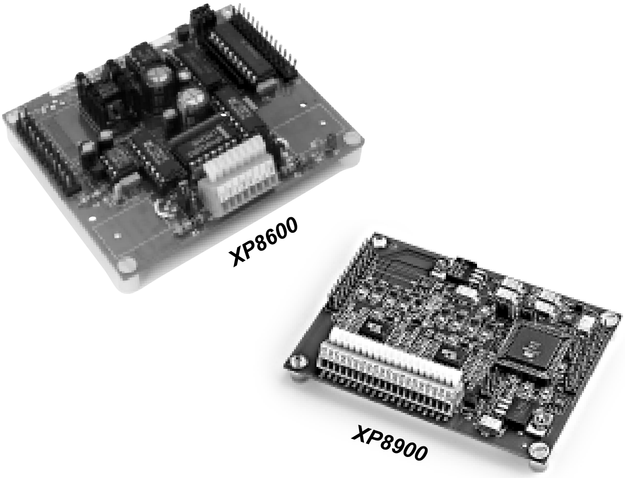
Z·WORLD



XP8600 and XP8900

Digital-to-Analog Conversion Expansion Boards

User’s Manual

Revision C

XP8600 and XP8900 User’s Manual

Part Number 019-0064 • Revision C
Last revised on July 21, 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

XP8900

APPENDICES

Blank

About This Manual

This manual provides instructions for installing, testing, configuring, and interconnecting the Z-World XP8600 and XP8900 Series digital-to-analog conversion 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 the controller used with the XP8600 or XP8900 Series expansion boards will control.
- Understanding of the basics of operating a software program and editing files under Windows on a PC.
- Knowledge of the basics of C programming.

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

  The C Programming Language by Kernighan and Ritchie
  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

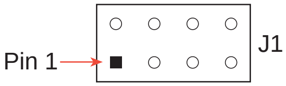
Table 3 lists and defines the typographical conventions that may be used in this manual.

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

XP8600

Blank

CHAPTER 1: OVERVIEW

Chapter 1 provides an overview and description of the XP8600 digital-to-analog conversion expansion boards.

The XP8600 is a 12-bit digital-to-analog (D/A) converter expansion board that can be used in conjunction with any Z-World PLCBus-compatible controller.

Like other Z-World expansion boards, the XP8600 can be installed in modular plastic circuit-board holders attached to a DIN rail. The XP8600 can also be mounted, with plastic standoffs, on any surface that will accept screws. Eight or more different XP8600 board addresses may be used on one PLCBus; up to 64 different logical addresses are provided for.

The XP8600 has two D/A output channels. Each channel can be operated either as a voltage source or a current source. The AD7543 D/A chips have a linearity of ±1 LSB and a gain stability of about 5 ppm per degree Celsius.

A factory-installed LT1021 provides a precision 10 V reference when used with 24 V controllers. An optional LT1019 provides a precision 2.5 V reference for 12 V controllers.

☎ Z-World offers the XP8600 for quantity orders with the optional LT1019 factory-installed. For more information, call your Z-World Sales Representative at (530) 757-3737.

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

CHAPTER 2: GETTING STARTED

Chapter 2 provides instructions for connecting XP8600 expansion boards to a Z-World controller. The following sections are included.

- XP8600 Components
- Connecting Expansion Boards to a Z-World Controller
- Setting Expansion Board Addresses
- Power

XP8600 Components

The XP8600 boards offer two channels of 12-bit digital-to-analog conversion outputs. Figure 2-1 illustrates the basic layout and orientation of components, headers, and connectors.

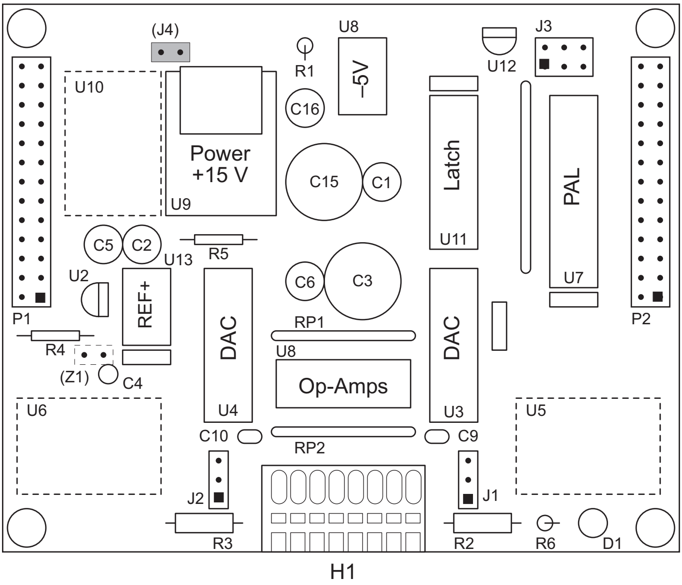


Figure 2-1. XP8600 Board Layout

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 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.

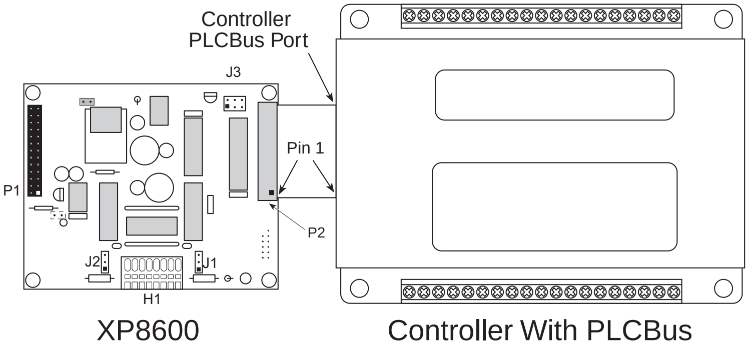


Figure 2-2. Connecting XP8600 Expansion Board to Controller PLCBus

⚠ 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.

4. Each expansion board comes with a factory-default board address. If more than one expansion board of each type is to be used, be sure to set a unique address for each board.

The following section on “Setting Expansion Board Addresses,” and Chapter 4, “Software Reference,” provide details on how to set and use expansion board addresses.

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

See Appendix D, “Simulated PLCBus Connections,” for details on the special connections that enable these expansion boards to be used with the BL1000, BL1100, BL1400, and BL1500 controllers.

Setting Expansion Board Addresses

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

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.

XP8600 expansion boards are shipped from the factory with no pins on header J3 connected. Each XP8600 can have one of eight different PALs. There are eight different ways to configure the three pairs of pins on header J3, and so up to 64 different logical XP8600 addresses are possible for a single PLCBus.

See Chapter 4, “Software Reference,” for further details on how to determine the physical address for XP8600 expansion boards based on which pins on header J3 are connected.

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 and the ±10 V reference used by the expansion boards. With no output, the XP8600 expansion boards draw about 30 mA; with all their output channels operating at maximum current (22 mA per channel), the XP8600 draws 75 mA.

Blank

Chapter 3: I/O Configurations

Chapter 3 describes the built-in flexibility of the XP8600 expansion boards, and describes how to configure the available inputs/outputs. The following sections are included.

- XP8600 Pin Assignments
- XP8600 Circuitry

XP8600 Pin Assignments

Analog signals (voltage or current output) leave the XP8600 via terminals 1 and 3 on Wago connector H1. Terminals 2 and 4 provide ground. Other terminals provide access to board voltages.

When the XP8600 is used with a BL1200, BL1600, BL1700, PK2100, or PK2200 controller, the +24 V from the PLCBus serves as the power source, and is accessible on pin 8 of Wago connector H1. When the XP8600 is used with BL1000, BL1100, BL1300, BL1400, or BL1500 controllers, the +24 V is not available and must be supplied from an external source to pin 8 on H1 or to the BL1400/BL1500 adapter board.

Figure 3-1 shows the pin assignments for Wago connector H1.

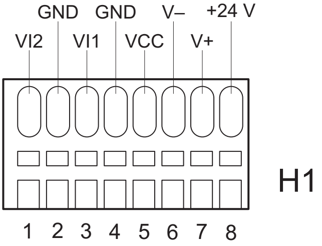


Figure 3-1. XP8600 Wago Connector H1

Voltage or Current Output

The XP8600 uses two jumper blocks, J1 and J2, to set the mode of output channel 1 (VI1 on H1) and output channel 2 (VI2 on H1), respectively. Connect pins 2–3 to select a voltage output for the channel corresponding to the jumper block, and connect pins 1–2 to select a current output.

Figure 3-2 summarizes the jumper settings.

