP8500 expansion boards. The ZB4100's PLCBus supports most expansion boards, except for the XP8700 and the XP8800. The SE1100 adds expansion capability to boards with or without a PLCBus interface.

Table A-1 lists Z-World's expansion devices that are supported on the PLCBus.

Table A-1. Z-World PLCBus Expansion Devices

Device	Description
Exp-A/D12	Eight channels of 12-bit A/D converters
SE1100	Four SPDT relays for use with all Z-World controllers
XP8100 Series	32 digital inputs/outputs
XP8200	"Universal Input/Output Board" —16 universal inputs, 6 high-current digital outputs
XP8300	Two high-power SPDT and four high-power SPST relays
XP8400	Eight low-power SPST DIP relays
XP8500	11 channels of 12-bit A/D converters
XP8600	Two channels of 12-bit D/A converters
XP8700	One full-duplex asynchronous RS-232 port
XP8800	One-axis stepper motor control
XP8900	Eight channels of 12-bit D/A converters

Multiple expansion boards may be linked together and connected to a Z-World controller to form an extended system.

Figure A-1 shows the pin layout for the PLCBus connector.

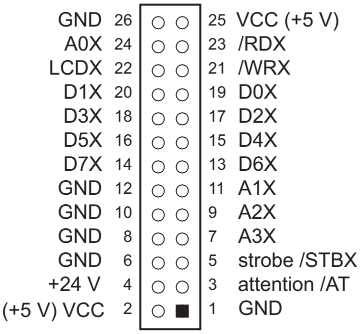


Figure A-1. PLCBus Pin Diagram

Two independent buses, the LCD bus and the PLCBus, exist on the single connector.

The LCD bus consists of the following lines.

- LCDX—positive-going strobe.
- /RDX—negative-going strobe for read.
- /WRX—negative-going strobe for write.
- A0X—address line for LCD register selection.
- D0X-D7X—bidirectional data lines (shared with expansion bus).

The LCD bus is used to connect Z-World's OP6000 series interfaces or to drive certain small liquid crystal displays directly. Figure A-2 illustrates the connection of an OP6000 interface to a PLCBus header.

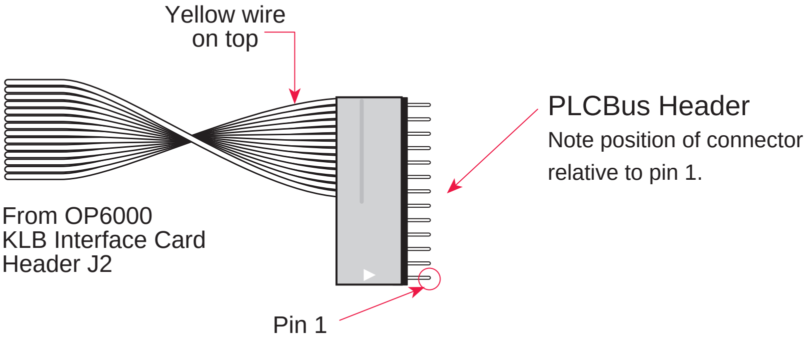

Yellow wire on top
PLCBus Header
Note position of connector relative to pin 1.
From OP6000 KLB Interface Card Header J2
Pin 1

Figure A-2. OP6000 Connection to PLCBus Header

The PLCBus consists of the following lines.

- /STBX—negative-going strobe.
- A1X–A3X—three control lines for selecting bus operation.
- D0X–D3X—four bidirectional data lines used for 4-bit operations.
- D4X–D7X—four additional data lines for 8-bit operations.
- /AT—attention line (open drain) that may be pulled low by any device, causing an interrupt.

The PLCBus may be used as a 4-bit bus (D0X–D3X) or as an 8-bit bus (D0X–D7X). Whether it is used as a 4-bit bus or an 8-bit bus depends on the encoding of the address placed on the bus. Some PLCBus expansion cards require 4-bit addressing and others (such as the XP8700) require 8-bit addressing. These devices may be mixed on a single bus.

There are eight registers corresponding to the modes determined by bus lines A1X, A2X, and A3X. The registers are listed in Table A-2.

Table A-2. PLCBus Registers

Register	Address	A3	A2	A1	Meaning
BUSRD0	C0	0	0	0	Read data, one way
BUSRD1	C2	0	0	1	Read data, another way
BUSRD2	C4	0	1	0	Spare, or read data
BUSRESET	C6	0	1	1	Read this register to reset the PLCBus
BUSADR0	C8	1	0	0	First address nibble or byte
BUSADR1	CA	1	0	1	Second address nibble or byte
BUSADR2	CC	1	1	0	Third address nibble or byte
BUSWR	CE	1	1	1	Write data

Writing or reading one of these registers takes care of all the bus details. Functions are available in Z-World's software libraries to read from or write to expansion bus devices.

To communicate with a device on the expansion bus, first select a register associated with the device. Then read or write from/to the register. The register is selected by placing its address on the bus. Each device recognizes its own address and latches itself internally.

A typical device has three internal latches corresponding to the three address bytes. The first is latched when a matching BUSADR0 is detected. The second is latched when the first is latched and a matching BUSADR1 is detected. The third is latched if the first two are latched and a matching BUSADR2 is detected. If 4-bit addressing is used, then there are three 4-bit address nibbles, giving 12-bit addresses. In addition, a special register address is reserved for address expansion. This address, if ever used, would provide an additional four bits of addressing when using the 4-bit convention.

If eight data lines are used, then the addressing possibilities of the bus become much greater—more than 256 million addresses according to the conventions established for the bus.

Place an address on the bus by writing (bytes) to BUSADR0, BUSADR1 and BUSADR2 in succession. Since 4-bit and 8-bit addressing modes must coexist, the lower four bits of the first address byte (written to BUSADR0) identify addressing categories, and distinguish 4-bit and 8-bit modes from each other.

There are 16 address categories, as listed in Table A-3. An “x” indicates that the address bit may be a “1” or a “0.”

Table A-3. First-Level PLCBus Address Coding

First Byte	Mode	Addresses	Full Address Encoding
– – – – 0 0 0 0 – – – – 0 0 0 1 – – – – 0 0 1 0 – – – – 0 0 1 1	4 bits × 3	256 256 256 256	0000 xxxx xxxx 0001 xxxx xxxx 0010 xxxx xxxx 0011 xxxx xxxx
– – – x 0 1 0 0 – – – x 0 1 0 1 – – – x 0 1 1 0 – – – x 0 1 1 1	5 bits × 3	2,048 2,048 2,048 2,048	x0100 xxxxx xxxxx x0101 xxxxx xxxxx x0110 xxxxx xxxxx x0111 xxxxx xxxxx
– – x x 1 0 0 0 – – x x 1 0 0 1	6 bits × 3	16,384 16,384	xx1000 xxxxxx xxxxxx xx1001 xxxxxx xxxxxx
– – x x 1 0 1 0	6 bits × 1	4	xx1010
– – – – 1 0 1 1	4 bits × 1	1	1011 (expansion register)
x x x x 1 1 0 0	8 bits × 2	4,096	xxxx1100 xxxxxxxx
x x x x 1 1 0 1	8 bits × 3	1M	xxxx1101 xxxxxxxx xxxxx xxx
x x x x 1 1 1 0	8 bits × 1	16	xxxx1110
x x x x 1 1 1 1	8 bits × 1	16	xxxx1111

This scheme uses less than the full addressing space. The mode notation indicates how many bus address cycles must take place and how many bits are placed on the bus during each cycle. For example, the 5 × 3 mode means three bus cycles with five address bits each time to yield 15-bit addresses, not 24-bit addresses, since the bus uses only the lower five bits of the three address bytes.

