PCI-AC28 USER'S GUIDE

FORM 1195-030903—SEPTEMBER, 2003

OPTO 22

43044 Business Park Drive • Temecula • CA 92590-3614
Phone: 800-321-OPTO (6786) or 951-695-3000
Fax: 800-832-OPTO (6786) or 951-695-2712
www.opto22.com

Product Support Services
800-TEK-OPTO (835-6786) or 951-695-3080
Fax: 951-695-3017
E-mail: support@opto22.com

PCI-AC28 User's Guide
Form 1195-030903—SEPTEMBER, 2003

The information in this manual has been checked carefully and is believed to be accurate; however, Opto 22 assumes no responsibility for possible inaccuracies or omissions. Specifications are subject to change without notice.

Opto 22 warrants all of its products to be free from defects in material or workmanship for 30 months from the manufacturing date code. This warranty is limited to the original cost of the unit only and does not cover installation, labor, or any other contingent costs. Opto 22 I/O modules and solid-state relays with date codes of 1/96 or later are guaranteed for life. This lifetime warranty excludes reed relay, SNAP serial communication modules, SNAP PID modules, and modules that contain mechanical contacts or switches. Refer to Opto 22 form number 1042 for more details.

Table of Contents

CHAPTER 1

Installing the PCI-AC28

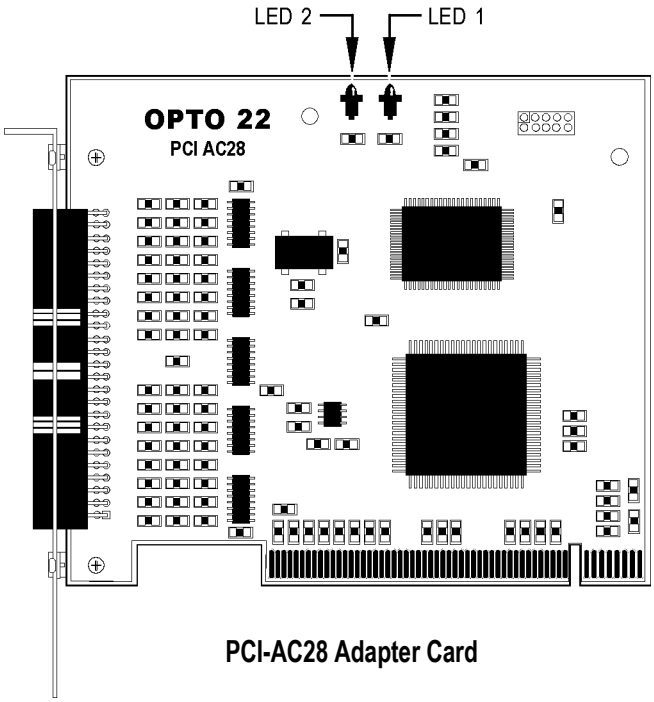
Introduction

The PCI-AC28 adapter card brings the industry-standard Pamux® to the world of Peripheral Component Interconnect (PCI), a local bus standard developed by Intel®. The PCI-AC28 adapter card is an ideal choice for customers who must replace an older ISA bus-based PC that currently uses an Opto 22 AC28 adapter card. The PCI-AC28 is compatible with computers that feature a 33 MHz PCI bus. Note that the PCI-AC28 supports 5-volt signalling environments only.

NOTE: The PCI-AC28 itself is no faster than its ISA cousin, the AC28, but it is the better choice if you have no available ISA slots or available I/O addresses.

With the PCI-AC28 adapter card, your PCI computer can communicate with Opto 22 classic B4, B5, and B6 brain boards and with SNAP-B4 and SNAP-B6 brains.

- Each Pamux bus can access up to 32 remote stations.
- Each Pamux bus supports up to 512 points.
- The card requires 5 VDC @ 600 mA and operates at temperatures of 0° C to 70° C.
- Configuration is jumperless.
- LEDs indicate bus operation and user application.
- Up to 32 PCI-AC28s are supported by the PCI Pamux driver.



PCI-AC28 Adapter Card

Free with the PCI-AC28 adapter card is the PCI Pamux Toolkit, included on the Opto 22 Adapter Card Toolkits CD that came with the card and also available from our Web site at www.opto22.com/products/softdevkits.asp. This developer toolkit includes sample applications, utility applications, and the PCI Pamux driver for 32-bit Windows.

PCI-AC28 with Linux

If you are using the PCI-AC28 adapter card with the Linux® operating system, you should know the bash shell, gcc, and C/C++. To obtain an open-source driver, send your request to linux@opto22.com.

What's in this Guide

This guide assumes that you are familiar with Pamux and the brains, racks, and input/output modules used with Pamux. For more information on Pamux, see Opto 22 form #726, *Pamux User's Guide*. If you are going to program the PCI-AC28 using the PCI Pamux Toolkit, this guide assumes that you are already familiar with programming in Microsoft® Visual Basic® or Visual C++®.

This guide includes four sections:

Chapter 1, "Installing the PCI-AC28," provides a Quick Start section to install the card, Product Support information, and specifications for the card.

Chapter 2, "Programming with the PCI Pamux Driver in Windows," includes a command reference as well as suggestions for customers moving from the AC28 to the newer PCI-AC28 adapter card.

Appendix A, "Writing Your Own Driver," provides additional technical information you need only if you are not using the PCI Pamux Driver, but writing your own driver for the PCI-AC28. For example, you would need to write your own driver if you are using an operating system other than Windows or Linux.

Appendix B, "Files for OEMs," lists files you need to include for installation if you are an original equipment manufacturer.

For Help

If you have problems installing or programming the PCI-AC28 adapter card and cannot find the help you need in this guide, contact Opto 22 Product Support.

Phone:	800-TEK-OPTO (835-6786) 951-695-3080 (Hours are Monday through Friday, 8 a.m. to 5 p.m. Pacific Time)
Fax:	951-695-3017
E-mail:	support@opto22.com
Opto 22 Web site:	www.opto22.com/support

When calling for technical support, be prepared to provide the following information about your system to the Product Support engineer:

- PC configuration (type of processor, speed, memory, operating system, and service packs)
- A complete description of your hardware and operating systems, including:
  - additional accessories installed (such as sound cards, NICs, etc.)
  - type of power supply
  - types of I/O units installed
  - third-party devices installed (for example, barcode readers)
- Software and version being used
- Specific error messages seen.

