PCI-AC51 AND PCIE-AC51 USER'S GUIDE

Form 1459-140612—June 2014

OPTO 22
Automation made simple.

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
Email: support@opto22.com
Web: support.opto22.com

PCI-AC51 and PCIe-AC51 User's Guide
Form 1459-140612—June 2014

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 newer 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. Opto 22 does not warrant any product, components, or parts not manufactured by Opto 22; for these items, the warranty from the original manufacturer applies. These products include, but are not limited to, OptoTerminal-G70, OptoTerminal-G75, and Sony Ericsson GT-48; see the product data sheet for specific warranty information. Refer to Opto 22 form number 1042 for complete warranty information.

Opto 22
Automation Made Simple.

Table of Contents

1: Introduction

The PCI-AC51 and PCIe-AC51 adapter cards bring industry-standard Pamux® to modern computers that use the Peripheral Component Interconnect (PCI) or PCI Express (PCIe) local bus.

- The PCI-AC51 is compatible with computers that feature a 5.0 or 3.3 volt PCI bus.
- The PCIe-AC51 is compatible with computers using the PCI Express bus.

These cards are an ideal choice for customers who must replace an older ISA bus-based PC that currently uses an Opto 22 AC28 adapter card.

NOTE: The PCI-AC28 is not recommended for new designs. Use the PCI-AC51 or the PCIe-AC51 instead.

Using the PCI-AC51 or PCIe-AC51 adapter card, your 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 brains.
- Each Pamux bus supports up to 512 points.

Free with these adapter cards is the Pamux Systems SDK, included on the CD that came with the card and also available on our website, www.opto22.com. The Pamux Systems SDK supports up to 32 adapter cards. Two SDKs are included on the CD:

- **Windows 8, Windows 7, and Vista** operating systems—samples, source code, and the driver. This SDK supports C#, VB.NET and other .NET languages. Existing applications n VB and Visual C++ can be updated to Win 7 using this SDK.
- **Microsoft Windows XP and 2000** operating systems—sample applications, utility applications, and the driver. This SDK supports Microsoft Visual Basic (version 6) and Microsoft Visual C++ (version 6). .

Use the DLL file for your OS, application, and language:

For this OS:	Windows 8, Windows 7, and Vista			Windows 95/98/ME/NT/2000/XP	
For existing or new application:	Existing app	New 32-bit app in C++	New app in C# or other .NET language	Existing app	New app in VB6 or Visual C++
Use this DLL file:	OptoPM32.dll	OptoPM32.dll	PCIAC5.dll	OptoPM32.dll	OptoPM32.dll

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, the *Pamux User's Guide*. If you are going to program the PCI-AC51 or PCIe-AC51 using the Pamux Systems SDK, this guide assumes that you are already familiar with programming in Microsoft.NET, Microsoft® Visual Basic®, or Visual C++®.

This guide includes the following chapters and appendixes:

Chapter 1: Introduction introduces the card and provides Product Support information.

Chapter 2: Installing the Card describes how to install the adapter card before you install the SDK software.

Chapter 3: Setting Up and Using the SDK describes how to set up and use the SDK.

Appendix A: Error Codes lists and describes the error codes that you might encounter when working with the Pamux Systems SDK.

Appendix B: Requirements, Specifications, & LEDs provides system requirements and specifications for the PCI-AC51 and PCIe-AC51 adapter cards. It also shows the location and function of card LEDs.

Appendix C: Converting Applications to a Newer Card describes how to convert applications that used the AC28 and PCI-AC28 adapter cards.

Appendix D: PCI-AC51 and PCIe-AC51 Technical Reference provides technical information for authoring a device driver for an unsupported operating system or a Windows kernel-mode driver.

For Help

If you have problems installing or programming the PCI-AC51 or PCIe-AC51 adapter card and cannot find the help you need in this guide, contact Opto 22 Product Support.

Phone:	800-TEK-OPTO (800-835-6786) 951-695-3080 (Hours are Monday through Friday, 7 a.m. to 5 p.m. Pacific Time)	*NOTE: Email messages and phone calls to Opto 22 Product Support are grouped together and answered in the order received.*
Fax:	951-695-3017	
Email:	support@opto22.com	
Opto 22 website:	www.opto22.com	

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 and/or numbers seen.