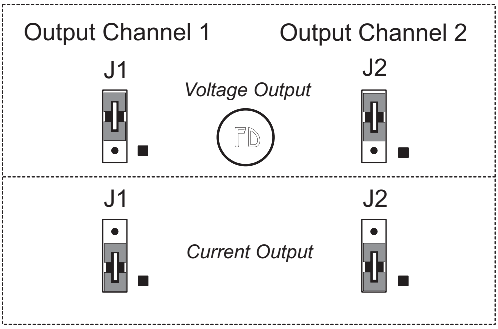


Figure 3-2. XP8600 Output Channel Jumper Settings

XP8600 Circuitry

The XP8600's D/A circuitry consists of two 12-bit AD7543 D/A converters, U3 and U4, and an LM324N quad op-amp chip, U1. The outputs of the D/A converters are amplified, and resistor packs RP1 and RP2 allow both current output and voltage output. The outputs appear on headers J1 and J2, and at Wago connector H1. The input comes on the PLCBus from the program running on the controller.

Figure 3-3 illustrates the operation of the D/A conversion.

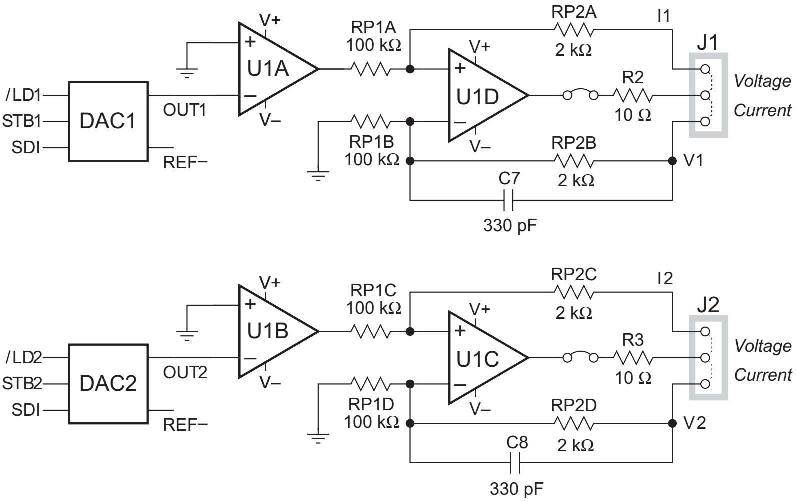


Figure 3-3. Schematic Illustration of D/A Conversion in XP8600

Capacitors C7 and C8 are normally not installed. Be sure to install C7 and C8 for applications that operate at more than ~100 Hz.

The AD7543 Chip

Each of the two AD7543 D/A converter chips receives serial data in Register A. When Register A is full, its contents are transferred to Register B, as shown in Figure 3-4. The data are then converted and the analog output is asserted.

Each AD7543 has a linearity of ±1 LSB and a gain stability of approximately 5 ppm per degree Celsius.

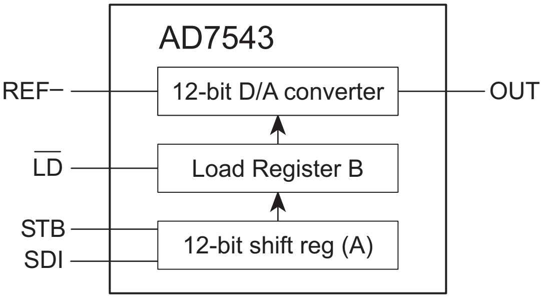


Figure 3-4. Operation of AD7543 D/A Converter Chip

The output of each D/A converter chip is given by Equation (3-1).

$$\text{OUT} = -(\textit{input}/4096) \times \text{REF–} \qquad (3\text{-}1)$$

The D/A converter output voltage ranges from 0 to REF–. REF– is 10 V when U13 is an LT1021 (the factory default).

REF- is 2.5 V when the LT1019 is used at U13 to allow the XP8600 to be used with 12 V controllers.

Voltage Output

The voltage output is selected by connecting pins 2–3 on header J1 or J2. (It is possible to operate one channel with a voltage output and the other channel with a current output.) The voltage output can be calculated using Equation (3-2).

$$V = \text{OUT} \times (1 + \text{RP2}/\text{RP1}) \qquad (3\text{-}2)$$

where OUT is the output of the D/A converter chip. The output values can be changed by changing resistor packs RP1 and RP2. The maximum output voltage is limited by the op-amp to 14 V.

The op-amps U1C (Channel 1) and U1D (Channel 2) can carry a maximum of 20 mA. When using a channel in the voltage-output mode, its load must have a resistance of at least 700 Ω to keep the op-amp from saturating prematurely.

Current Output

A current output is selected by connecting pins 1–2 on jumper block J1 or J2. (It is possible to operate one channel with a voltage output and the other channel with a current output.) The current output can be calculated using Equation (3-3).

$$\mathrm{I1} = (\mathrm{OUT1}/\mathrm{R2}) \times (\mathrm{RP2}/\mathrm{RP1})$$

$$\mathrm{I2} = (\mathrm{OUT2}/\mathrm{R3}) \times (\mathrm{RP2}/\mathrm{RP1})$$

where OUT1 and OUT2 are the outputs of the D/A converter chips. The output values can be changed by changing resistor packs RP1 and RP2. The maximum output current is limited by the op-amp to 20 mA.

(3-3)

The op-amps U1C (Channel 1) and U1D (Channel 2) can carry a maximum of 20 mA. The output channels can each put out a maximum of 14 V, and so an output load must be less than 700 Ω if the maximum current is to be delivered.

Reset and Power-Up Delay

The XP8600 has a reset device, U12, to monitor VCC. Whenever VCC is interrupted, U12 pulls /RST low. The /RST line remains low until ~350 ms after VCC has been restored, allowing time for the power supply and the D/A circuits to stabilize. U12 also monitors the /RST line. When it detects a low-going edge, U12 pulls /RST low for at least 350 ms. Thus, the board may be reset with either software or a pushbutton. Software must allow a 350 ms delay after each reset or power-up cycle.

Blank

CHAPTER 4: SOFTWARE REFERENCE

Chapter 4 describes the Dynamic C functions used to initialize the XP8600 and XP8900 Series expansion boards and to control the resulting analog outputs. The following major sections are included.

- Expansion Board Addresses
- XP8600 Software
- Advanced Programming

Expansion Board Addresses

XP8600

Up to 64 XP8600s may be addressed individually over a single PLCBus. Each XP8600 board has a 12-bit address. Once the board has latched its address, the board may be accessed repeatedly without having to send its address again.

The 12-bit address of a particular XP8600 is determined by the encoding of PAL chip U7 on the board and by jumper block J3. Eight different PALs are available, and J3 may be set eight different ways, giving 64 unique addresses in the form

000z 001y pppx

where

x = 1 when J3 pins 1–2 are not connected
y = 1 when J3 pins 3–4 are not connected
z = 1 when J3 pins 5–6 are not connected

and the PAL determines ppp.

The 12-bit address can be placed on the bus using 4-bit addressing. The functions **`set12adr`**, **`read12data`**, and **`write12data`** (in **`DRIVERS.LIB`**) use 12-bit bus addresses.

When using these, and certain other functions, swap the first and third nibbles of the physical address before passing the address to the function. For example, if the address is **`0x125`**, pass **`0x521`**.

Logical Addresses

PLCBus expansion boards have "logical addresses." D/A-specific software defines 64 integer board addresses, 0–63. The formula mapping physical address to logical address is

logical address = ppp × 8 + zyx

where ppp (PAL encoding) and x, y and z (jumper bits) are defined above. For example, an XP8600 has PAL FPO4850 (ppp = 101) and J3 pins 3–4 connected (xyz = 101). Its physical address is

000z 001y pppx = 0001 0010 1011 = **`0x12B`**.

The XP8600's logical address is

$101_B \times 8 + 101_B = 45 =$ **`0x2D`**.

Certain library functions expect a logical D/A address.

LED

The XP8600 has an LED, D1, which is illuminated whenever the board is selected.

XP8600 Software

This section describes a set of simple software functions to use when controlling the XP8600 expansion boards.

Dynamic C Libraries

Several Dynamic C function libraries need to be used with the routines defined in this section. The chart in Table 4-1 identifies which libraries are used with particular Z-World controllers.

Table 4-1. Dynamic C Libraries Required by Z-World Controllers for XP8600 Expansion Boards

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 **PLC_EXP.LIB**, be sure there is a line at the beginning of the program in the following format.

```
#use PLC_EXP.LIB
```

Using Digital-to-Analog Converter Boards

The follow steps summarize how to use the D/A converter boards.

1. Send a reset command to the PLCBus.
2. Place the address of the D/A converter on the PLCBus.
3. Send data serially to one of the D/A converters (Register A). When Register A is filled, load the data to D/A converter Register B where it is converted and output.
4. Use the board's analog output to control motors, attenuators or other analog devices.

These steps are done using software drivers in Dynamic C function libraries.