Quick Start

Follow the steps in this section to:

- Install hardware
- Install software
- Use the utilities provided to identify and test the card
- Change your application for the PCI-AC28.

Installing Hardware

The PCI-AC28 adapter card installs into any PCI expansion slot of a PCI-capable computer. The toolkit supports a maximum of 32 PCI-AC28 cards. You may add multiple PCI-AC28 adapter cards for convenience, but note that multiple cards do not increase Pamux throughput. The number of Pamux accesses per computer is constant.

Follow these steps to install the card:

1. Turn off the computer. Remove the power cord and the computer's cover.

   The power cord must be removed, or any sudden spike may cause the computer to automatically boot.

2. Before handling the PCI-AC28, discharge excess static electricity by touching the computer's metal chassis.

3. Install the card in one of the PCI expansion slots.

   CAUTION: *Do not scratch this card or other cards in the computer, as scratching may irreversibly damage the card or other devices.*

4. Verify that the PCI-AC28 card is properly seated in the motherboard PCI socket. Secure the card with the screw.

5. Reinstall the power cord. Leave the computer cover off temporarily so you can see the card's LEDs.

Because the card is self-configuring, it has no jumpers. Configuration is automatically done by PCI BIOS when the card is installed.

If You Are Running Windows 95, 98, 2000, XP, or ME

If you are running Windows (except NT), the computer discovers the new card and tries to find driver information in the system's .INF file. Follow these steps to provide the needed information:

1. Turn on the computer.
2. When the New Hardware Found dialog box appears, highlight Driver From Disk Provided by Hardware Manufacturer. Click OK.
3. Insert the Opto 22 Adapter Card Toolkits CD, which came with the card, in the CD-ROM drive. Browse to the CD, then to Drivers→your operating system→Pciac28.inf. Click OK.

   If you do not have the CD, you can download the AdapterCardToolkit or just the device driver from our Web site at www.opto22.com/products/softdevkits.asp. (If you want the CD, contact Product Support and ask for part number ADAPTERCARDTOOLKITCD. Contact information is on page 2.)
4. Continue with the next section, "Installing Software."

Installing Software

Necessary software for the PCI-AC28 adapter card is included on the Opto 22 Adapter Card Toolkits CD that came with the card. If you do not have the CD, you can download the software from our Web site at www.opto22.com/products/softdevkits.asp. This PCI Pamux Toolkit includes sample applications, utility applications, and the driver for 32-bit Windows.

1. Insert the CD in your CD-ROM drive. It should start automatically; if it does not, open the file setup.exe.
2. Run the installation, which automatically places the necessary files in the correct locations for the operating system you are using.

Using Utilities to Identify and Test the Card

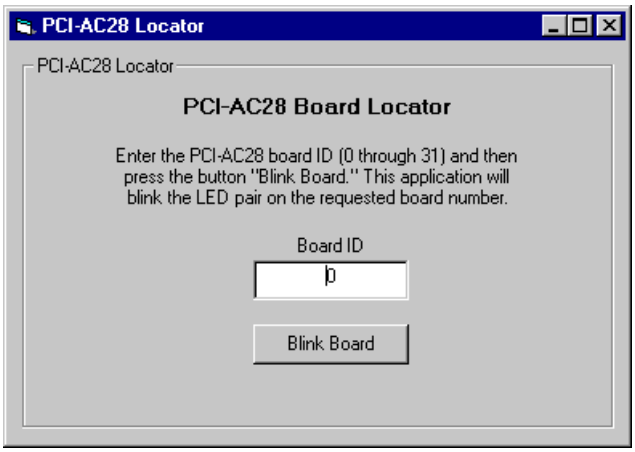
Identifying the Card

The driver assigns each PCI-AC28 adapter card an ID number between 0 and 31, starting with 0 for the first card. This ID number bears no relationship to the motherboard slot number into which you placed the card. Note that **any time you install or remove a PCI card in the computer, the ID number may change, and you should run this utility again.**

If you install only one PCI-AC28 card in the computer, its ID number is zero.

If you install more than one PCI-AC28 card in the computer, follow these steps to determine the ID number for each card.

1. From the Start menu, choose Programs→Opto 22→PCI Pamux Toolkit→PCI Locator.



2. Enter a number from 0–31 in the Board ID field. While watching the boards in the computer, click Blink Board.

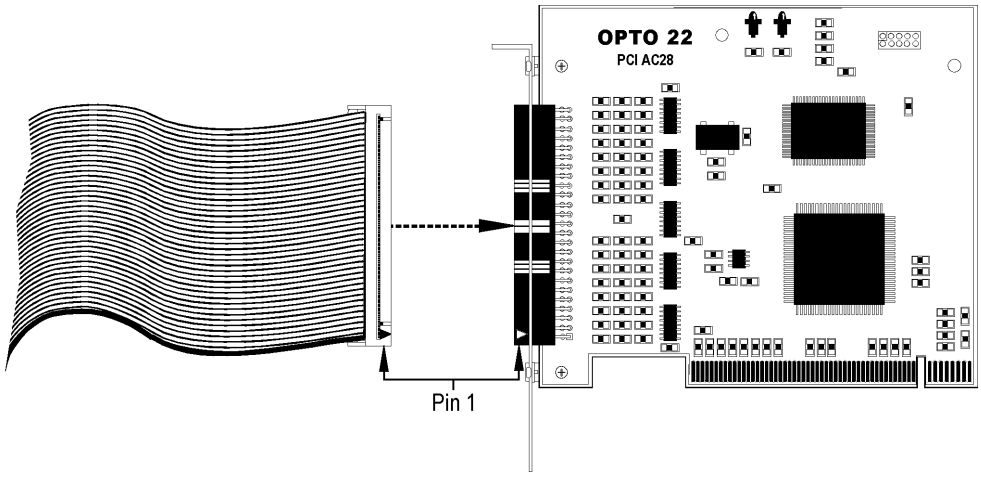
   The LEDs on the card for that ID number will blink.