Z-World provides software drivers that access the PLCBus. To allow access to bus devices in a multiprocessing environment, the expansion register and the address registers are shadowed with memory locations known as *shadow registers*. The 4-byte shadow registers, which are saved at predefined memory addresses, are as follows.

		SHBUS1	SHBUS1+1
SHBUS0	SHBUS0+1	SHBUS0+2	SHBUS0+3
Bus expansion	BUSADR0	BUSADR1	BUSADR2

Before the new addresses or expansion register values are output to the bus, their values are stored in the shadow registers. All interrupts that use the bus save the four shadow registers on the stack. Then, when exiting the interrupt routine, they restore the shadow registers and output the three address registers and the expansion registers to the bus. This allows an interrupt routine to access the bus without disturbing the activity of a background routine that also accesses the bus.

To work reliably, bus devices must be designed according to the following rules.

1. The device must not rely on critical timing such as a minimum delay between two successive register accesses.
2. The device must be capable of being selected and deselected without adversely affecting the internal operation of the controller.

Allocation of Devices on the Bus

4-Bit Devices

Table A-4 provides the address allocations for the registers of 4-bit devices.

Table A-4. Allocation of Registers

A1	A2	A3	Meaning
000j	000j	xxxj	digital output registers, 64 registers 64 × 8 = 512 1-bit registers
000j	001j	xxxj	analog output modules, 64 registers
000j	01xj	xxxj	digital input registers, 128 registers 128 × 4 = 512 input bits
000j	10xj	xxxj	analog input modules, 128 registers
000j	11xj	xxxj	128 spare registers (customer)
001j	xxxj	xxxj	512 spare registers (Z-World)

j	controlled by board jumper
x	controlled by PAL

Digital output devices, such as relay drivers, should be addressed with three 4-bit addresses followed by a 4-bit data write to the control register. The control registers are configured as follows

bit 3	bit 2	bit 1	bit 0
A2	A1	A0	D

The three address lines determine which output bit is to be written. The output is set as either 1 or 0, according to D. If the device exists on the bus, reading the register drives bit 0 low. Otherwise bit 0 is a 1.

For digital input, each register (BUSRD0) returns four bits. The read register, BUSRD1, drives bit 0 low if the device exists on the bus.

8-Bit Devices

Z-World's XP8700 and XP8800 expansion boards use 8-bit addressing. Refer to the ***XP8700 and XP8800*** manual.

Expansion Bus Software

The expansion bus provides a convenient way to interface Z-World's controllers with expansion boards or other specially designed boards. The expansion bus may be accessed by using input functions. Follow the suggested protocol. The software drivers are easier to use, but are less efficient in some cases. Table A-5 summarizes the libraries.

Table A-5. Dynamic C PLCBus Libraries Used by Z-World Controllers

Library Needed	Controller
DRIVERS.LIB	All controllers
EZIOTGPL.LIB	BL1000
EZIOLGPL.LIB	BL1100
EZIOMGPL.LIB	BL1400, BL1500
EZIOPLC.LIB	BL1200, BL1600, PK2100, PK2200, ZB4100
EZIOPLC2.LIB	BL1700
PBUS_TG.LIB	BL1000
PBUS_LG.LIB	BL1100, BL1300
PLC_EXP.LIB	BL1200, BL1600, PK2100, PK2200

There are 4-bit and 8-bit drivers. The 4-bit drivers employ the following calls.

- **`void eioResetPlcBus()`**

  Resets all expansion boards on the PLCBus. When using this call, make sure there is sufficient delay between this call and the first access to an expansion board.

  LIBRARY: **`EZIOPLC.LIB`**, **`EZIOPLC2.LIB`**, **`EZIOMGPL.LIB`**.

- **`void eioPlcAdr12( unsigned addr )`**

  Specifies the address to be written to the PLCBus using cycles BUSADR0, BUSADR1, and BUSADR2.

  PARAMETER: **`addr`** is broken into three nibbles, and one nibble is written in each BUSADR*x* cycle.

  LIBRARY: **`EZIOPLC.LIB`**, **`EZIOPLC2.LIB`**, **`EZIOMGPL.LIB`**.

- **`void set16adr( int adr )`**

  Sets the current address for the PLCBus. All read and write operations access this address until a new address is set.

  PARAMETER: **`adr`** is a 16-bit physical address. The high-order nibble contains the value for the expansion register, and the remaining three 4-bit nibbles form a 12-bit address (the first and last nibbles must be swapped).

  LIBRARY: **`DRIVERS.LIB`**.

- **`void set12adr( int adr )`**

  Sets the current address for the PLCBus. All read and write operations access this address until a new address is set.

  PARAMETER: **`adr`** is a 12-bit physical address (three 4-bit nibbles) with the first and third nibbles swapped.

  LIBRARY: **`DRIVERS.LIB`**.

- **`void eioPlcAdr4( unsigned addr )`**

  Specifies the address to be written to the PLCBus using only cycle BUSADR2.

  PARAMETER: **`addr`** is the nibble corresponding to BUSADR2.

  LIBRARY: **`EZIOPLC.LIB`**, **`EZIOPLC2.LIB`**, **`EZIOMGPL.LIB`**.

- **`void set4adr( int adr )`**

  Sets the current address for the PLCBus. All read and write operations access this address until a new address is set.

  A 12-bit address may be passed to this function, but only the last four bits will be set. Call this function only if the first eight bits of the address are the same as the address in the previous call to **`set12adr`**.

  PARAMETER: **`adr`** contains the last four bits (bits 8–11) of the physical address.

  LIBRARY: **`DRIVERS.LIB`**.

- **`char _eioReadD0( )`**

  Reads the data on the PLCBus in the BUSADR0 cycle.

  RETURN VALUE: the byte read on the PLCBus in the BUSADR0 cycle.

  LIBRARY: **`EZIOPLC.LIB`**, **`EZIOPLC2.LIB`**, **`EZIOMGPL.LIB`**.

- **`char _eioReadD1( )`**

  Reads the data on the PLCBus in the BUSADR1 cycle.

  RETURN VALUE: the byte read on the PLCBus in the BUSADR1 cycle.

  LIBRARY: **`EZIOPLC.LIB`**, **`EZIOPLC2.LIB`**, **`EZIOMGPL.LIB`**.

- **`char _eioReadD2( )`**

  Reads the data on the PLCBus in the BUSADR2 cycle.

  RETURN VALUE: the byte read on the PLCBus in the BUSADR2 cycle.

  LIBRARY: **`EZIOPLC.LIB`**, **`EZIOPLC2.LIB`**, **`EZIOMGPL.LIB`**.

- **`char read12data( int adr )`**

  Sets the current PLCBus address using the 12-bit **`adr`**, then reads four bits of data from the PLCBus with BUSADR0 cycle.

  RETURN VALUE: PLCBus data in the lower four bits; the upper bits are undefined.

  LIBRARY: **`DRIVERS.LIB`**.

- **`char read4data( int adr )`**

  Sets the last four bits of the current PLCBus address using adr bits 8–11, then reads four bits of data from the bus with BUSADR0 cycle.

  PARAMETER: adr bits 8–11 specifies the address to read.

  RETURN VALUE: PLCBus data in the lower four bits; the upper bits are undefined.

  LIBRARY: **`DRIVERS.LIB`**.