2: Installing the Card

Follow the steps below to install the PCI-AC51 or PCIe-AC51 adapter card before you install the SDK software. Use any available expansion slot on the computer.

You may add multiple adapter cards for convenience, but note that multiple cards do not increase Pamux throughput. The number of Pamux accesses per computer is constant. **Note the power requirements** listed in the Specifications table on page 33.

Choose the installation steps for your card:

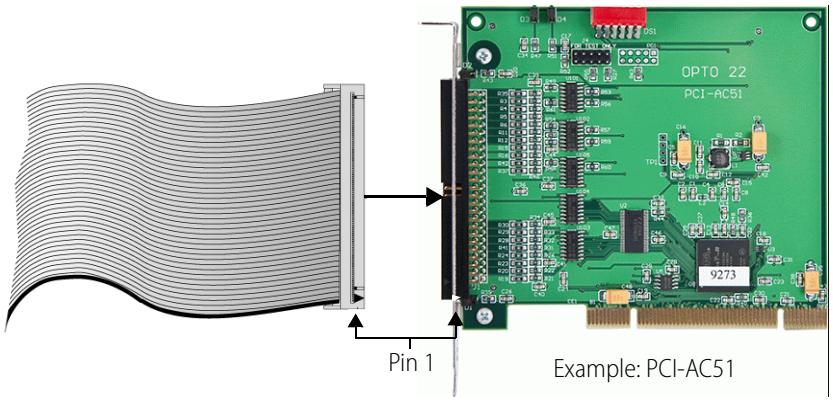
Installing the PCI-AC51

IMPORTANT: *Install the adapter card before you install the SDK software.*

You can install the PCI-AC51 adapter card into any PCI expansion slot on your computer.

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

   The power cord must be removed, as a sudden spike may cause the computer to boot.
2. Before handling the card, discharge excess static electricity by touching the computer's metal chassis.
3. Install the card in the expansion slot. Verify that the card is properly seated in the motherboard socket. Secure the card with the screw.
4. Attach the cable to the adapter card. Connect the other end of the cable to the I/O mounting rack. See the *Pamux User's Guide* (form 0726) for more information.



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

5. Reinstall the power cord. If you wish, leave the computer cover off temporarily to see the card's LEDs.
6. Turn on the computer.

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

   If you are using Windows Vista or an earlier version of Windows and a "Found New Hardware" message appears, simply click Cancel. The setup program will install the device driver automatically.

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 adapter card. See Appendix C: Converting Applications to a Newer Card for specific information.

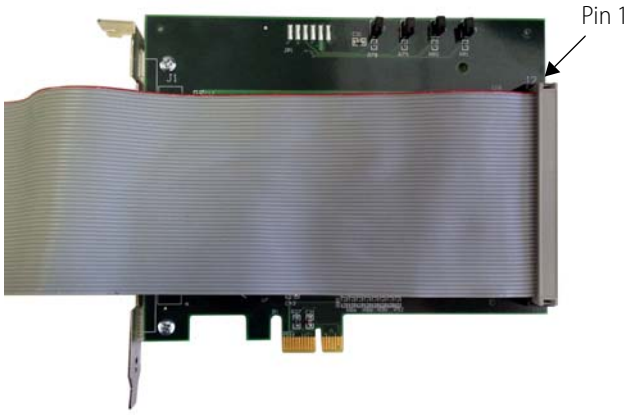
Installing the PCIe-AC51

IMPORTANT: *Install the adapter card before you install the SDK software.*

On your computer, install the PCIe-AC51 into any PCIe expansion slot.

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

   The power cord must be removed, as a sudden spike may cause the computer to boot.
2. Before handling the card, discharge excess static electricity by touching the computer's metal chassis.
3. Starting with the connector end that allows the cable to lie flat, push the ribbon cable through one of the slot openings in the computer.
4. Run the cable across the top of the adapter card (so it covers the card) and attach the cable to the card, as shown.



5. Install the card in the expansion slot. Verify that the card is properly seated in the motherboard socket. Secure the card with the screw.
6. Connect the other end of the cable to the I/O mounting rack. See the *Pamux User's Guide* (form 0726) for more information.