3. Write the ID number on a label and place the label on the card for future reference. Replace the computer's cover.

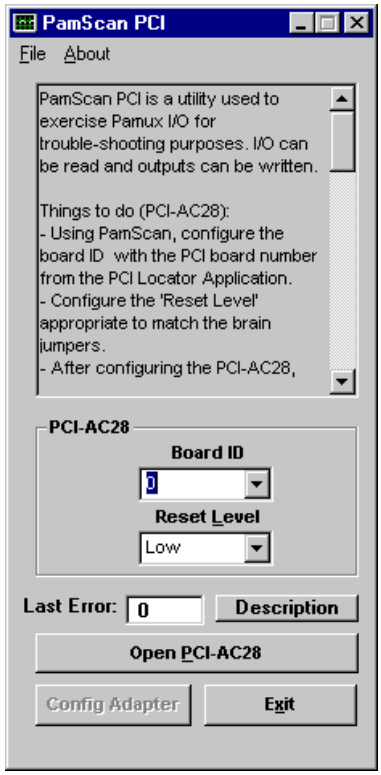
NOTE: Remember to rerun this utility to check the ID number if you remove or add any PCI card.

Testing the Card

The easiest way to test the card is to use the PamScan PCI utility, included in the PCI Pamux Toolkit. Using PamScan PCI, you can read and write directly to points on the I/O unit, without going through your application. Before testing, attach the I/O unit using a flat HH-series ribbon cable with a 50-pin header connector. Connections are shown in the diagram below. For additional information, see Opto 22 form #726, *Pamux User's Guide*.



PamScan PCI also serves as an application example, as source code is provided for both Visual Basic and Visual C. (See Opto 22\PCIPamux Toolkit\ Vb or \ Vc.)



1. From the Start menu, choose Programs✎Opto 22✎PCI Pamux Toolkit✎PamScan PCI.

   The main window appears, as shown at right.

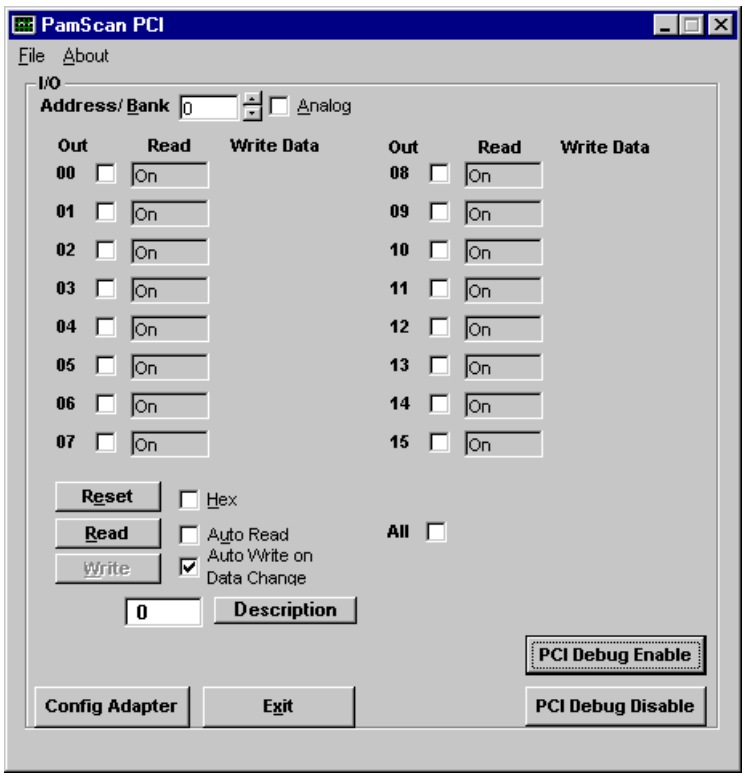
2. Enter the board's ID number in the Board ID field.

3. Set the Reset Level to match the brain configuration on the Pamux bus.

   We recommend that the Reset Level be set to Low. If the computer is turned off while racks and brains are still running, some of the outputs may change state. If the Reset Level is set to Low, however, in this situation the brains will reset and the system will go to a safe state.

   NOTE: The reset level set here and the reset level set on all the Pamux brains in your system ***must match****.*

4. Click Open PCI-AC28.



5. Test the card by doing any of the following:
   - Update the brain's address.
   - Click the Read button and watch as points are updated.
   - To write to a point, click to put a check mark in the Out box. Enter a value in the field that appears. Click the Write button to write to that point.
   - Check Auto Read or Auto Write. (Watch values change once per second as they are automatically read. Change values and watch them automatically written.)

6. To verify that communication occurs through the card, click PCI Debug Enable to make the card's LEDs blink when you read or write.

   LED 1 blinks when I/O points are read. LED 2 blinks when you write to points. For LED locations, see the diagram on page 1.

   NOTE: So that the blinks are long enough to be seen, a one millisecond delay is built into the debugging mode. If you are using a time-critical application, the delay may cause adverse slowing. To stop the LEDs from blinking, click PCI Debug Disable.

7. When you have finished testing the card, click Exit.

Changing Your Application for the PCI-AC28

If you have been using an AC28 adapter card for the ISA bus, you will need to make some changes to your application because of the new PCI adapter card. See Chapter 2 for specific information.

CHAPTER 2

Programming with the PCI Pamux Driver in Windows

Overview

To simplify communication to the Pamux bus, you can use Opto 22's PCI Pamux driver. This chapter explains how to use the driver.

What Is the PCI Pamux Driver?

The PCI Pamux analog/digital driver provides an interface between Pamux stations and application programs written in Microsoft Visual C++ and Visual Basic. The driver saves you time and effort that would otherwise be spent learning the intricacies of the Pamux bus structure.

The Pamux driver is 32-bit Microsoft Windows® software, a dynamically linked library called OptoPM32.dll. The driver may be used with Microsoft Visual Basic or Visual C++.

The driver performs the following functions:

- Converts the data returned by Pamux to a form that is easily manipulated in a high-level language.
- Carries out all necessary handshaking with the Pamux bus.
- Transparently handles input masking on write operations for digital stations.
- Performs error checking and returns diagnostic codes.