- **`void _eioWriteWR( char ch)`**

  Writes information to the PLCBus during the BUSWR cycle.

  PARAMETER: ch is the character to be written to the PLCBus.

  LIBRARY: **`EZIOPLC.LIB`**, **`EZIOPLC2.LIB`**, **`EZIOMGPL.LIB`**.

- **`void write12data( int adr, char dat )`**

  Sets the current PLCBus address, then writes four bits of data to the PLCBus.

  PARAMETER: **`adr`** is the 12-bit address to which the PLCBus is set.

  `dat` (bits 0–3) specifies the data to write to the PLCBus.

  LIBRARY: **`DRIVERS.LIB`**.

- **`void write4data( int address, char data )`**

  Sets the last four bits of the current PLCBus address, then writes four bits of data to the PLCBus.

  PARAMETER: **`adr`** contains the last four bits of the physical address (bits 8–11).

  `dat` (bits 0–3) specifies the data to write to the PLCBus.

  LIBRARY: **`DRIVERS.LIB`**.

The 8-bit drivers employ the following calls.

- **`void set24adr( long address )`**

  Sets a 24-bit address (three 8-bit nibbles) on the PLCBus. All read and write operations will access this address until a new address is set.

  PARAMETER: **`address`** is a 24-bit physical address (for 8-bit bus) with the first and third bytes swapped (low byte most significant).

  LIBRARY: **`DRIVERS.LIB`**.

- **`void set8adr( long address )`**

  Sets the current address on the PLCBus. All read and write operations will access this address until a new address is set.

  PARAMETER: **`address`** contains the last eight bits of the physical address in bits 16–23. A 24-bit address may be passed to this function, but only the last eight bits will be set. Call this function only if the first 16 bits of the address are the same as the address in the previous call to **`set24adr`**.

  LIBRARY: **`DRIVERS.LIB`**.

- **`int read24data0( long address )`**

  Sets the current PLCBus address using the 24-bit address, then reads eight bits of data from the PLCBus with a BUSRD0 cycle.

  RETURN VALUE: PLCBus data in lower eight bits (upper bits 0).

  LIBRARY: **`DRIVERS.LIB`**.

- **`int read8data0( long address )`**

  Sets the last eight bits of the current PLCBus address using address bits 16–23, then reads eight bits of data from the PLCBus with a BUSRD0 cycle.

  PARAMETER: **`address`** bits 16–23 are read.

  RETURN VALUE: PLCBus data in lower eight bits (upper bits 0).

  LIBRARY: **`DRIVERS.LIB`**.

- **`void write24data( long address, char data )`**

  Sets the current PLCBus address using the 24-bit address, then writes eight bits of data to the PLCBus.

  PARAMETERS: **`address`** is 24-bit address to write to.

  `data` is data to write to the PLCBus.

  LIBRARY: **`DRIVERS.LIB`**.

- **`void write8data( long address, char data )`**

  Sets the last eight bits of the current PLCBus address using address bits 16–23, then writes eight bits of data to the PLCBus.

  PARAMETERS: **`address`** bits 16–23 are the address of the PLCBus to write.

  `data` is data to write to the PLCBus.

  LIBRARY: **`DRIVERS.LIB`**.

Blank

APPENDIX B: SPECIFICATIONS

XP8100 Hardware Specifications

Inputs

Table B-1 summarizes the input specifications for the XP8100 Series expansion boards.

Table B-1. Input Specifications

Input Specifications	Standard Input
Input Voltage	-20 V to +24 V
Logic Threshold	2.5 V
Bias Resistors	User-settable "pull up" or "pull down"
Transient Voltage	–48 V to +48 V max
Input Protection	22 kΩ current-limiting series resistor, input-protection diode
Noise-Spike Filter	t_{RC} = 220 µs low-pass filter
I/O Connectors	Four 10-pin headers
Input Leakage Current	5 µA

The inputs will accept a voltage level between -20 and +24 volts with a logic threshold of 2.5 volts. A 22 kΩ current-limiting resistor paired with a CMOS input diode provides input protection. The resistor/capacitor connection to ground acts as a low-pass filter, where T_{RC} = 220 µs. Jumpers pull inputs to either +5 volts or ground through a bias resistor in groups of four or eight.

Figure B-1 shows a typical XP8100 Series expansion board input.

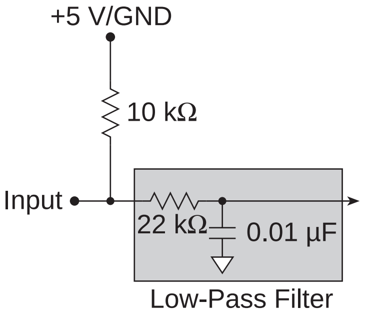


Figure B-1. XP8100 Series Input

Outputs

Table B-2 summarizes the output specifications for the XP8100 Series expansion boards.

Table B-2. Output Specifications

Output Specifications	Default Sinking Driver
Maximum Current	500 mA, single channel ON
Connections	(4) 10-pin headers
Noninductive voltage	+5 V to +48 V
Inductive Voltage	+5 V to +30 V
Switching Response Time	1 µs
Output Leakage Current	100 µA max

The maximum current is subject to the maximum power dissipation for the package and the ambient temperature. Make sure that the maximum current is properly derated for temperature and package power dissipation.

See Chapter 3, "I/O Configurations," for more information on derating.

All outputs are arranged in groups of eight and are driven by a ULN2803 sinking driver. If installed, the chip would be located at U5, U6, U13, or U14, shown in Figure 2-1 and in Chapter 1.

The sinking driver is rated up to a maximum voltage of 48 V and a maximum current of 500 mA per individual output. When all the outputs are on simultaneously, thermal limits restrict the current to 100 mA per output. Similarly, if multiple outputs are activated at the same time, the driver current should not exceed 350 mA per output.

A UDN2985 sourcing driver is optional. The UDN2985 is rated at 30 V and 250 mA for an individual output at 25°C. The sourcing drivers would be installed at U5, U6, U13, or U14 instead of the sinking drivers, and jumpers on headers J1 and J3 would be reconfigured, as discussed in Appendix D.

Refer to Appendix E, "Sourcing and TTL/CMOS Outputs," for information on installing and configuring your board for sourcing outputs and for TTL/CMOS outputs.

Mechanical Dimensions

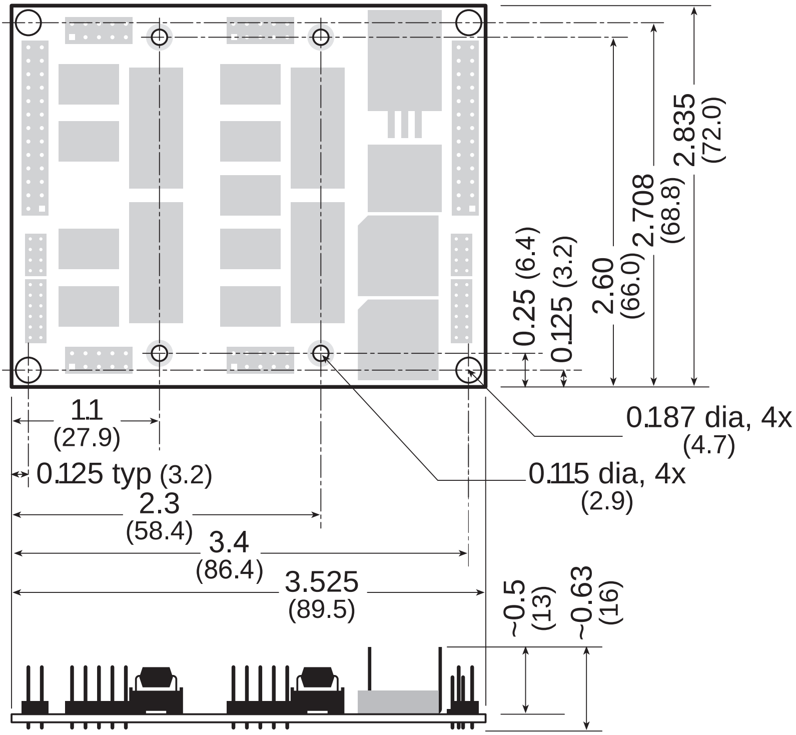


Figure B-2. XP8100 Series Board Dimensions

XP8200 Hardware Specifications

Table B-3 summarizes the specifications for the XP8200 expansion board.