Reset Boards on PLCBus

These Dynamic C functions are used to initialize the PLCBus. Use these functions in a program before introducing any code to operate the relays.

- **`VdInit()`**

  Initializes the timer mechanism.

  LIBRARY: **`VDRIVER.LIB`**

- **`void plcBusReset()`**

  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: **`EZIOPBDV.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. Call this function 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.

Address Target Board

- **`int plcXP86Init( int Addr )`**

Initializes and turns on XP8600. Call this function before calling **`plcXP86Out`**.

PARAMETERS: **`Addr`** is the logical address of the board set by jumpers, and ranges from 0–7. With special PALs, the address can range from 0–63.

RETURN VALUE: –1 if the board cannot be found, 0 if the initialization is completed.

LIBRARY: **`EZIOPBDV.LIB`**

Operate Target Board

- **`int plcXP86Out( int Addr, unsigned int oValue )`**

Sends the 12-bit **`oValue`** to the proper D/A converter channel. Call **`plcXP86Init`** before calling **`plcXP86Out`**.

PARAMETERS: **`Addr`**, the logical address of the D/A converter channel, is **`2*board_number + channel_number`**. Note that **`board_number`** and **`channel_number`** start from zero. Without a special PAL, **`board_number`** ranges from 0 to 7 as set by the address jumpers. With the special PAL, **`board_number`** can range from 0 to 63. **`channel_number`** ranges from 0 to 1.

`oValue` is the 12-bit value to send to the D/A converter.

RETURN VALUE: -1 if the D/A converter cannot be found, 0 if the operation is successful.

LIBRARY: **`EZIOPBDV.LIB`**

Sample Program

The sample program **`XP86_1.C`** in the Dynamic C **`SAMPLES\PLCBUS`** subdirectory demonstrates higher level function calls for the D/A converter outputs. The program is used to detect XP8600 boards attached to the PLCBus, and maximizes the D/A converter outputs.

Use the following steps to run the sample program.

1. 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.
2. Run the program by pressing **F9** or by choosing **Run** from the **RUN** menu. It is also possible to single-step through the program with **F7** or **F8**.
3. To halt the program, press **<CTRL-Z>**.
4. To restart the program, press **F9**.

XP86_1.C

```
#use eziocmmnn.lib
#use eziobl17.lib
#use ezioplc2.lib

char TITLE[] = "XP86xx DAC Output";

main(){
   int i;
   unsigned long delayCounter;
   printf("%s\n\n", TITLE);
   plcBusReset();           // reset the PLCBs
   for (delayCounter = 0x8000; delayCounter--;)
      hitwd();              // delay
   // locate all possible jumper-set addresses
   // from 0 to 7 and display status
   for (i = 0; i <= 7; ++i) {
      if (plcXP86Init(i)==-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);
        plcXP86Out(i*2,0x0ffff);
      }
      printf("\n");
      hitwd();
   }
}
```

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.

Advanced Programming

Functions in PLC_EXP.LIB

- **`int plcdac_addr( int bd )`**

  Converts logical board address 0–63 to 12-bit (3-nibble) PLCBus analog address.

  RETURN VALUE: 12-bit (nibble-interchanged) bus address for an XP8600 identified by a logical address (0–63).

- **`void plc_fdac1( int dac_value )`**
  `void plc_sdac1( int dac_value )`

  Each of these routines writes data serially to DAC1 (U3) on the presently addressed XP8600. These functions are equivalent and may be used interchangeably. However, **`plc_fdac1`** is slightly faster, whereas **`plc_sdac1`** is smaller (50 bytes vs. 130). A call to one of these functions must be followed by a call to **`plc_latch1`**—or its equivalent in user-supplied code—otherwise no digital-to-analog conversion occurs.

- **`void plc_fdac2( int dac_value )`**
  `void plc_sdac2( int dac_value )`

  Each of these routines writes data (serially) to DAC2 (U4) on the presently addressed XP8600. These functions are equivalent and may be used interchangeably. However, **`plc_fdac2`** is slightly faster, whereas **`plc_sdac2`** is smaller (50 bytes vs. 130). A call to one of these functions must be followed by a call to **`plc_latch2`**—or its equivalent in user-supplied code—otherwise no digital-to-analog conversion occurs.

- **`void dac_latch1()`**

  Transfers the data in Register A of DAC1 (U3) to Register B of DAC1, whereupon the data are converted and the analog signal is output. A call to this function must be preceded by a call to **`plc_fdac1`** or **`plc_sdac1`**—or their equivalent in user-supplied code.

- **`void dac_latch2()`**

  Transfers the data in Register A of DAC2 (U3) to Register B of DAC2, whereupon the data are converted and the analog signal is output. A call to this function must be preceded by a call to **`plc_fdac2`** or **`plc_sdac2`**—or their equivalent in user-supplied code.

- **`void dac_init()`**

  Initializes the presently addressed XP8600. Sets both D/A converter outputs to zero. Board must be currently addressed on PLCBus with **`set12adr()`**.

Sample Program

The sample program demonstrates how to drive the XP8600 in a voltage-output mode. The program runs on controllers with a PLCBus.

Materials Required

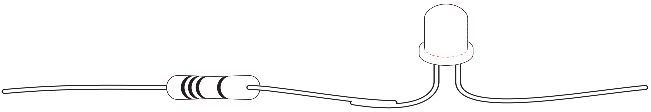
- 1 kΩ, ¼ W resistor and an LED

  OR

- digital volt-ohm meter

Instructions

1. Power up the controller and make sure it is working properly. Consult the controller user's manual, if necessary. Now disconnect power from the controller.
2. Connect the XP8600 to the controller. See Chapter 2, "Getting Started," for more details.
3. Check headers J1 through J3. Leave J3 unjumpered. Connect pins 2–3 on headers J1 and J2 (for voltage output).
4. Wire-wrap or solder together the resistor and the LED, as shown below, to make a load circuit 3" to 4" (7 cm to 10 cm) long.



5. Connect the resistor end of the load circuit to V/I1 on header H1. Connect the LED to GND on H1. Figure 4-1 shows the pins on Wago connector H1. Alternatively, the volt-ohm meter may be connected between V/I1 and GND.

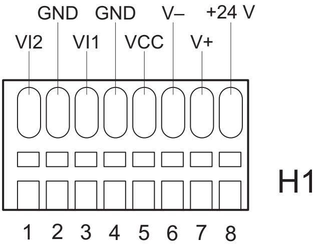


Figure 4-1. XP8600 Wago Connector H1

6. Power up the controller and bring up Dynamic C on the PC.
7. Open and run the sample program **DASAMPL1.C** in the Dynamic C **SAMPLES\PLCBUS** subdirectory .
8. The LED will increase and decrease in intensity as the program runs.
9. If the LED does not light up at all, its polarity may be reversed. Try turning the circuit around, connecting the LED at V/I1 and the resistor at GND.

Program

The program demonstrates how to drive the XP8600 in voltage output mode. The program first polls the PLCBus, looking for XP8600s. It saves the highest address found. The program then increments the voltage at both output channels on the selected board. When it reaches maximum voltage, it resets the voltage to zero and repeats the process indefinitely.

The setup requires a 1 kΩ resistor and LED in series—or a volt-ohm meter—between V/I1 and GND on header H1 to show the behavior of the program.

DASAMPL1.C

```
main(){
   int i, log_addr, hi_addr;
   reset_pbus();            // reset all boards on PLCBus
   hitwd();
   reset_pbus_wait();       // delay after reset

// Locate XP8600s; save highest addr found

   for( log_addr=0; log_addr<64; log_addr++ ){
      if( find_dac(log_addr) ) hi_addr = log_addr;
   }

// Set PLCBus to XP8600, initialize DAC

   set12adr( plcdac_addr(hi_addr) );
   dac_init();              // initialize both channels

// Increment Voltages on Channels out 1 and 2

   while(1){
      for( i=0; i<4097; i++ ){
        plc_fdac1(i);       // fdac1 (fast but big),
                            // writes to DAC channel 1
        dac_latch1();       // convert data
        plc_sdac2(i);       // sdac2 (slower, smaller),
                            // write to DAC channel 2
        dac_latch2();       // convert data
        hitwd():
      }
   }
}
```

Use the following steps to run the sample program.

1. 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.
2. Run the program by pressing **F9** or by choosing **Run** from the **RUN** menu. It is also possible to single-step through the program with **F7** or **F8**.
3. To halt the program, press **<CTRL-Z>**.
4. To restart the program, press **F9**.

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.

Functions in PBUS_LG.LIB

- **`int DAC_Board_Addr( int bd )`**

  Converts a logical D/A converter board address to a physical PLCBus address.