To use the driver in your application program, you need to know the following:

- How to call a subroutine or function from the language you chose for your application
- How to tell the driver what Pamux command to send by assigning values to parameters
- How to interpret the data passed back by the driver.

The PCI Pamux driver may be used only with the PCI-AC28 adapter card.

Installation

The PCI Pamux driver is included on the Opto 22 Adapter Card Toolkits CD, which came with the card. If you do not have the CD, you can order it through Product Support, or you can download the driver free from our Web site, www.opto22.com/products/softdevkits.asp, as part of the PCI Pamux Toolkit. The driver is automatically installed with the Toolkit.

Pamux Functions

Required Function Calls

For many applications, only four Pamux functions are required:

1. Open a PCI-AC28 to configure the PCI-AC28 and get a handle.
2. Configure outputs.
3. Read and write to I/O.
4. Close the PCI-AC28 when the application is about to end.

Naming Conventions

Function names in the Pamux library start with "Pamux." Example: "PamuxDigPointRead."

Function names follow the object-operation format, with the object first and the operation second. Example: "PamuxDigPointRead," where "PamuxDigPoint" (the object) is first and "Read" (the operation) follows.

Utility functions, provided primarily for Visual Basic, start with "PamuxUtil." Example: "PamuxUtilBitEqual."

Specific PCI-AC28 functions start with "PamuxPCI."

Banks and Points

Some I/O points can be addressed in multiple ways. A 16-channel I/O board has two banks. Point 0, the first point, is accessed using a bank number of 0 and a point number of 0. Point 8 can be accessed in two ways:

- A bank number of 0 and a point number of 8, or
- A bank number of 1 and a point number of 0.

Common Function Parameters and Return Values

`int hPCI-AC28` Handle to a PCI-AC28 card. Handles are acquired using PamuxPCICardOpen().

`int Bank` A bank number (0 to 63).

`int Point` A point or channel number on a rack starting with zero.

`OutputMask` A "1" bit represents an output. Used to configure outputs.

Return Values

All functions in the OptoPM32.dll return an error value. A non-zero value indicates an error has occurred. For proper application operation, make sure your program checks error codes. See the list of error codes on page 19.

Developing the I/O Application

Use the following basic steps in your application (in Visual Basic or Visual C++):

1. Open a handle to the board using the function PamuxPCICardOpen.
2. Inspect the error code of the device open function.
3. Configure the direction of the points.
4. Start the application loop that continuously reads or writes points. At the same time, continue to inspect the error codes from the OptoPM32.DLL.
5. When the application loop is complete, close the handle to the board using PamuxPCICardClose.

Special Directions for Visual Basic Programmers

Include the OptoPM32.bas file as a module in your project. This file includes subroutine declarations, function declarations, and access paths to the OptoPM32.dll.

These files may be found in the toolkit under Opto22\PCI Pamux Toolkit\Vb\VB dll header.

Special Directions for Visual C++ Programmers

Include the header OptoPM32.h in your source code modules that reference the OptoPM32.dll functions. Also include the DLL link library OptoPM32.lib in your project so the DLL references are resolved.

These files may be found in the toolkit under Opto22\PCI Pamux Toolkit\Vc\VC Project Includes.

Function Command Reference

The functions listed in this section include parameters and descriptions.

PCI-AC28 Operations

Function Type	Function	Parameter Type	Parameters	Description

long	`PamuxPCICardOpen`	long
		long
		long
			`* phPCI-AC28`
			`BoardID`
			`ResetLevel`
				Opens access to a PCI-AC28 card.
				`phPCI-AC28` gets a handle.
				Three parameters must be specified: the I/O Port addresses for the PCI-AC28 base port, its reset port, and the reset level. These three parameters must correspond to the PCI-AC28 card.
				`ResetLevel`: 1 = high reset; 0 = low reset.
				This function also performs a PCI-AC28 and Pamux bus reset, and turns off LEDs 1 and 2.

long	`PamuxPCICardClose`	long	`hPCI-AC28`	Releases the handle to the PCI-AC28 and turns on LEDs 1 and 2.
long	`PamuxPCIEnableDebug`	n/a	n/a	Enables debug mode, so the PCI-AC28 will display reads and writes on the internal LED indicators. A read blinks LED 1; a write blinks LED 2. See page 1 for LED locations. When the DLL is in debug mode, application requests to PamuxPCILightShow are ignored. **Warning:** Due to the extremely short Pamux bus cycles, a 1millisecond delay is added to each read and write cycle when debug is enabled. This delay allows the LEDs to be easily seen. However, the delay may cause adverse slowing in a time-critical application. **Note:** This function is only intended for debugging purposes. However, the side effect of this delay for time-independent applications may not be noticeable, and the developer may elect to leave the DLL in debug mode as a diagnostic tool for Pamux bus accesses. Debug mode is disabled by default.
long	`PamuxPCIDisableDebug`	n/a	n/a	Disables the PCI-AC28 adapter debug indication mode.
long	`PamuxPCILightShow`	long long	`hPCI-AC28` `Data`	Provides application layer control of the LEDs on the PCI-AC28. Useful for determining which adapter is being addressed.
long	`PamuxCardReset`	long	`hPCI-AC28`	Resets the PCI-AC28 card and resets the Pamux bus, turning off all outputs. LEDs 1 and 2 flash.

* Note for Visual Basic users: * indicates a “by reference” argument.

Digital Bank Operations

The term "bank" refers to groups of eight digital I/O points. A 32-channel Pamux board with a B4 brain board has four banks. It is faster to read a bank all at once than to read each point individually.

Note that channel 0 corresponds to the least significant bit. For example, if you read a bank with channels 0, 3, and 4 on and all other channels off, the returned value would be 19 hex (11001 binary, or 25 decimal).