Table B-3. XP8200 Specifications

Board Size	2.835" × 3.525" × 0.75" (72.0 mm × 89.5 mm × 19 mm)
Operating Temperature Range	-40°C to +70°C
Inputs	16 "universal" inputs up to 24 V, 500 mA
Outputs	• 6 outputs, each sinking 150 mA continuously at 50°C and 48 V dc • 1 output channel sinks 500 mA continuously at 25°C • no sourcing option

Mechanical Dimensions

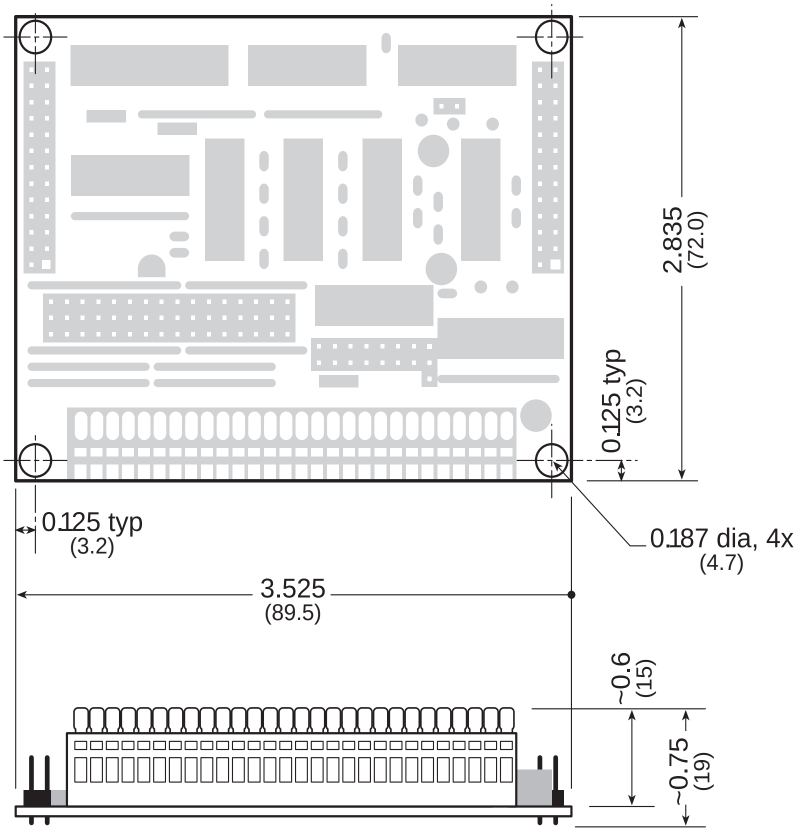


Figure B-3. XP8200 Board Dimensions

Appendix C: Connecting and Mounting Multiple Boards

Connecting Multiple Boards

Eight or more expansion boards can be connected ("daisy chained") at one time. The actual number of expansion boards may be limited by capacitative loading on the PLCBus.

Be sure that each expansion board has a unique address to prevent communication problems between the controller and the expansion board.

Follow these steps to install several expansion boards on a single PLCBus.

1. Place all expansion boards right side up.
2. Use the ribbon cable supplied with the boards.
3. Connect one board to the main controller.
4. Connect another expansion board to the first expansion board, connecting each board's header P1 to the adjacent board's header P2.

Figure C-1 illustrates a controller with expansion boards attached.

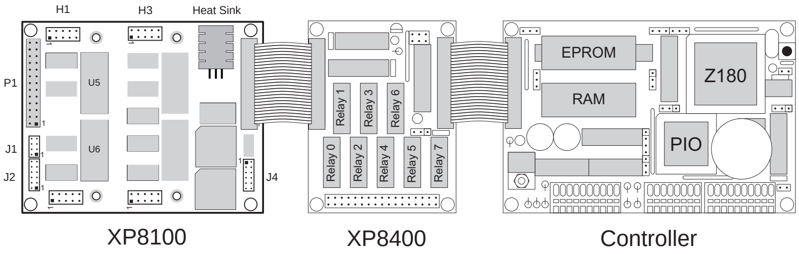


Figure C-1. Connecting Multiple Expansion Boards

Do not twist the ribbon cable or mount the expansion boards upside down! Damage may occur. Be sure Pin 1 of P1 and P2 of each board matches up with Pin 1 of the previous board. Pin 1 should be at the lower right when the expansion board is right side up, that is, the board markings are right side up.

When several expansion boards are connected, there may be a voltage drop along the network of expansion boards. No action is necessary as long as the digital voltage, VCC, is greater than 4.9 V on the last board.

VCC can be measured at pin 2 on header P1, and GND is pin 1 on header P1.

There are two ways to compensate for the voltage dropoff. The easiest way is to connect +5 V DC and ground from the host controller to pins 2 and 1 of header P1 on the last expansion board. Another solution, which can approximately double the number of boards that could otherwise be connected to a single controller, is a Y cable available from Z-World. Figure C-2 illustrates the use of the Y cable.

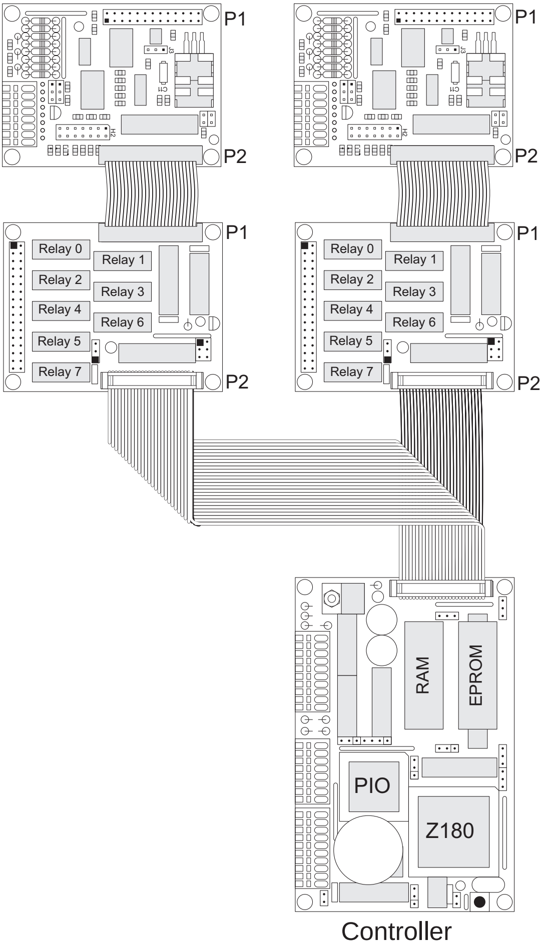


Figure C-2. Use of Y Cable to Connect Multiple Expansion Boards

☎ For more information, call your Z-World Technical Support Representative at (530) 757-3737.