  PARAMETER: **`bd`** must be a number between 0 and 63 representing the D/A converter board to access. This number has the binary form pppzyx where ppp is determined by the board PAL number and x, y, and z are determined by jumper block J3 on the board. ppp values of 000, 001, 010, etc., correspond to PAL numbers of FPO4800, FPO4810, FPO4820, etc.; x, y, and z correspond to jumper block J3 pins 1–2, 3–4, and 5–6, respectively (0 = closed, 1 = open). The resulting address is in the form pppx001y000z.

  RETURN VALUE: Nibble-interchanged PLCBus address of the board specified. This address may be passed directly to **`PBus12_Addr`**.

- **`void write_DAC1( int val )`**

  Writes data serially to DAC1 (U3) on the presently addressed XP8600. The board address must have been set previously with a call to **`PBus12_Addr`**. A call to one of these functions must be followed by a call to **`latch_DAC1`**—or its equivalent in user-supplied code—otherwise no digital-to-analog conversion occurs.

- **`void write_DAC2( int val )`**

  Writes data serially to DAC2 (U4) on the presently addressed XP8600. The board address must have been set previously with a call to **`PBus12_Addr`**. A call to one of these functions must be followed by a call to **`latch_DAC2`**—or its equivalent in user-supplied code—otherwise no digital-to-analog conversion occurs.

- **`void latch_DAC1()`**

  Transfers the data in Register A of DAC1 (U3) to Register B of DAC1, whereupon it is converted and the analog signal is output. The board address must have been set previously with a call to **`PBus12_Addr`**. A call to this function must be preceded by a call to **`write_DAC1`** or its equivalent in user-supplied code.

- **`void latch_DAC2()`**

  Transfers the data in Register A of DAC2 (U4) to Register B of DAC2, whereupon it is converted and the analog signal is output. The board address must have been set previously with a call to **`PBus12_Addr`**. A call to this function must be preceded by a call to **`write_DAC2`** or its equivalent in user-supplied code.

- **`void Init_DAC()`**

  Initializes the presently addressed XP8600, setting both outputs to zero. The board address must have been set previously with a call to **`PBus12_Addr`**.

- **`void Set_DAC1( int val )`**
  `void Set_DAC2( int val )`

  Combines the functions of **`write_DACx`** and **`latch_DACx`** into a single function for convenience. The board address must have been set previously with a call to **`PBus12_Addr`**.

Functions in Other Libraries

Functions in the other libraries listed in Table 4-1 are equivalent to those listed above except for minor internal details.

Blank

XP8900

Blank

CHAPTER 5: OVERVIEW

Chapter 5 provides an overview and description of the XP8900 digital-to-analog conversion expansion boards.

The XP8900 Series is a 12-bit digital-to-analog (D/A) converter expansion board that can be used in conjunction with any Z-World PLCBus-compatible controller.

Like other Z-World expansion boards, the XP8900 Series boards can be installed in modular plastic circuit-board holders attached to a DIN rail. The XP8900 Series boards can also be mounted, with plastic standoffs, on any surface that will accept screws. Up to eight different XP8900 board addresses may be used on one PLCBus.

The XP8900 Series consists of two boards, the XP8900 with eight channels of D/A converter outputs, and the XP8910 with four channels of D/A converter outputs. Each channel produces a bipolar output of up to ±10 V DC.

The XP8900 Series features an onboard voltage regulator for PLCBus-powered operation. The XP8900 Series has connectors for user-supplied analog voltage rails, and is able to sink or source up to 7 mA with the user rails, or up to 2 mA on its own. The D/A outputs are monotonic.

☎ An XP8900 Series board can be factory-built with one to eight D/A channels, with 8-bit or 10-bit D/A outputs, or with user-defined output voltage ranges. For more information, call your Z-World Sales Representative at (530) 757-3737.

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

CHAPTER 6: GETTING STARTED

Chapter 6 provides instructions for connecting XP8900 Series expansion boards to a Z-World controller. The following sections are included.

- XP8900 Series Components
- Connecting Expansion Boards to a Z-World Controller
- Setting Expansion Board Addresses
- Power

XP8900 Series Components

The XP8900 Series of expansion boards offers up to eight channels of digital-to-analog conversion outputs. Figure 6-1 shows the basic layout and orientation of components, headers, and connectors.

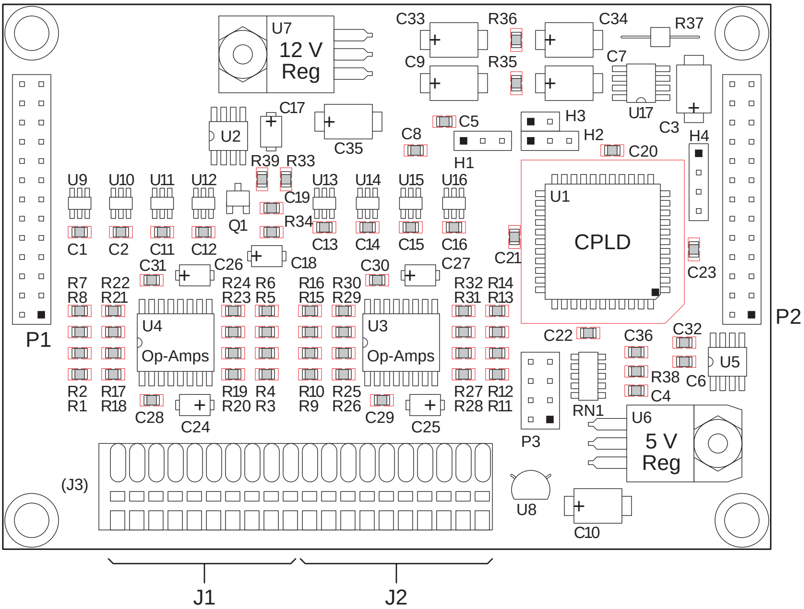


Figure 6-1. XP8900 Series Board Layout

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.

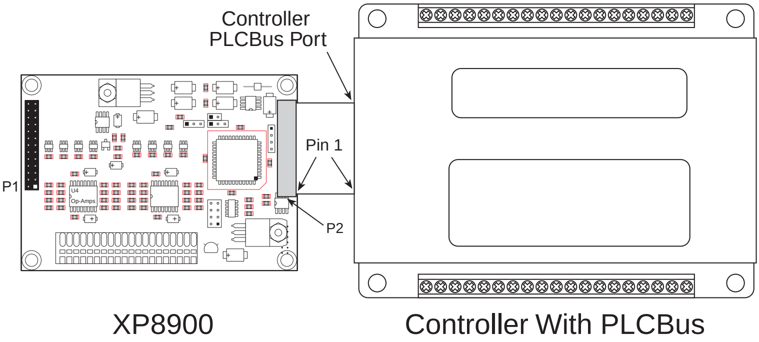


Figure 6-2. Connecting XP8600 Expansion Board to Controller PLCBus

> ⚠ 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.

4. Each expansion board comes with a factory-default board address. If more than one expansion board of each type is to be used, be sure to set a unique address for each board.

The following section on “Setting Expansion Board Addresses,” and Chapter 8, “Software Reference,” provide details on how to set and use expansion board addresses.

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

See Appendix D, “Simulated PLCBus Connections,” for details on the special connections that enable these expansion boards to be used with the BL1000, BL1100, BL1400, and BL1500 controllers.

Setting Expansion Board Addresses

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

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.

Every XP8900 Series board is shipped from the factory with a default address of 7. An XP8900 Series board may be assigned any address between 0 and 7 using jumpers on the pins of header P3 to configure the board address. Figure 6-3 shows the jumper settings to set addresses 0–7. A maximum of eight XP8900 Series boards may be addressed by a controller at one time.

Pins 1–2 on header P3 are on the lower end of P3 when the XP8900 board is oriented in line with a controller and other expansion boards as shown in Figure 6-2.

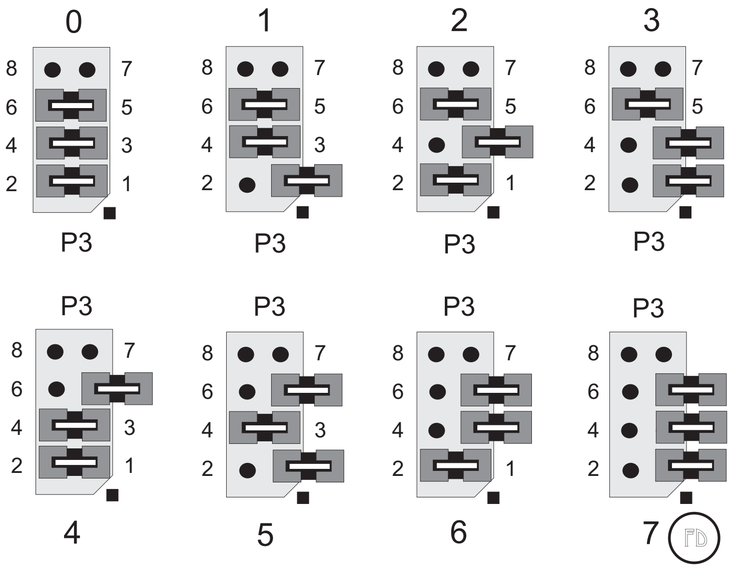


Figure 6-3. P3 Jumper Settings for XP8900 Series PLCBus Addresses

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 and the ±12 V reference used by the expansion boards. With no output, the XP8900 Series expansion boards draw about 30 mA; with all their output channels operating at maximum current (2 mA per channel on internal power, 7 mA per channel with external voltage), the XP8900 draws 45 mA (75 mA if the external power rails are connected).