Function Type	Function	Parameter Type	Parameter	Description
long	PamuxDigBankConfig	long long long long	hPCI-AC28 Bank OutputMask Byte Qty	Configures a bank of digital I/O points as either inputs or outputs. A "1" in the mask indicates an output. Use PamuxDigBankConfig for configuring between 8 and 32 points, PamuxDigBank16Config for 16 points, or PamuxDigBank32Config for 32 points. The Byte Qty parameter in PamuxDigBankConfig indicates how many banks to configure (Byte Qty = 4 for 32 points).
long	PamuxDigBank16Config	long long long	hPCI-AC28 Bank OutputMask	
long	PamuxDigBank32Config	long long long	hPCI-AC28 Bank OutputMask	
long	PamuxDigBankRead	long long long	int hPCI-AC28 Bank * pData	Reads inputs and outputs and places the result in pData. Use PamuxDigBankRead for reading eight points, PamuxDigBank16Read for 16 points, or PamuxDigBank32Read for 32 points.
long	PamuxDigBank16Read	long long long	int hPCI-AC28 Bank * pData	
long	PamuxDigBank32Read	long long long	int hPCI-AC28 Bank * pData	
long	PamuxDigBankWrite	long long long	hPCI-AC28 Bank Data	Writes outputs using the value in Data. Inputs are not affected if written to. Use PamuxDigBankWrite for writing to eight points, PamuxDigBank16Write for 16 points, or PamuxDigBank32Write for 32 points.
long	PamuxDigBank16Write	long long long	hPCI-AC28 Bank Data	
long	PamuxDigBank32Write	long long long	hPCI-AC28 Bank Data	

* Note for Visual Basic users: * indicates a "by reference" argument.

Digital Point Operations

Function Type	Function	Parameter Type	Parameter	Description
long	PamuxDigPointConfig	long long long long	hPCI-AC28 Bank Position bOutput	Configures a point as either an input or output. A non-zero value in bOutput configures the point as an output.
long	PamuxDigPointRead	long long long long	hPCI-AC28 Bank Point * pData	Reads the value of a point and puts the value in pData. The value is either 1 for on or 0 for off.
long	PamuxDigPointWrite	long long long long	hPCI-AC28 Bank Point Data	Writes to a point using the value in Data. A non-zero value for Data turns the point on.

* Note for Visual Basic users: * indicates a “by reference” argument.

Digital “Fast” Operations

For high-speed applications, these functions can be used to bypass some error-checking and port calculations. The configure functions should be used to configure outputs.

Function Type	Function	Parameter Type	Parameter	Description
long	PamuxDigIoPortGet	long long long long	hAc28 * pBank * pPoint * pIoPort	Provides the Pamux metrics needed: the PCI-AC28 handle, the bank number, and the point number.
long	PamuxDirectRead	long long long	hAc28 Bank *Data	Reads one byte (eight bits) from the specified I/O port.
void	PamuxDirectWrite	long long long	hAc28 Bank Data	Writes one byte (eight bits) to the specified port.

* Note for Visual Basic users: * indicates a “by reference” argument.

Analog Bank Operations

Function Type	Function	Parameter Type	Parameter	Description
long	PamuxAnaBank16Config	long long long	hPCI-AC28 Bank OutputMask	Configures a bank of analog I/O points as either inputs or outputs. A 1 in the mask indicates an output.
long	PamuxAnaBank16Read	long long long	hPCI-AC28 Bank DataArray16[]	Reads a bank of 16 analog points and places the values in the DataArray (channel 0 in element 0 and channel 15 in element 15).
long	PamuxAnaBank16Write	long long long	hPCI-AC28 Bank DataArray16[]	Writes values to a bank of analog points (channel 0 value in element 0).

* Note for Visual Basic users: * indicates a “by reference” argument.

Analog Point Operations

Function Type	Function	Parameter Type	Parameter	Description
long	PamuxAnaPointConfig	long long long long	hPCI-AC28 Bank Point bOutput	Configures an analog point as either an input or output. A non-zero value to bOutput configures the point as an output.
long	PamuxAnaPointRead	long long long long	hPCI-AC28 Bank Point * pData	Reads the value of an analog point.
long	PamuxAnaPointWrite	long long long long	hPCI-AC28 Bank Point Data	Writes the value in Data to an analog point.

* Note for Visual Basic users: * indicates a “by reference” argument.

Analog Watchdog Operations

Function Type	Function	Parameter Type	Parameter	Description
long	PamuxAnaWatchdogSet	long long long long	hPCI-AC28 Bank Time100 Data 16[]	Sets up the watchdog timer for an analog point. Time100 units are hundredths of a second.
long	PamuxAnaWatchdogTime	long long long	hPCI-AC28 Bank Time100	Sets the value of the watchdog timer in units of hundredths of a second. Setting Time100 to 0 disables the watchdog. Setting the time to 0 can also be used to reset the watchdog error flag if it has tripped. This and any other analog function can be used to "tickle" the watchdog to prevent it from tripping.

* Note for Visual Basic users: * indicates a "by reference" argument.

Analog Status Operations

Function Type	Function	Parameter Type	Parameter	Description
long	PamuxAnaStatusGetAsError	long long	hPCI-AC28 Bank	Gets an analog board's status and returns an equivalent error. Analog read functions also return the same result after performing their read. An analog configure function can be used to clear the "power-up" error. The "old data" error is cleared by the B6 processor as it updates inputs. This function could be used to wait for fresh input data.

* Note for Visual Basic users: * indicates a "by reference" argument.

Pamux Utility Operations

The following utility functions are provided primarily for Visual Basic applications. These are not Pamux-specific functions.

Function Type	Function	Parameter Type	Parameter	Description
Bit Operations				
void	PamuxUtilBitSetTo	long long long	* pData BitNumber0 bBitValue	These bit operations are useful in Visual Basic applications to access individual bits within an integer. The BitSetTo function either sets or clears the specified bit based on the value of bBitValue. Any non-zero value for Data turns on the bit. The BitSet and BitClr functions set and clear the specified bit. Bit numbers start at zero for the least significant bit (LSB). The zero at the end of the parameter name "BitNumber0" serves as a reminder of this fact for anyone looking through the function definitions in either the .BAS file or the .H header files. PamuxUtilBitTest returns true if bit number BitNumber0 in Data is set.
void	PamuxUtilBitSet	long long	* pData BitNumber0	
void	PamuxUtilBitClr	long long	* pData BitNumber0	
long	PamuxUtilBitTest	long long	Data BitNumber0	

* Note for Visual Basic users: * indicates a "by reference" argument.