Mounting

The XP8100 Series and XP8200 expansion boards can be installed in modular plastic circuit-board holders attached to a DIN rail, a widely used mounting system, as shown in Figure C-3.

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 and from other suppliers, snap together to form a tray of almost any length. Z-World's expansion boards are 72 mm wide and fit directly in these circuit-board holders.

Z-World's expansion boards can also be mounted with plastic standoffs to any flat surface that accepts screws. The mounting holes are 0.125 inches (1/8 inch) in from the edge of a board, and have a diameter of 0.190 inches.

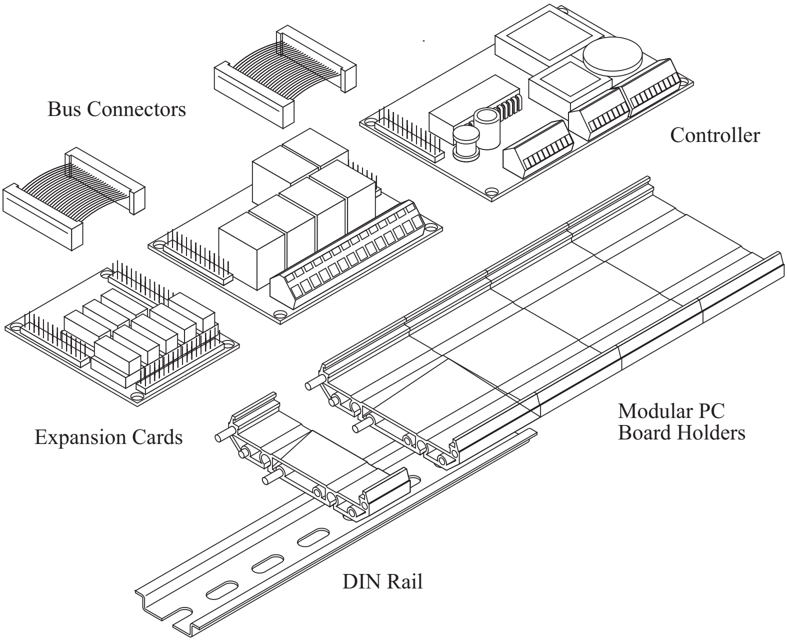


Figure C-3. Mounting Expansion Boards on DIN Rail

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

APPENDIX D: SINKING AND SOURCING DRIVERS

XP8100 Series Sinking and Sourcing Outputs

The XP8100 Series expansion boards are normally supplied with ULN2803 sinking drivers. Figure D-1 shows a typical sinking driver output configuration.

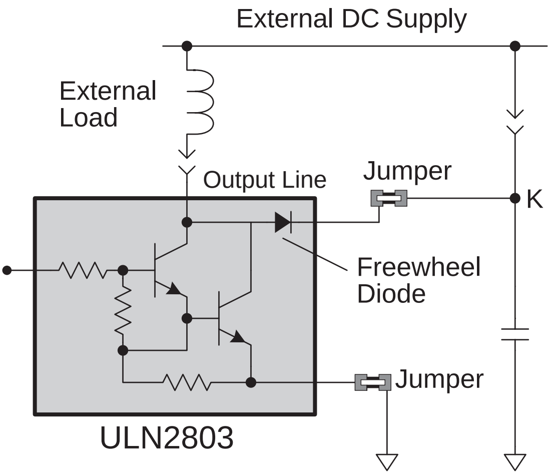


Figure D-1. XP8100 Series Sinking Driver Output

Figure D-2 shows the jumper configurations for a sinking driver output.

SINKING DRIVER JUMPER SETTINGS

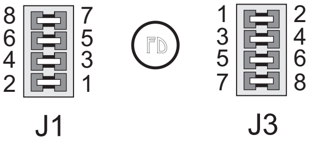


Figure D-2. Sinking Driver Jumper Configurations

Sourcing outputs are possible by replacing the factory-installed sinking driver chips with sourcing output drivers (UDN2985). The UDN2985 sourcing driver chip is capable of sourcing a maximum of 250 mA per output.

Figure D-3 shows a typical sourcing driver output.

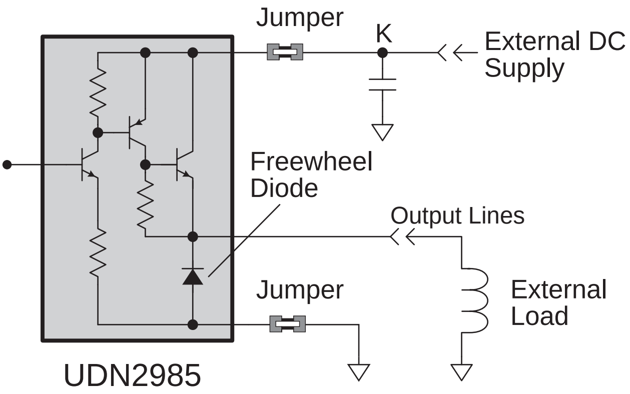


Figure D-3. XP8100 Series Sourcing Driver Output

Figure D-4 shows the jumper configurations for a sourcing driver output.

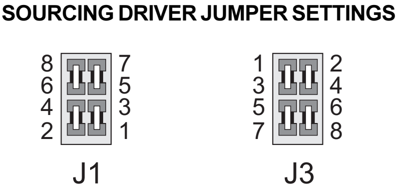


Figure D-4. Sourcing Driver Jumper Configurations

☎ Z-World also offers all XP8100 Series expansion boards with factory-installed sourcing drivers. For ordering information, call your Z-World Sales Representative at (530) 757-3737.

Installing Sourcing Drivers

Figure D-5 shows the location of the drivers and headers with jumpers to be changed.

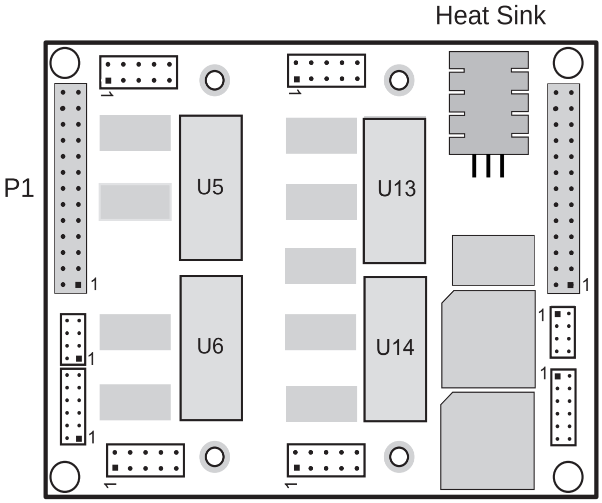


Figure D-5. U5, U6, U13 and U14 Locations of Sinking Drivers

Pay particular attention to the orientation of the jumpers when changing the driver output from sinking to sourcing. Exercise caution when installing sourcing drivers in the field.

1. Be sure power is removed from the controller, then disconnect the expansion card from the controller..
2. Remove the ULN2803 sinking drivers from the IC sockets. Note that the XP8100 has two ULN2803 chips (at U5 and U6) and the XP8120 has four (U5, U6, U13 and U14).
3. Install the jumpers on header J1 in the sourcing configuration, as shown in Figure C-4, for "Bank A" output channels 0–15. This step applies to both the XP8100 and the XP8120 expansion boards. Note the location of pin number 1.
4. For the XP8120 expansion board, install the J3 jumpers in the sourcing configuration, as shown in Figure D-4, for "Bank B" output channels 0–15. Note the location of pin number 1.