Blank

CHAPTER 7: I/O CONFIGURATIONS

Chapter 7 describes the built-in flexibility of the XP8900 Series expansion boards, and describes how to configure the available inputs/outputs. The following sections are included.

- XP8900 Series Pin Assignments
- XP8900 Series Circuitry

XP8900 Series Pin Assignments

The XP8900 has eight channels of bipolar voltage outputs, each with its own individual ground, and terminals for user-supplied positive and negative voltage rails, also with their own individual grounds. These are all located on Wago connectors J1 and J2, as shown in Figure 7-1.

The pin assignments for the XP8910 are similar, except there are only four output channels. There are no outputs on pin 9 of J1, and there are no outputs on pins 1, 3, and 5 of J2.

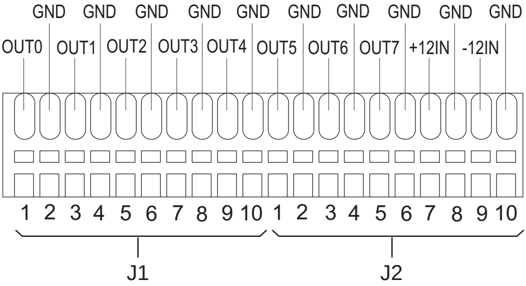


Figure 7-1. XP8900 Wago Connectors J1 and J2

No special configurations are needed for the D/A converter outputs, which are controlled by the software drivers.

An external ±12 V DC may be connected to the XP8900 Series boards to reduce analog noise or to increase the current drive. Figure 7-2 provides the jumper settings for headers H1, H2, and H3 to accommodate the external power. The external ±12 V supply is connected to the XP8900 Series board via pins 7 and 9 on Wago connector J2.

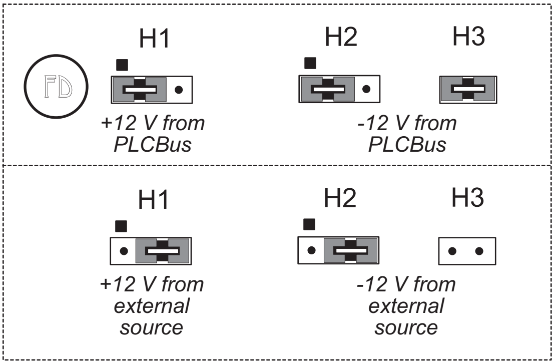


Figure 7-2. XP8900 Series ±12 V Supply Jumper Settings

XP8900 Series Circuitry

The XP8900's D/A circuitry consists of eight 12-bit AD5320 D/A converters, U9 to U16, and two OP497G quad op-amp chips, U3 and U4. The outputs of the D/A converters are amplified, and the analog outputs appear on Wago connectors J1 and J2. The input comes on the PLCBus from the program running on the controller.

Figure 7-3 illustrates the operation of the D/A conversion.

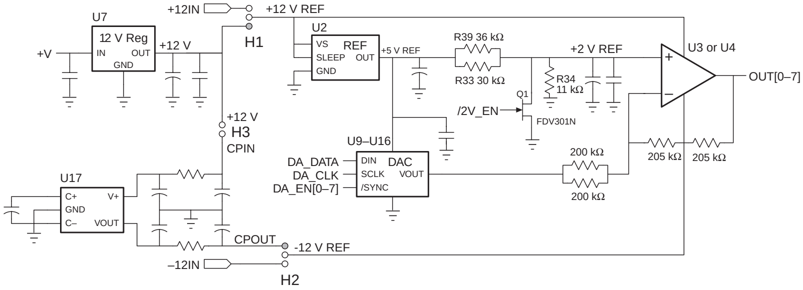


Figure 7-3. Schematic Illustration of D/A Conversion in XP8900 Series

The analog outputs do not need any special configuration. The desired analog output voltage is set using the software drivers.

The XP8900 Series expansion boards derive +5 V digital power from the +24 V PLCBus supply via LM7805 at U6. When operating without user-supplied external voltage rails, the XP8900 Series D/A converters get their +12 V analog power from the PLCBus +24 V supply via LM7812 at U7. Charge pump NJU7662 at U17 inverts this for onboard -12 V analog power. Precision +5 V and + 2 V reference voltages are derived from the +12 V supply via REF195 at U2 and the voltage divider formed from R33, R34, and R39. The n-channel FET FDV301N at Q1 is used to switch the 2 V reference to 0 V during a power-on reset.

The XP8900 Series D/A converters have the capability of receiving their +12 V or -12 V supply from an external source. This provides for greater control of electrical noise in the analog output signals.

The XP8900 Series D/A converters may be used with 12 V controllers only when ±12 V is supplied externally to pins 7 and 9 of Wago connector J2. Remember to set the jumpers on headers H1, H2, and H3 as shown in Figure 7-2.

Blank

CHAPTER 8: SOFTWARE REFERENCE

Chapter 4 describes the Dynamic C functions used to initialize the XP8600 and XP8900 Series expansion boards and to control the resulting analog outputs. The following major sections are included.

- Expansion Board Addresses
- XP8600 Software
- XP8900 Series Software

Expansion Board Addresses

XP8900 Series

Up to eight XP8900 Series expansion boards may be addressed over a single PLCBus using a logical address of 0 to 7.

The 12-bit address of a particular XP8900 is determined by the jumper setting on header P3. P3 may be set eight different ways. The unique physical address is in the form

    0010 000x yzRR

where

x = 1 when P3 pins 1–2 are not connected
y = 1 when P3 pins 3–4 are not connected
z = 1 when P3 pins 5–6 are not connected

and RR is reserved for the registers. There are no PAL codes.

The 12-bit address can be placed on the bus using 4-bit addressing. The functions **`set12adr`**, **`read12data`**, and **`write12data`** (in **`DRIVERS.LIB`**) use 12-bit bus addresses.

When the address is passed to **`set12adr`**, it should be in the format

    yzRR 000x 0010

where the least significant nibble in the physical address, yzRR, has swapped places with the most significant nibble in the physical address, 0010.

XP8900 Series Software

This section describes a set of simple software functions to use when controlling the XP8900 Series expansion boards.

Dynamic C Libraries

Several Dynamic C function libraries need to be used with the routines defined in this section. 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 for XP8900 Series Expansion Boards

Library Needed	Controller
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
EZIOBL17.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 eziopl c.lib
```

The **`#use eziopbdv.lib`** already included in other library calls for the XP8900 Series expansion boards, and does not have to be repeated.

Using Digital-to-Analog Converter Boards

The follow steps summarize how to use the D/A converter boards.

1. Send a reset command to the PLCBus.
2. Place the address of the D/A converter on the PLCBus.
3. Send data serially to one of the D/A converters (Register A). When Register A is filled, load the data to D/A converter Register B where it is converted and output.
4. Use the board's analog output to control motors, attenuators or other analog devices.

These steps are done using software drivers in Dynamic C function libraries.

Reset Boards on PLCBus

These Dynamic C functions are used to initialize the PLCBus. Use these functions in a program before introducing any code to operate the relays.

- **`VdInit()`**

  Initializes the timer mechanism.

  LIBRARY: **`VDRIVER.LIB`**

- **`void plcBusReset()`**

  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: **`EZIOPBDV.LIB`**

  The XP8900 output voltages cannot be reset by resetting the PLCBus. The rest of this chapter provides information on setting or resetting the XP8900 output voltages.

- **`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. Call this function 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.

Address Target Board

- **int plcXP89Init(int Addr)**

  Initializes XP8900 Series board. Call this function before using the other **plcXP89**... functions. This function also initializes the XP8900 Series D/A converters to tristate their outputs. Call **plcXP89Sw** to turn the voltage reference on. The first **plcXP89Out** call enables the output of the corresponding D/A converter channel. Both the voltage reference and the D/A converter channel must be set up correctly to get the proper output.

  PARAMETER: **Addr** is the logical address, 0–7, of the board set by jumpers.

  RETURN VALUE: –1 if the board cannot be found, 0 if the initialization is completed.

  LIBRARY: **EZIOPBDV.LIB**