Function Type	Function	Parameter Type	Parameter	Description
Pack/Unpack Utility Operations				
void	PamuxUtilBitPackArray2I	long long long	* DestInt SourceArray[] Qty	Converts an array of boolean (0 or not 0) values to a bit packed integer.
void	PamuxUtilBitUnPackI2Array	long long long	DestArray[] SourceInt Qty	Converts a bit packed integer to an array of boolean (1 or 0) values. Qty indicates the number of bits to be packed or unpacked.

* Note for Visual Basic users: * indicates a "by reference" argument.

Function Type	Function	Parameter Type	Parameter	Description
Scaling Operations				
long	PamuxUtilScaleI2I	long	X1	These interpolation functions are useful for converting between engineering units and raw analog counts. Pamux analog input and output values range between 0 and FFF hex (4,095 decimal). These values typically correspond to engineering units, such as pH and psi. For example, to convert raw counts (from 0 to FFF hex) to a percentage, use: float fPercent=PamuxUtilScaleI2F (0,0xFFF,0.0,100.0,RawCount);
		long	X2	
		long	Y1	
		long	Y2	
		long	Xin	
float	PamuxUtilScaleI2F	long	X1	
		long	X2	
		float	Y1	
		float	Y2	
		long	Xin	
float	PamuxUtilScaleF2F	float	X1	
		float	X2	
		float	Y1	
		float	Y2	
		float	Xin	
long	PamuxUtilScaleF2I	float	X1	
		float	X2	
		long	Y1	
		long	Y2	
		float	Xin	

* Note for Visual Basic users: * indicates a “by reference” argument.

Error Codes

In general, most functions return an integer error number. Zero indicates no error. You may see the following error codes when working with the PCI Pamux driver.

Code Decimal	Code Hex	Description	Remedy
0	0x0000	No Error Occurred	Not an error.
8192	0x2000	Invalid Handle	The handle that was passed to the OptoPM32.DLL is invalid. The handle may represent a closed handle or the value of the handle is corrupted. Inspect when the handle flaw is first detected and ensure that the handle was allocated with a successful open. Also trace a sudden change in the value of the handle. The handle should remain static between a PamuxPCICardOpen and a PamuxPCICardClose.
8193	0x2001	Bad bank number used	The bank number specified is either less than zero or greater than 63.
8194	0x2002	Bad I/O port used	A historic WinRT error that doesn't apply to the OptoPM32.DLL.
8195	0x2003	Out of handles to allocate	The PCI-AC28 you attempted to open is already open.
8196	0x2004	The Open command has conflicting parameters	Not currently used in the OptoPM32.DLL.
8197	0x2005	Point number is bad	The point argument is lower than zero or greater than 7.
8198	0x2006	Could not acquire access to B6 DPRAM	The attempt to gain access to the B6 or SNAP-B6 analog memory failed. This is not an error; it may mean that the brain was involved in reading and writing to this memory array.
8199	0x2007	Power-up clear /B6 needs configuring	The B6 or SNAP B6 was recently powered up due to a manual power enable or from a power dip event, causing the brain to reset. This brain may require special reconfiguration.
8200	0x2008	B6 didn't update – PC polling too quick	The rate of the PC's polling is very fast. This is not an error. It only indicates that the application should be modified to decrease the analog scan intervals. This code may be seen on faster CPU computers.
8201	0x2009	Watchdog timeout has occurred	The brain reports a watchdog timeout. This is caused when a communication cycle to the bus exceeds the watchdog timeout time.
8202	0x200A	Wrong OptoPMux DLL (trying to open ISA card)	This error indicates that a call was made to a function that is not currently supported in this version of the OptoPM32.DLL.
8203	0x200B	Board ID doesn't exist in system	The PCI board number matching the Opto 22 vendor ID (0x148A) and device ID (0xAC28) could not be found. Use the PCI Locator to verify the existence of this board. (See page 4.) Also remember that board IDs start from zero and end at "n-1" (where n is the number of boards).

Code Decimal	Code Hex	Description	Remedy
8204	0x200C	A closed handle tried to be used	The handle that was sent to this function is closed. This may represent corruption of the handle, failure to successfully open the handle, or a PamuxPCICardClose may have been previously executed on this handle.
8205	0x200D	A handle number out of range tried to be used	The handle submitted is invalid, as it is beyond the number of handles the OptoPM32.DLL supports. Inspect the handle for corruption. Also, validate that the handle number does not change when PamuxPCICardOpen opens a valid handle. It is also possible that an incorrect argument is being passed to the function.
8206	0x200E	Specified device is already open	When inspecting the opening of a PCI-AC28, the handle's data structure is marked as open. The OptoPM32.DLL does not support multiple access handles to individual PCI-AC28s.
8207	0x2010	Registry entry does not exist.	A historic WinRT error that doesn't apply to the OptoPM32.DLL.
8384	0x20C0	Specified PCI board ID was not found.	The board ID is beyond the range of valid board ID numbers. The range is always from zero to one less than the number of adapters installed in the computer.
8385	0x20C1	The device layer was not found.	The customer application is copied onto a system that does not have the driver layer installed.
8386	0x20C2	The device layer file version is too old for the toolkit.	May happen if a toolkit with an older WinDriver version is installed onto the system. Update all toolkit installations with the latest drivers from the latest toolkits.
8387	0x20C3	OptoPM32 does not support this function.	An unsupported PCI-AC28 function is called from the application. The PCI-AC28 does not support the identify type and does not support a reset level function.
8388	0x20C4	The device layer could not create a handle.	Another open is blocking the requested device. Call Product Support if this error is detected.
8389	0x20C5	The requested PCI board doesn't exist, or the device layer is improperly configured.	With a DOS prompt box, try the command "wdreg install". If this command fails, reinstall the toolkit. Otherwise, call Product Support.
8390	0x20C6	The PCI board failed to register with the PCI BIOS.	Try on a system with a newer PCI-BIOS or see if the manufacturer has a PCI-BIOS or BIOS ugrade for the system.