> ⚠ Be sure the jumper settings conform to what is specified. Failure to install jumpers correctly will cause the expansion board to fail.

5. Install UDN2985 sourcing driver chips into the IC sockets.

Tables D-1 and D-2 indicate which I/O channels are modified by the jumpers on the J1 and J3 headers. The tables also list the specific location of each output chip.

Table D-1. Header J1 I/O Channels

J1 Pins	"Bank A" I/O Channels Modified	IC Location
1–4	8–15	U6
5–8	0–7	U5

Table D-2. Header J3 I/O Channels

J3 Pins	"Bank B" I/O Channels	IC Location
1-4	8-15	U14
5-8	0-7	U13

TTL/CMOS Outputs

Z-World also offers TTL- or CMOS-compatible outputs for the XP8100 Series expansion boards. Input and output channels may be configured independently in any combination. However, the functionality of each input is not independent; the inputs are still characterized in groups of eight.

☎ Z-World offers all XP8100 Series expansion boards with factory-installed TTL- or CMOS-compatible outputs. For ordering information, call your Z-World Sales Representative at (530) 757-3737.

Using Output Drivers

The common supply for the digital output channels supplied by a driver chip is called "K," and is available on pin 10 of headers H1, H2, H3, and H4 when they are configured to operate as digital outputs. "K" must be used with digital outputs to allow proper operation.

The “K” connection performs two vital functions to the high-voltage driver circuitry on the XP8100.

1. “K” supplies power to driver circuitry inside the driver chip.
2. “K” also allows a diode internal to the driver chip to “snub” voltage transients produced during the inductive kick associated with switching inductive loads such as relays, solenoids, and speakers.

Long leads may present enough induction to also produce large potentially damaging voltage transients. The anodes of the protection diodes for each channel are common, and so only one voltage supply can be used for all high-voltage driver loads.

The following points summarize the functions of “K.”

- K provides power to the driver chip circuitry.
- K provides “clamping” for all high-voltage driver loads.
- It is mandatory to connect K regardless of whether sourcing or sinking.
- The load’s supply must have a common ground with all other supplies in your system.
- All loads must use same supply voltage.

K must be connected to the power supply used for the high-voltage load as shown in Figure D-6.

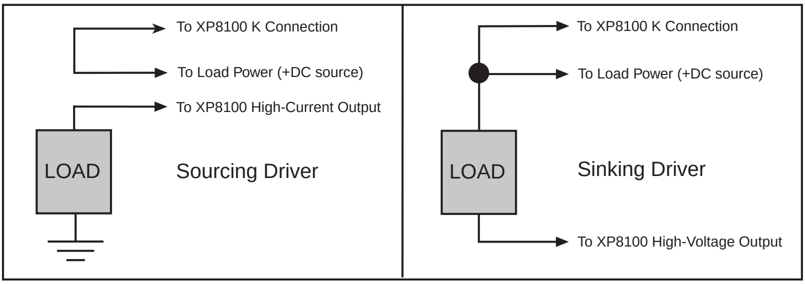

To XP8100 K Connection
To Load Power (+DC source)
To XP8100 High-Current Output
LOAD
Sourcing Driver
To XP8100 K Connection
To Load Power (+DC source)
LOAD
Sinking Driver
To XP8100 High-Voltage Output

Figure D-6. XP8100 K Connections

APPENDIX E: FIELD WIRING TERMINALS

Discrete input/output lines may be connected to any of the XP8100 Series expansion boards with field wiring terminal (FWT) modules. This eliminates the need for ribbon cables. The optional quick-disconnect modules provide screw terminals for simple wiring.

Each module mates to two of the XP8100 Series board headers (H1–H2 and H3–H4). This is equivalent to 16 connections per module. One XP8100 Series expansion board can accept up to two FWT modules in any combination. The FWT50, FWT38, and the FWT-Opto modules are available.

Figures E-1 and E-2 show the mounting configuration for the FWT modules.

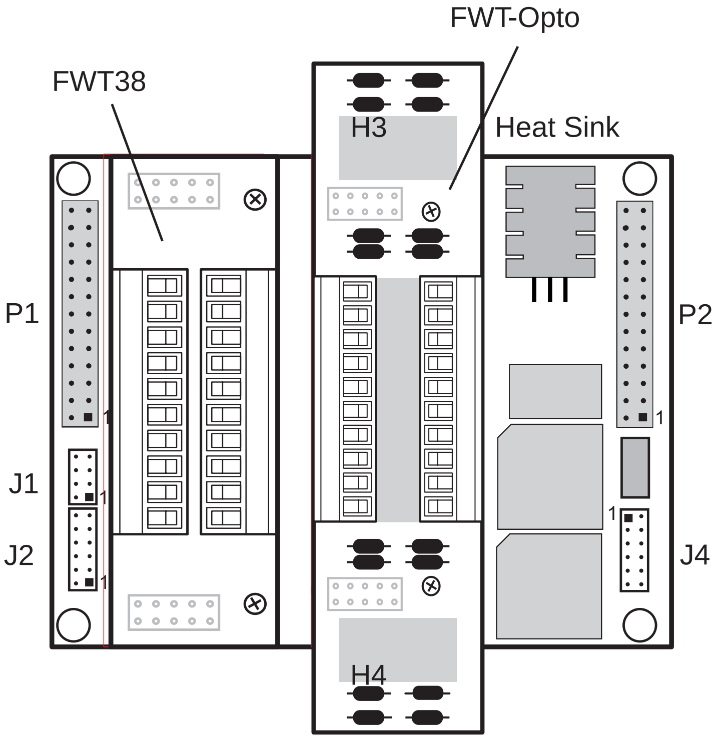


Figure E-1. Top View of XP8100 with FWT Modules

The four FWT styles described in this appendix are available from Z-World. Your application may use a different arrangement than that shown in Figure E-1.

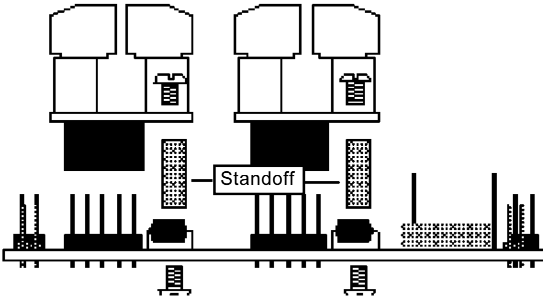


Figure E-2. FWT Installation

FWT38

The FWT38 has 20 terminals in two groups with 10 terminals each. Each group of terminals may be removed independently.

Table E-1 summarizes the specifications for the FWT38.

Table E-1. FWT38 Specifications

Parameter	Specification
Total I/O Channels	16
Screw Terminal Pitch	3.81 mm
Maximum Wire Gauge	28-16 AWG
Quick-Disconnect Capability	Wiring banks can be unplugged from the board separately (Phoenix Combicon type connection)
Wire Orientation	Top-exiting wires

Figure E-3 provides the dimensions for the FWT38.