```
void main(void){
   plcBusReset();    // reset the PLCBus
   if(plcXP89Init(4)){
   ...
   } else {
   ...
   }
}
```

Operate Target Board

- **`int plcXP89Sw( int Addr, int state )`**

  Turns the D/A converters and references to op-amps on or off. Note that all channels on a particular board are switched at the same time.

  PARAMETERS: **`Addr`** is the logical address, 0–7, of the board set by jumpers. Both the reference (switched on by this call) and the D/A converter output (switched off by this call, switched on by **`plcX89Out`**) must be set correctly to get the proper output.

  `state` indicates whether the D/A converter and reference voltage should be turned on or off. The reference is turned on when **`state`** is nonzero. Otherwise the D/A converters will tristate and the reference will output 0. The output voltage of all channels should be approximately 0 at the op-amp when the D/A converter is off.

  RETURN VALUE: –1 if the board cannot be found, 0 if the operation is completed.

  LIBRARY: **`EZIOPBDV.LIB`**

  The XP8900 output voltages may fluctuate to 2 V for each channel while **`plcX89Sw`** is executing to turn on the op-amp reference and to switch off the D/A converter.

- **`int plcXP89Out( int Addr, unsigned int oValue )`**

  Sends the 12-bit **`oValue`** to the proper D/A converter channel. Call **`plcXP89Init`** and **`plcXP89Sw`** before calling **`plcXP89Out`**. Note that **`plcXP89Out`** does not switch the voltage reference on or off. Both the D/A converter and the voltage reference must be set up correctly to get the proper voltage output. **`plcXP89Sw`** enables the voltage reference.

  PARAMETERS: **`Addr`** is *8*board_number* + *channel_number*. Note that *board_number* and *channel_number* start from zero. *board_number* ranges from 0 to 7 as set by the address jumpers. *channel_number* ranges from 0 to 7 (XP8900), or from 0 to 3 (XP8910).

  `oValue` is the 12-bit value to send to the D/A converter.

  RETURN VALUE: –1 if the D/A converter cannot be found, 0 if the operation is successful. If the D/A converter does not exist, this function also bit-ors the constant **`EIO_NODEV`** to **`eioErrorCode`**.

  LIBRARY: **`EZIOPBDV.LIB`**

```
plcXP89Out(42,2048)
     // make channel 2 on board 5 output about 0 V
```

Table 8-2 summarizes these three functions. The order in which they appear in Table 8-2 is the sequence in which they should be used to start an XP8900 Series board.

Table 8-2. Summary of Basic XP8900 Series Function Calls

Function	Description
plcXP89Init	Disables everything, leaves output of 0 V for all channels
plcXP89Sw	Enables voltage reference so the output will be at the voltage level specified by **plcXP89Out**
plcXP89Out	Sets all channels to midpoint or other acceptable value (the output experiences a slight jump as channels are being set; remember to set all 4 or 8 channels since one call sets only one channel)

-
  ```
  int plcXP89WrCalib( int chan,
                      struct _eioAdcCalib *pCalib )
  ```

  Writes a calibration structure to the EEPROM storage corresponding to a channel on the XP8900 Series board.

  PARAMETERS: **chan** is the channel number, 0–63, of the XP8900 Series D/A channel. **chan** = *8*board_number* + *channel_number*.

  _eioAdcCalib *pCalib is a pointer to a calibration structure initialized by calling **eioAdcMakeCoeff**.

  RETURN VALUE: 0 if the calibration is successful, otherwise returns a negative number.

  LIBRARY: **EZIOPBDV.LIB**

```
plcXP89WrCalib(15,&cstruct)
     // write calib info in cstruct to channel 7 of
     // XP8900 Series board 1
```

- **int plcXP89RdCalib(int chan,**
  struct _eioAdcCalib *pCalib)

  Reads the calibration structure of a D/A channel from an XP8900 Series board.

  PARAMETERS: **chan** is the channel number, 0–63, of the XP8900 Series D/A channel. **chan** = *8*board_number* + *channel_number*.

  _eioAdcCalib *pCalib is a pointer to a calibration structure. Use **eioAdcDigitize** to compute the actual D/A output of a given analog value.

  RETURN VALUE: 0 if the operation is successful, otherwise returns a negative number.

  LIBRARY: **EZIOPBDV.LIB**

```
plcXP89RdCalib(32,&cinfo)
   // read calib info of channel 0 of XP8900
   // XP8900 Series board 4 into cinfo
```

- **int eioAdcMakeCoeff(struct _eioAdcCalib *cnvrsn,**
  unsigned d1, unsigned d2, float f1, float f2)

  Takes the raw values and actual values of two data points, then computes the calibration coefficients (assumes linearity).

  PARAMETERS: **struct _eioAdcCalib *cnvrsn** is a pointer to a calibration structure that stores the coefficients.

  d1 is the raw (quantized) value of the first data point.

  d2 is the raw (quantized) value of the second data point.

  f1 is the actual (real) value (in volts) of the first data point.

  f2 is the actual (real) value (in volts) of the second data point.

  RETURN VALUE: –1 if it is not possible to compute the calibration coefficients, otherwise 0.

  LIBRARY: **EZIOPBDV.LIB**

```
eioAdcMakeCoeff(&cinfo,96,4000,9.97,-10.33)
   // the actual value at quantized value 96 is 9.97 V
   // the actual value at quantized value 4000 is -10.03 V
   // compute the coefficients and put into cinfo
```

- **long eioAdcDigitize(float f,**
  struct _eioAdcCalib *pCalib)

  Converts analog value to digital number according to calibration coefficients. This function is used to convert an analog value such as voltage to the actual digital number for a D/A converter device.

  PARAMETERS: **f** is the analog value to output.

  _eioAdcCalib *pCalib is a pointer to a structure that stores the calibration coefficients.

  RETURN VALUE: Long integer that corresponds to the number to send to a D/A converter device.

  LIBRARY: **EZIOPBDV.LIB**

```
L=eioAdcDigitize(2.54,&cinfo);
   // L will contain the digitized value to output
   // to D/A converter device given the
   // calibration coefficients in cinfo so that
   // the output is about 2.54 of some real units
```

Sample Program

The sample program **XP89_1.C** in the Dynamic C **SAMPLES\PLCBUS** subdirectory demonstrates how to calibrate the D/A converter channels.

The basic sample program is designed for the BL1200, BL1600, PK2100, and PK2200 controllers. Remember to uncomment the lines that apply to the controller being used with the XP8900 Series expansion board.

To use this program properly, it may be necessary to edit the statements that initialize the channel, margin, **f1**, and **f2**. The program may also be compiled as is, with watch expressions added to override the assignment statements (be sure to execute the watch expression AFTER the assignment statement is executed).

Use the following steps to run the sample program.

1. 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.
2. Run the program by pressing **F9** or by choosing **Run** from the **RUN** menu. It is also possible to single-step through the program with **F7** or **F8**.
3. To halt the program, press **<CTRL-Z>**.
4. To restart the program, press **F9**.

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.

XP89_1.C

```
#use eziocmmn.lib
/* #use ezioplc.lib  // for BL1200, BL1600, PK2100, PK2200 */
/* #use eziotgpl.lib // for BL1000 */
/* #use eziolqpl.lib // for BL1100 */
/* #use eziomgpl.lib // for Bl1400 & BL1500 */
/* #use eziobl17.lib // for BL1700 */
/* #use ezioplc2.lib // for BL1700 */