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

7. Reinstall the computer's cover and power cord.
8. Turn on the computer.

   Configuration is automatically performed by the BIOS when the card is installed.

   If you are using Windows Vista or an earlier version of Windows and a "Found New Hardware" message appears, simply click Cancel. The Pamux Systems SDK setup program will install the device driver automatically.

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 adapter card. See Appendix C: Converting Applications to a Newer Card for specific information.

3: Setting Up and Using the SDK

This chapter describes how to set up and use the Pamux Systems SDK. Before you install the SDK software, first make sure to install the adapter card. See Chapter 2: Installing the Card.

In this chapter:

NOTE: Version 2.0 and higher support only the PCI-AC51 or PCIe-AC51—not the AC28 and PCI-AC28 adapter cards.

Running the SDK Setup Program

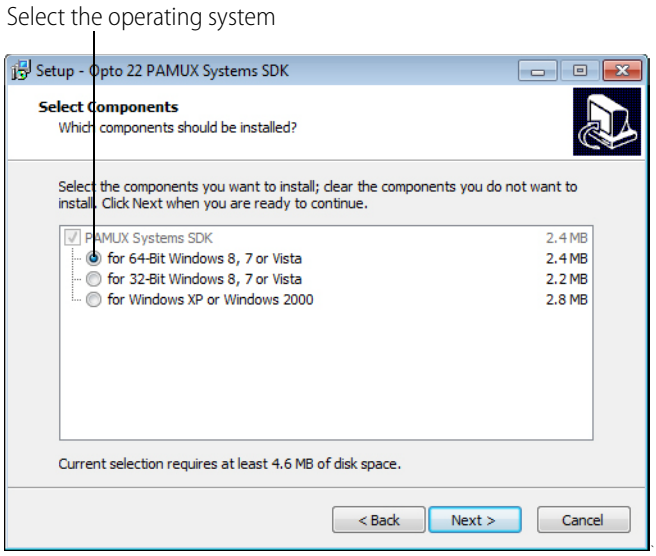
The Pamux Systems SDK comes on the CD with the card. If you do not have the CD, you can order it through Product Support, or you can download the SDK free from our website, www.opto22.com.

The SDK includes a sample application, Windows drivers, and user documentation.

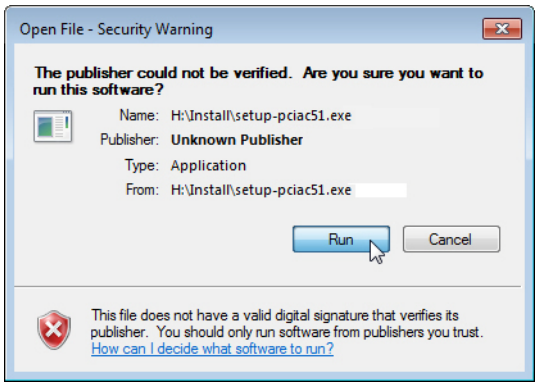
NOTE: If you are using a different operating system or a non-PC hardware platform, you will need to write your own driver. See Appendix D: PCIe/PCI-AC5 Register Set and Memory Locations and Appendix E: AC5 Hardware Description for more information.

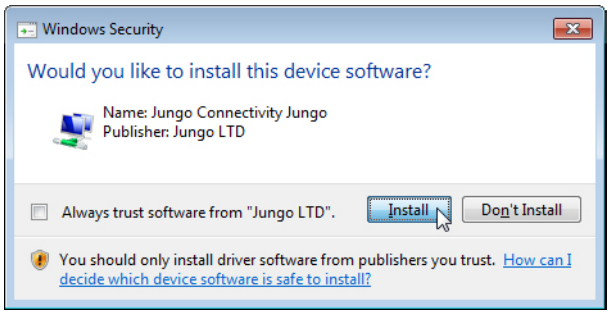
1. Insert the CD in your CD-ROM drive to start the Pamux Systems SDK setup wizard.
2. Follow the wizard's instructions.
3. On the Select Components screen, select your computer's operating system.

For Windows 8, Windows 7, and Vista, choose the 64-bit version or the 32-bit version.