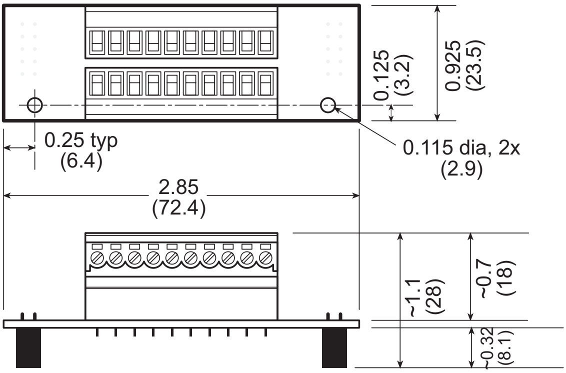


Figure E-3. FWT38 Dimensions

Figure E-4 shows the I/O channel assignments and pinouts for the FWT38.

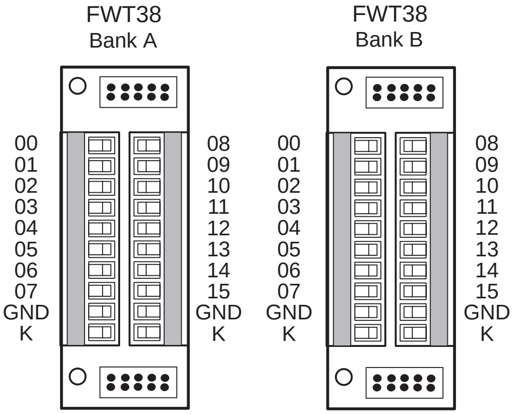


Figure E-4. FWT38 Pinouts

FWT50

The FWT50 provides 20 screw terminals. The terminal connectors are fixed to the FWT module and cannot be removed.

Table E-2 summarizes the specifications for the FWT50.

Table E-2. FWT50 Specifications

Parameter	Specification
Total I/O Channels	16
Screw Terminal Pitch	5.00 mm
Maximum Wire Gauge	24-12 AWG
Quick-Disconnect Capability	Unplugs from the XP8100 board as a single unit
Wire Orientation	Side-exiting wires

Figure E-5 provides the dimensions for the FWT50.

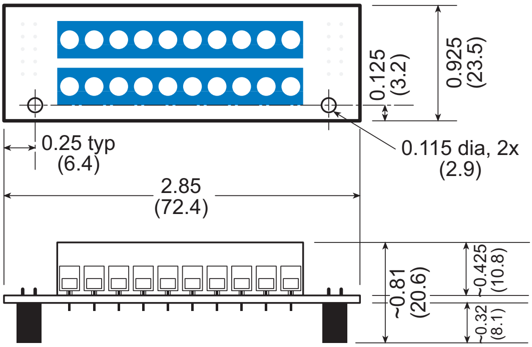


Figure E-5. FWT50 Dimensions

Figure E-6 shows the I/O channel assignments and pinouts for the FWT50.

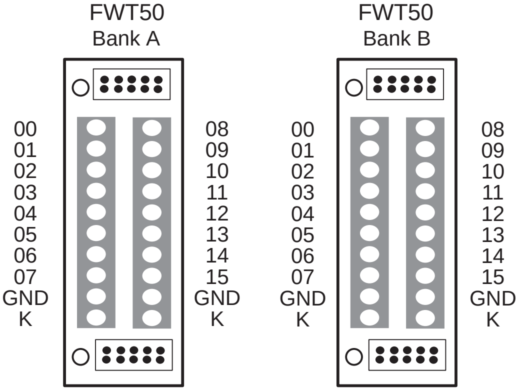


Figure E-6. FWT50 Pinouts

FWT-Opto

The FWT-Opto provides optical isolation to the input channels. The FWT-Opto is used only for inputs, and is not used with the XP8120 expansion board. All 16 channels must be committed to inputs when an FWT-Opto module is used.

Every four FWT-Opto inputs share a common return. The excitation resistors need to be pulled up to +5 V when the FWT-Opto module is used.

Table E-3 lists the specifications for the FWT-Opto module.

Table E-3. FWT-Opto Specifications

Parameter	Specification
Total Input Channels	16 optically isolated input channels only
Screw Terminal Pitch	3.81 mm
Maximum Wire Gauge	28-16 AWG
Quick-Disconnect Capability	Wiring banks can be unplugged from the board separately (Phoenix Combicon type connection)
Wire Orientation	Top-exiting wires
Input Protection Range	5 kV rms between input and output
Maximum Input Voltage	±40 V
Guaranteed Input Switching Threshold	±9.5 V

The FWT-Opto module uses 4.7 kΩ input resistors to accommodate the large range of input voltages. This limits the input switching threshold to ±9.5 V. These 4.7 kΩ input resistors need to be replaced with 1.2 kΩ input resistors to handle smaller input voltages such as 5 V logic. If 0.125 W resistors are used, this will limit the maximum input voltage to ±12.2 V.

Figure E-7 provides the dimensions for the FWT-Opto module.

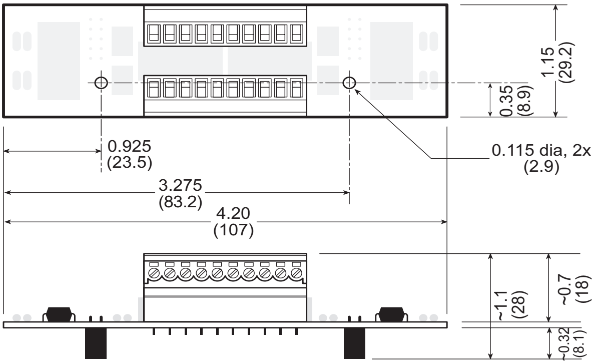


Figure E-7. FWT-Opto Dimensions

Figure E-8 shows the input channel assignments and pinouts for the FWT-Opto module.

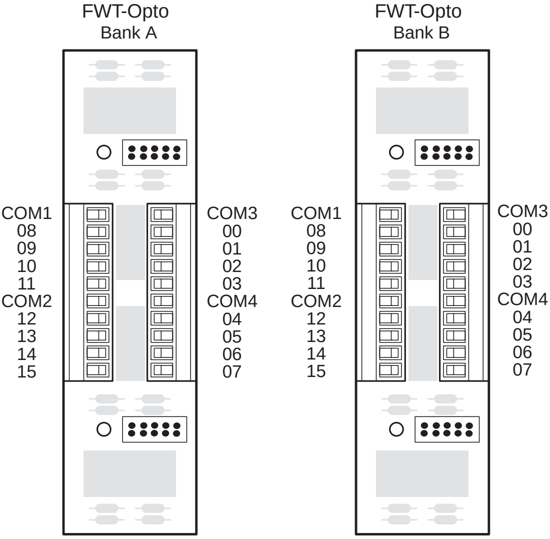


Figure E-8. FWT-Opto Pinouts

Figure E-9 shows an FWT-Opto optical isolation circuit.

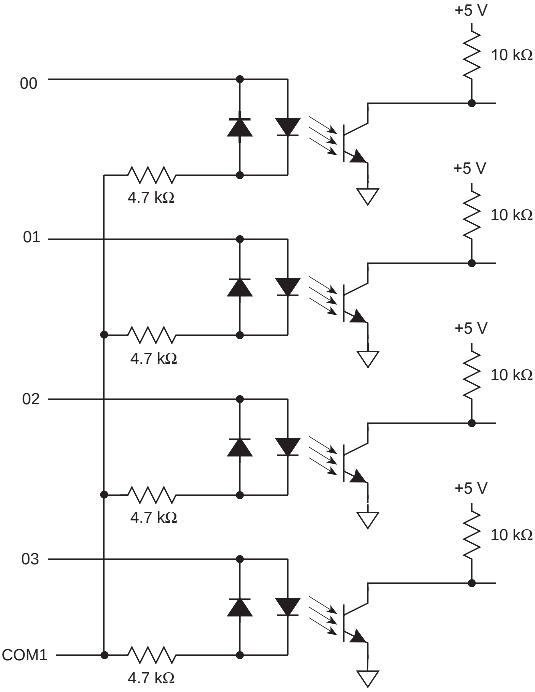


Figure E-9. FWT-Opto Optical Isolation Circuit

The opto-isolated inputs share a common return in groups of four. The software channel assignments remain the same for Banks A and B.