Converting Applications from the AC28 to the PCI-AC28

Applications that Used the OptoPMUX.DLL for the AC5

The Windows 32/PCI version of this library does not support the AC5 adapter card. Special historic versions of the OptoPMux.dll supported the AC5. This support is not possible due to independent hardware identification issues and the PCI bus. For more information, please contact Opto 22 Product Support.

Converting Applications that Use Inp( ) and Outp( )

This PCI driver library provides a consistent application code model for Windows NT, Windows 95, and Windows 98 operating systems. Inp() and Outp() function calls at the user level are unsupported because of the Windows NT hardware abstraction layer.

The PCI-AC28 interfaces to the existing Pamux interface by mimicking the 50-wire IDC connector and the Pamux timing interface. The hardware model to the PC is radically different compared to the AC28. Also, the ISA interface was static, whereas the PCI model is dynamic and automatically reconfigurable. The AC28 relied on jumper settings for configuration, but the PCI-AC28 has no jumpers.

The primary advantage of converting your application to this library is the encapsulation of the Pamux functions. This library provides high-level functionality, as opposed to setting and clearing bits. Additionally, porting the application to a 32-bit operating system takes advantages of 32-bit optimized processors, other current operating system features, and some code for Windows 95/98 and Windows NT.

For the Windows 95/98 or Windows NT Historic OptoPMux User

This new library eliminates the need for a WinRT/OptoPort utility. Each PCI card is referenced by PCI slot number. The lowest numbered board, zero, is the PCI-AC28 installed in the lowest PCI slot number and bus number. Note that internal PCI slot numbers have no correlation with any "SLOT" number that may appear on the motherboard. See "Identifying the Card" on page 4.

You may want to use a cyclic reset command to flash the LEDs on and off, in order to identify the board in a final application. Or you can open a handle to the board and use Pamux PCI Light Show to toggle the LEDs.

The following tables list obsolete, changed, and new functions.

Functions No Longer in Use

Function	Comments
PamuxDigBankWriteFast	Always returns an incorrect DLL version error number O22_PM_WRONG_PMUX_DLL. Use the new function PamuxDirectWrite.
PamuxDigBankReadFast	Always returns an incorrect DLL version error number O22_PM_WRONG_PMUX_DLL. Use the new function PamuxDirectRead.
PamuxCardOpen	Always returns an incorrect DLL version error number O22_PM_WRONG_PMUX_DLL. Use the new function PamuxPCICardOpen.
PamuxCardClose	Always returns an incorrect DLL version error number O22_PM_WRONG_PMUX_DLL. Use the new function PamuxPCICardClose.
PamuxReadType	Always returns an error. This function is not supported by this DLL.

Changed Functions

Function	Comments
PamuxDigIoPortGet	Historic function used an absolute I/O port address to generate references. Since the PCI card hides the base address of the hardware, this function is modified to only compute the bank number and the point number.
PamuxCardSetReset	Always returns an incorrect DLL version error number O22_PM_WRONG_PMUX_DLL. This ISA function sets the reset level of the Pamux bus. The PCI-AC28 version also internally uses this function to reset the Pamux state machine.

New Functions

Function	Comments
PamuxPCICardOpen	Replaces PamuxCardOpen. The BoardID is an argument with a range [0:31]. The function also turns off both LEDs.
PamuxPCICardClose	Replaces PamuxCardClose. This function also turns on both LEDs.
PamuxDirectRead	Permits a direct read from the Pamux register map. Replaces PamuxDigBankReadFast. This function conforms to the PCI-AC28 read requirements.
PamuxDirectWrite	Permits a direct write to the Pamux register map. Replaces PamuxDigBankWriteFast.
PamuxPCILightShow	Provides application layer control of the light-emitting diodes on the PCI-AC28. Useful for determining which adapter is being addressed or the status of the application.

Function	Comments
`PamuxPCIEnableDebug`	Enables debug mode, so the PCI-AC28 will display reads and writes on the internal LED indicators. A read blinks LED 1; a write blinks LED 2. See page 1 for LED locations. When the DLL is in debug mode, application requests to PamuxPCILightShow are ignored. **Warning:** Due to the extremely short Pamux bus cycles, a 1millisecond delay is added to each read and write cycle when debug is enabled. This delay allows the LEDs to be easily seen. However, the delay may cause adverse slowing in a time-critical application. **Note:** This function is only intended for debugging purposes. However, the side effect of this delay for time-independent applications may not be noticeable, and the developer may elect to leave the DLL in debug mode as a diagnostic tool for Pamux bus accesses. Debug mode is disabled by default.
`PamuxPCIDisableDebug`	Disables the PCI-AC28 adapter debug indication mode.

Special Precautions for the Software Developer

No exclusive access—The PCI Pamux driver allows up to 32 PCI-AC28 cards to be used. Only a single handle to a card is permitted. If you use a multiple threaded application, implement a mutex on the handle to avoid thread collision. If multiple applications are required to access the hardware, another application is required to synchronize the access. Multiple applications cannot access the PCI-AC28s simultaneously.

When reading the PCI-AC28 at the hardware level—The first read begins the asynchronous Pamux bus cycle, and the second read retrieves data from the Pamux bus. If the second read is delayed, you receive the data initiated from the first read. However, if the application is abnormally terminated, a reset re-initializes the PCI-AC28 and the Pamux bus and re-synchronizes the Pamux read state machine.

Functional changes to the Pamux reset—Historically, the Pamux signal only resets the functions on the Pamux bus. The PCI-AC28 resets functions on the Pamux bus and also uses the reset signal to reset an onboard controller or state machine. Use the reset after an abrupt termination to reset the onboard controller's last read status.

APPENDIX A

Writing Your Own Driver

Introduction

Some customers with non-Windows operating systems may wish to use the PCI-AC28 adapter card. Since the PCI Pamux driver is for Windows only, these customers will need to write their own drivers. This appendix provides technical information you will need to write your own driver.

NOTE: If you are using Linux, request an open-source driver from linux@opto22.com.

As an additional reference, see *PCI Hardware & Software, 4th Edition* by Edward Solari and George Willse (ISBN #: 092939259-0).

Hardware Architecture

Since the PCI-AC28 uses a standard PCI core interface for handling the PCI handshake, other machine architectures that conform to the PCI specifications should be able to successfully communicate with the PCI-AC28.