main() {
   auto int i;
   auto struct _eioAdcCalib c;
   auto int channel;
   auto float f1, f2, fout;
   auto long l;
   auto int margin;
   channel = 0;      // execute watch expression
        // to override
   margin = 0x40;    // execute watch expression
        // to override
   plcBusReset();
   if (plcXP89Init(channel / 8)) {
      printf(“DAC8 board not found\n”);
   } else {
      plcXP89Sw(channel / 8,1);
             // enable voltage reference
      plcXP89Out(channel,margin);
             // use meter to record level
      f1 = 10;       // use watch expr to override
      plcXP89Out(channel,0xfff-margin);
             // use meter to record level
      f2 = -10;      // use watch expr to override
      eioAdcMakeCoeff(&c,margin,0xfff-margin,f1,f2);
      if (plcXP89WrCalib(channel,&c)) {
        printf("Can’t write calibration constant\n");
      }
      memset(&c,0,sizeof(struct _eioAdcCalib));
      if (plcXP89RdCalib(channel,&c)) {
        printf("Can’t read calibration constant\n");
      }
      fout = 2.345;  // use watch expr to override
      l = eioAdcDigitize(fout, &c);
      plcXP89Out(channel,(unsigned)l);
                     // use meter to check voltage now
   }
}
```

Blank

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, PK2200 and PK2600 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.

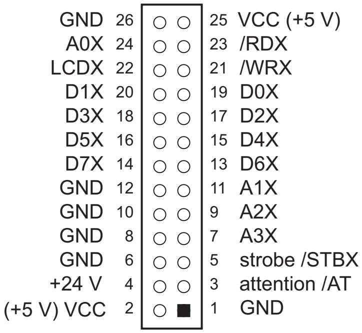


Figure A-1. PLCBus Pin Diagram

Actually, 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.

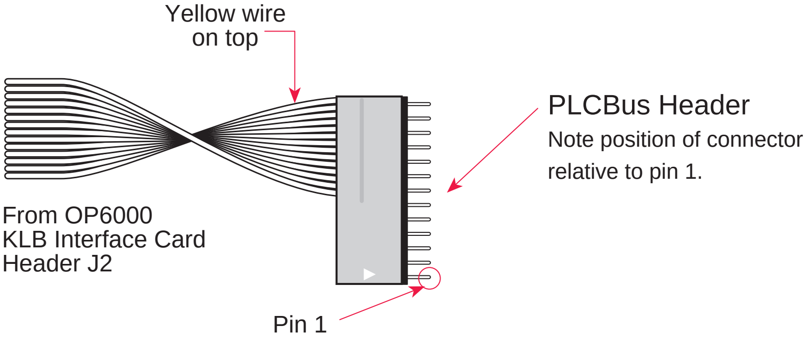


Figure A-2. OP6000 Connection to PLCBus Header

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	4 bits × 3	256	0000 xxxx xxxx
– – – – 0 0 0 1		256	0001 xxxx xxxx
– – – – 0 0 1 0		256	0010 xxxx xxxx
– – – – 0 0 1 1		256	0011 xxxx xxxx
– – – x 0 1 0 0	5 bits × 3	2,048	x0100 xxxxx xxxxx
– – – x 0 1 0 1		2,048	x0101 xxxxx xxxxx
– – – x 0 1 1 0		2,048	x0110 xxxxx xxxxx
– – – x 0 1 1 1		2,048	x0111 xxxxx xxxxx
– – x x 1 0 0 0	6 bits × 3	16,384	xx1000 xxxxxx xxxxxx
– – x x 1 0 0 1		16,384	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 xxxxxxxx
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

XP8600 Hardware Specifications

Table B-1 summarizes the specifications for the XP8600 expansion board.

Table B-1. XP8600 Specifications

Board Size	2.835" × 3.525" × 0.75" (72 mm × 90 mm × 19 mm)
Operating Temperature Range	-40°C to +70°C
Humidity	5% to 95%, noncondensing
Power (quiescent, no output)	24 V DC, 30 mA
Outputs	2 D/A channels • voltage output 0 V to 10 V each OR • current output 0 mA to 22 mA each

The XP8600's voltage regulator, the LM340T-15 at U9, provides 15 V up to 1 A. The voltage regulators at U8 and U2 provide V-, a regulated -5 V up to 50 mA. U13, and LT1021 provides a precision 10 V reference (REF-). The LT1021 is replaced with an LT1019 to provide a precision 2.5 V reference for REF- with 12 V controllers.

Figure B-1 shows the dimensions of the XP8600 expansion board.

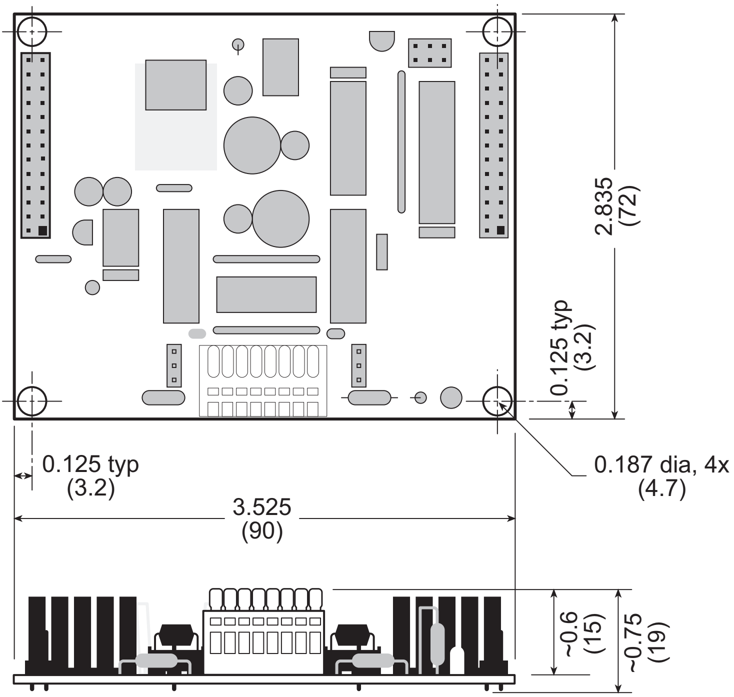


Figure B-1. XP8600 Board Dimensions

XP8900 Hardware Specifications

Table B-2 summarizes the specifications for the XP8900 Series expansion boards.

Table B-2. XP8900 Series Specifications

Board Size	2.835" × 4.00" × 0.75" (72 mm × 102 mm × 19 mm)
Operating Temperature Range	-40°C to +70°C
Humidity	5% to 95%, noncondensing
Power (quiescent, no output)	24 V DC, 30 mA
Outputs	8 D/A channels (4 channels for XP8910), bipolar voltage output 0 V to ±10 V, can source/sink up to 2 mA per channel on internal power (up to 7 mA per channel with user-supplied rails)

The XP8900 Series expansion boards derive +5 V digital power from the PLCBus supply via LM7805 at U6. When operating without user-supplied external voltage rails, the XP8900 Series expansion boards get their +12 V analog power from the PLCBus +24 V supply via LM7812 at U7. Charge pump NJU7662 at U17 inverts this for onboard -12 V analog power. Precision +5 V and + 2 V reference voltages are derived from the +12 V supply via the REF195 at U2 and the voltage divider formed from R33, R34, and R39. The n-channel FET FDV301N at Q1 is used to switch the 2 V reference to 0 V during a power-on reset.

Figure B-2 shows the dimensions of the XP8900 Series expansion boards.

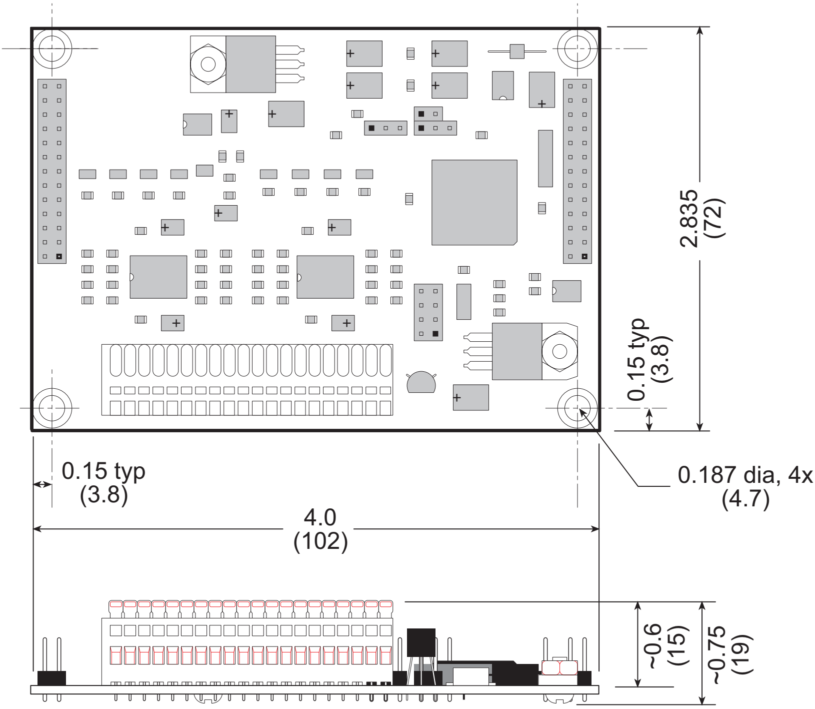


Figure B-2. XP8900 Board Dimensions

Blank

Appendix C: Connecting and Mounting Multiple Boards

Connecting Multiple Boards

Eight or more XP8600 expansion boards and eight XP8900 Series expansion boards may be connected ("daisy chained") at one time. Be sure that each board has a unique address to prevent communication problems between the controller and the expansion board.