APPENDIX F:

SIMULATED PLCBUS CONNECTIONS

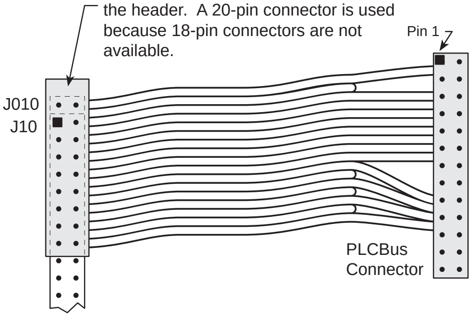
BL1100

The XP8200 expansion board may be connected to a BL1100 using an expander cable (Z-World part number 540-0015). Fasten the cable's 20-pin connector to the combined headers J010 and J10 as shown in Figure F-1. Pins 1 and 2 of the expander cable connector must hang over the end of the combined headers. Fasten the cable's PLCBus connector to XP8200 header P1 or P2. Note the orientation of pin 1.

Software for interfacing the BL1100's PIO port to a PLCBus port may be found in the Dynamic C **`PBUS_LG.LIB`** library.

Picks up VCC, GND, and PA0–PA7. Leaves PB0–PB7 available.

PIO Signal	PLCBus Signal
PA0 (J10:1)	/STBX
PA1 (J10:3)	A3X
PA2 (J10:5)	A2X
PA3 (J10:7)	A1X
PA4 (J10:9)	D2X
PA5 (J10:11)	D3X
PA6 (J10:13)	D0X
PA7 (J10:15)	D1X
+5 V (J010:1)	+5 V



Note that the first two pins of this connector must hang over the end of the header. A 20-pin connector is used because 18-pin connectors are not available.

Figure F-1. BL1100 Expander Cable Connection

Use an external power supply when connecting expansion boards to the BL1100. There is no provision in the expander cable to supply +24 V from the controller to header P1 or P2 on the XP8200.

BL1400 or BL1500

XP8200 expansion boards may be connected to header H3 on either the BL1400 or the BL1500. To add XP8200 expansion boards, the user must either make a custom cable or use an adapter board (Z-World part number 101-0050). To assist with making the connection via a ribbon cable, Table F-1 maps the signals from the controller's PIO to the XP8200 PLCBus. Dynamic C's **`EZIOMGPL.LIB`** library may be used for programming.

Table F-1. PIO to PLCBus Signal Map

BL1400/Bl1500		Expansion Board	
H3 Pin No.	**PIO Port Signal**	**Pin No.**	**PLCBus Signal**
1	VCC (+5 V)	2	VCC (+5 V)
2	PA0	5	/STBX
3	PA1	19	D0X
4	PA2	20	D1X
5	PA3	17	D2X
6	PA4	18	D3X
7	PA5	11	A1X
8	PA6	9	A2X
9	PA7	7	A3X
10	GND	10	GND

The adapter board provides an easy way to add an XP8200 to either BL1400 or BL1500 controllers. Power is supplied to the controller via the power jack and to the expansion board via a screw terminal. For specifics on how to install an adapter board with a specific controller, see that controller's user's manual.

Use the appropriate external voltage supply with expansion boards connected to the BL1400 and BL1500.

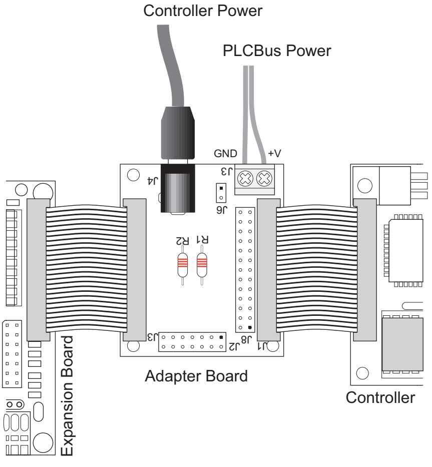


Figure F-2. Adapter Board Connections

Blank

Appendix G: PLCBus States

PLCBus State Tables

This appendix is provided for advanced programmers who wish to write their own application drivers, and require detailed information about PLCBus cycles.

Two state tables are provided. Table G-1 presents the PLCBus states, and Table G-2 describes what state the PLCBus transitions to from a given input.

Table G-1. State Definitions

State	State Number (see Table G-2)
0	Board not selected
1	BUSADR0 recognized
2	BUSADR1 recognized
3	BUSADR2 recognized for ID mode
4	BUSADR2 recognized for Group 0
5	BUSADR2 recognized for Group 1
6	BUSADR2 recognized for Group 2
7	BUSADR2 recognized for Group 3

Reading State Table G-2

The letters **`pqr`** represent the binary version of the jumper-set address of the XP8100 Series board (**`r`** is the least-significant bit). The letters **`xyz`** represent the group number for data from an I/O channel, and **`efgh`** represent the data bits D3–D0.

To use Table G-2, read across the "state" row to the current state of the PLCBus. Then read down the "action" column to the particular PLCBus cycle to be performed in that state. The number corresponding to the next state that the PLCBus will transition to is at the intersection of the row and column. Some of the cases also have a superscripted reference to a note that explains how to interpret the value returned.

Table G-2. PLCBus State Table

	State							
Action	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
BUSADR0 ← **0001**	1	1	1	1	1	1	1	1
BUSADR1 ← **00pq**	0	2	2	2	2	2	2	2
BUSADR2 ← **r000**	0	1	3	3	3	3	3	3
BUSADR2 ← **r100**	0	1	4	4	4	4	4	4
BUSADR2 ← **r101**	0	1	5	5	5	5	5	5
BUSADR2 ← **r110**	0	1	6	6	6	6	6	6
BUSADR2 ← **r111**	0	1	7	7	7	7	7	7
BUSWR ← **efgh**	0	1	2	3	4[b]	5[b]	6[b]	7[b]
BUSDR0 → **efgh**	0	1	2	3[a]	4[c]	5[c]	6[c]	7[c]
BUSDR1 → **efgh**	0	1	2	3	4[d]	5[d]	6[d]	7[d]
BUSADR0 ← ! **0001**	0	0	0	0	0	0	0	0
BUSADR1 ← ! **01pq**	0	1	1	1	1	1	1	1
BUSADR2 ← ! **rxyz**	0	1	2	2	2	2	2	2

Notes

(a) **h**=**0** indicates an XP8100 exists; **h**=**1** indicates there is no XP8100 Series board at this address.

(b) **h**=**0** indicate off, **1** indicates on, **efg** specifies which of the 8 output channels in the group is selected.

(c) **h** indicates the state of the zeroth (lowest number) input channel in the group of 8, **e** indicates the state of the third input channel in the group of 8.

(d) **h** indicates the state of the fourth input channel in the group of 8, **e** indicates the state of the seventh input channel in the group of 8.

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. 019-0045
Revision D

Printed in U.S.A.