The PCI-AC28 is auto-configuring due to the PCI BIOS feature on most computers. The historic I/O port and the reset port jumpers of the AC28 are now configured automatically. On powerup, the computer's PCI BIOS reads the configuration registers from the card and assigns the card's base addresses. The PCI-AC28 uses only I/O memory space.

The base address provides the Data Register access on the Pamux bus, and base address + 1 provides the Control Register access.

Notes for the Developer

The PCI Pamux driver keeps track of inputs and outputs to prevent a 1 from being written to a point that is configured as a digital input. Writing a 1 to a digital input causes a 1 to be read from the input regardless of the state of the input module. When you are writing your own driver, you need to **make sure that a 1 is not written to a digital input**.

Functional changes to the Pamux reset—Historically, the Pamux signal only resets the functions on the Pamux bus. The PCI-AC28 resets functions on the Pamux bus and also uses the

reset signal to reset an onboard controller or state machine. Use the reset after an abrupt termination to reset the onboard controller's last read status.

The hardware is not thread safe—Due to the back-to-back reading and resetting configuration, multiple thread/application accessing will corrupt the operation of the card.

Additionally, because specific analog and digital Pamux functions require multiple reads and writes, the card will not properly operate if multiple Pamux reads and writes occur from multiple threads and/or applications. **Make sure your control or monitoring system cannot execute multiple instances.**

When accessing a Pamux bank with the PCI-AC28, a back-to-back read of the same bank address is required. The first read initiates the Pamux bus cycle and the second read captures the Pamux data. No other operations except a PCI-AC28 hardware reset should occur during this operation.

Pamux Base Address Expansion

The read/write I/O space address is the PCI-AC28 base address register 1 (BAR1) added with four times the Pamux register number.

PCI_AC28_IO_Address = BAR1 + (Pamux_Register * 4)

Pamux Register (Brain Address)	PCI-AC28 Offset
0x0	BAR1 + 0x00
0x2	BAR1 + 0x08
0x4	BAR1 + 0x10
0x6	BAR1 + 0x18
0x8	BAR1 + 0x20
0xA	BAR1 + 0x28
0xC	BAR1 + 0x30
0xE	BAR1 + 0x38
0x10	BAR1 + 0x40
0x12	BAR1 + 0x48
0x14	BAR1 + 0x50
0x16	BAR1 + 0x58
0x18	BAR1 + 0x60
0x1A	BAR1 + 0x68
0x1C	BAR1 + 0x70
0x1E	BAR1 + 0x78

Pamux Register (Brain Address)	PCI-AC28 Offset
0x20	BAR1 + 0x80
0x22	BAR1 + 0x88
0x24	BAR1 + 0x90
0x26	BAR1 + 0x98
0x28	BAR1 + 0xA0
0x2A	BAR1 + 0xA8
0x2C	BAR1 + 0xB0
0x2E	BAR1 + 0xB8
0x30	BAR1 + 0xC0
0x32	BAR1 + 0xC8
0x34	BAR1 + 0xD0
0x36	BAR1 + 0xD8
0x38	BAR1 + 0xE0
0x3A	BAR1 + 0xE8
0x3C	BAR1 + 0xF0
0x3E	BAR1 + 0xF8

The PCI BIOS assigns an interrupt when it configures the PCI-AC28. This interrupt is not used by the Pamux board. This assignment is a configuration issue within PCI.

PCI-AC28 Function Registers

The following table shows the PCI register map.

Function	Direction	PCI-BAR	Value	Offset
Set Reset Assert High	Write	BAR0	0x0F	0x0F
Set Reset Assert Low	Write	BAR0	0xFF	0x0F
Assert Reset	Write	BAR0	0x01	0x3F
Negate Reset	Write	BAR0	0x00	0x3F (after an assert). Insert a Sleep between writes to widen the pulse width. An interval of 100 msec is adequate to reset all Pamux hardware.
Both LEDs Off	Write	BAR2	n/a*	0x00
LED1 On; LED2 Off	Write	BAR2	n/a*	0x01
LED1 Off; LED2 On	Write	BAR2	n/a*	0x02
LED1 On; LED2 On	Write	BAR2	n/a*	0x03
Pamux Memory Map	Read/Write	BAR1	Protocol	0x00 through 0xFF (see bank address expansion)

* Data is ignored.

NOTE: All reads and writes must be byte-wide reads and writes.

APPENDIX B

Files for OEMs

If you are a developer for an original equipment manufacturer (OEM), you will need the files listed in the following table for system installation. Note that you can use the PCI Pamux Toolkit installation program to automatically install and configure these files.

Instructions are also included for the "Wdreg.exe" application, used for automatic installation programs or manual removal of the driver.

File Name	Location	Description and Notes
Windrvr.sys	\winnt\system32\drivers	Windows NT/2000 device layer driver
Windrvr.vxd	\windows\system\vmm32	Windows 95/98/Me device layer driver
Wdreg.exe	\winnt (NT/2000)	Registry management program for the device layer drivers and for the Windows registry. *wdreg install*—installs necessary registry keys into the Windows Registry and starts the device layer driver. This application is useful for installation programs to automatically configure the device layer driver. *wdreg remove*—stops the device layer driver and removes the appropriate keys from the Windows Registry. It is useful for removal programs to remove the driver from the system.
	\windows (95/98/Me)	Registry management program for the device layer drivers and for the Windows registry. *wdreg-vxd install*—installs necessary registry keys into the Windows Registry and starts the device layer driver. This application is useful for installation programs to automatically configure the device layer driver. *wdreg-vxd remove*—stops the device layer driver and removes the appropriate keys from the Windows Registry. It is useful for removal programs to remove the driver from the system.
OptoPM32.DLL	\winnt\system32 (NT/2000) \windows\system (95/98/Me)	Opto 22 DLL driver interface file for PCI Pamux (PCI-AC28) installations
PCI Locator.exe	\winnt (NT/2000) \windows (95/98/Me)	Application to help locate boards in systems with more than one PCI-AC28. By default, this file is installed into \Opto22\PCI Pamux Toolkit. It is a VB program and may require other Visual Basic files to properly run on other platforms.