Follow these steps when installing several expansion boards on a single PLCBus.

1. Make sure all expansion boards are orientated 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.
5. Add expansion boards side-by-side until the "daisy chain" is complete.

Figure C-1 shows an example of "daisy chained" connections.

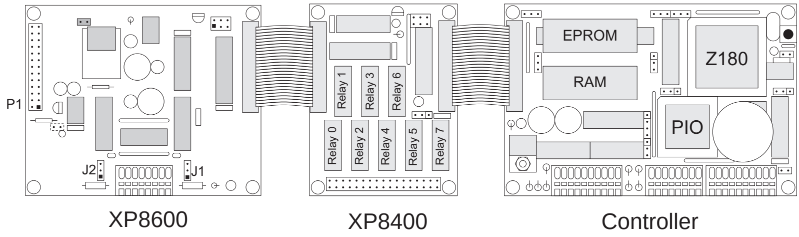


Figure C-1. Example of "Daisy Chain" Connections

> *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.

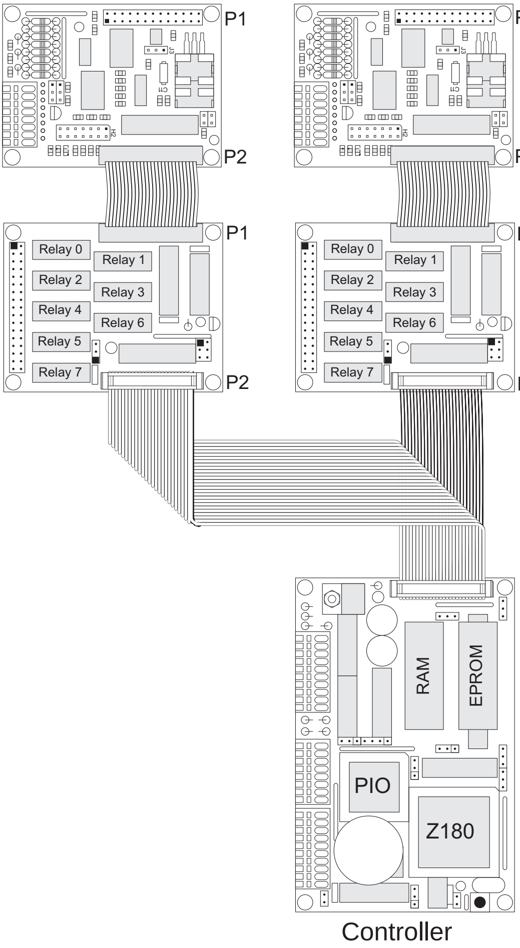


Figure C-2. Use of Y Cable to Connect Multiple Expansion Boards

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

Mounting Expansion Boards

The XP8600 and XP8900 Series 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.

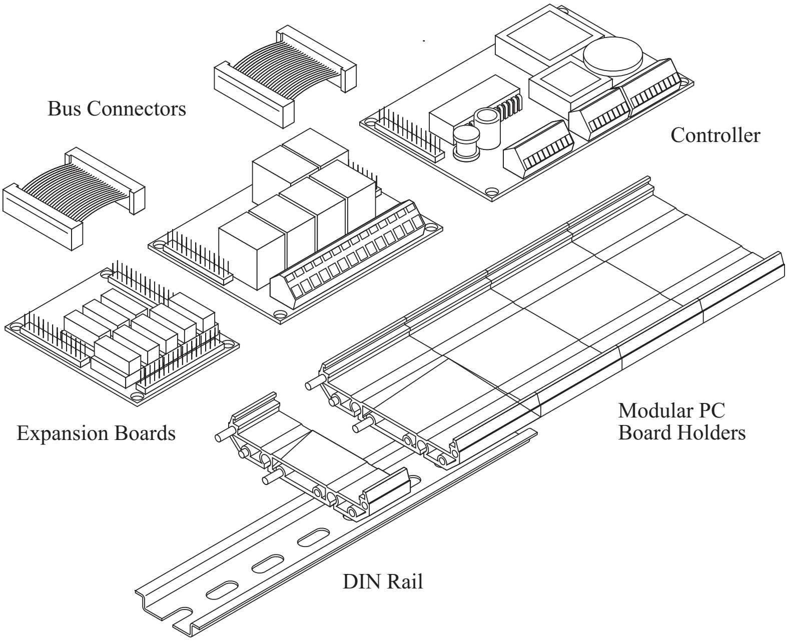


Figure C-3. Mounting Expansion Boards on DIN Rail

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

APPENDIX D:

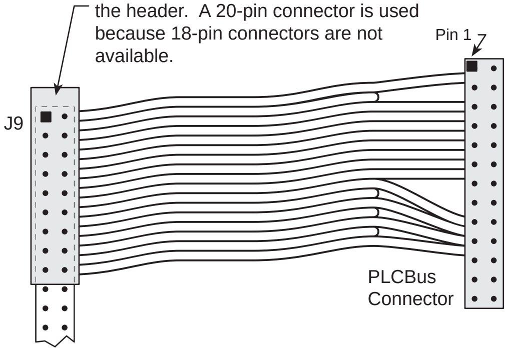
SIMULATED PLCBUS CONNECTIONS

BL1000

Connecting a Z-World expansion board to a BL1000 requires a special cable. Fasten the cable's 20-pin connector to header J9 as shown in Figure D-1. Pins 1 and 2 of the connector must hang over the end of the header. Fasten the cable's PLCBus connector to header P1 or P2 of the expansion board, observing the orientation of pin 1, as shown.

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

PIO Signal	PLCBus Signal
PB0 (J9:17)	D1X
PB1 (J9:15)	D0X
PB2 (J9:13)	D3X
PB3 (J9:11)	D2X
PB4 (J9:9)	A1X
PB5 (J9:7)	A2X
PB6 (J9:5)	A3X
PB7 (J9:3)	/STBX
+5 V (J9:1)	+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 D-1. Connecting BL1000 to Expansion Boards

Software for interfacing the BL1000's PIO port to a PLCBus port may be found in the Dynamic C `PBUS_TG.LIB` library.

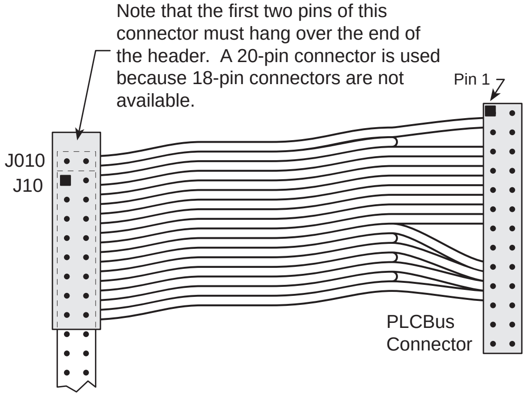
Use an external power supply with expansion boards connected to the BL1000. There is no provision in the special cable to supply +24 V from the controller to header P1 or P2 on the expansion boards.

BL1100

Connecting a Z-World expansion board to a BL1100 requires a special cable. Fasten the cable’s 20-pin connector to headers J010 and J10 as shown in Figure D-2. Pins 1 and 2 of the connector must hang over the end of the headers. Fasten the cable’s PLCBus connector to header P1 or P2 of the expansion board, observing the orientation of pin 1, as shown.

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 D-2. Connecting BL1100 to Expansion Boards

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

Use an external power supply with expansion boards connected to the BL1100. There is no provision in the special cable to supply +24 V from the controller to header P1 or P2 on the expansion boards.

BL1300

Expansion boards may be connected to header P5 on the BL1300 using the same special cable used to connect them to the BL1000 or to the BL1100, as shown in Figure D-2. The first two pins of the special cable hang over the end of header P5 as before. However, the wire leading to pin 1 on the BL1300’s header P5 must be cut, and may then be used to supply +5 V from an external source to the expansion board. Software from the Dynamic C **PBUS_LG.LIB** library may be used.

Use an external power supply with expansion boards connected to the BL1300. There is no provision in the special cable to supply +24 V from the controller to header P1 or P2 on the expansion boards.

BL1400 and BL1500

Expansion boards may be connected to header H3 on the BL1400 and BL1500. The signals, listed in Table D-1, are laid out differently from those on the other controllers, and so the special cable used for the BL1000, the BL1100, or the BL1300 controllers will not work. The user may either make a custom cable or use an adapter board available from Z-World. Software from the Dynamic C **`EZIOMGPL.LIB`** library may be used.

Table D-1. PIO Header to PLCBus Signal Map

BL1400/BL1500		Expansion Board	
Header H3 Pin	**PIO Port Signal**	**Header P1/P2 Pin**	**PLCBus Signal**
1	+5 V	2	+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

Use an external power supply with expansion boards connected to the BL1400 or BL1500. There is no provision to supply power from the controller to header P1 or P2 on the expansion boards. The adapter board has a jack and a screw terminal for the external +12 V/+24 V.

For more information on the adapter board, call your Z-World Sales Representative at (530) 757-3737.

INDEX

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. 019-0064
Revision C

Printed in U.S.A.