4. If a Windows Security dialog box appears, click the Run or Install button to continue.





NOTE: If you have already installed the driver for another adapter card, you will receive an error that there is currently an open application using WinDriver. If this happens, simply click Cancel to continue installing.

5. When setup is complete, click the Finish button.

Using the SDK on Windows 8, Windows 7, and Vista

SDK Contents

The Pamux Systems SDK includes a driver, sample applications, and tools to help you develop applications for the PCI-AC51 or PCIe-AC51 adapter card and integrate it with your system. The SDK saves you time and effort that would otherwise be spent learning the intricacies of the Pamux bus structure.

The SDK supports C#, VB.NET and other .NET languages

NOTE: Version 2.0 and higher support only the PCI-AC51 or PCIe-AC51—not the AC28 and PCI-AC28 adapter cards.

Using the PamScan Utility

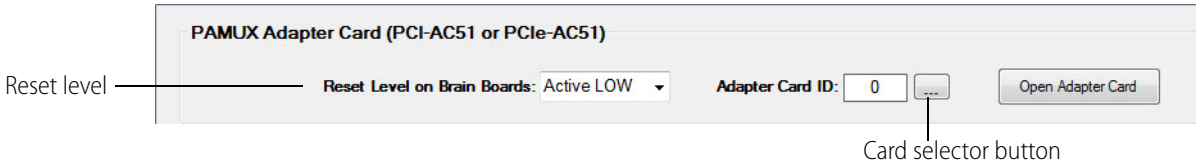
The PamScan utility is available on the Windows Start menu. It demonstrates turning digital points on and off and configuring points as either input or output. In addition, analog point values can be displayed and output values can be changed. PamScan is written in C# and all the source code is included.

NOTE: Instructions are the same for the PCI-AC51 and the PCIe-AC51, although only the PCI-AC51 is shown.

To use PamScan:

1. Select Start > All Programs > Opto 22 > PCI-AC51 > PamScan.exe

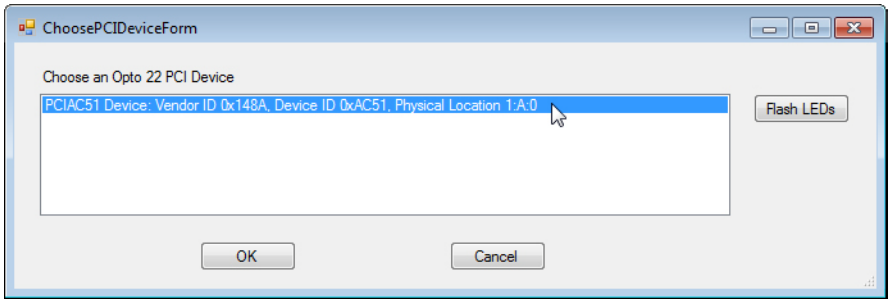
2. Make sure the reset level matches the reset level configured on the Pamux brain.



3. If you have more than one PCI-AC51 or PCIe-AC51card installed, click the card selector button.

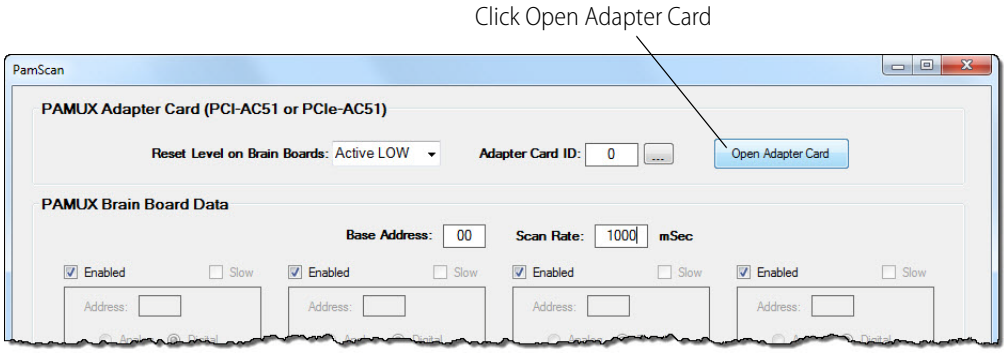
   NOTE: Normally you will use the default Adapter Card ID, which is 0.

4. In the dialog box that opens, select a card.

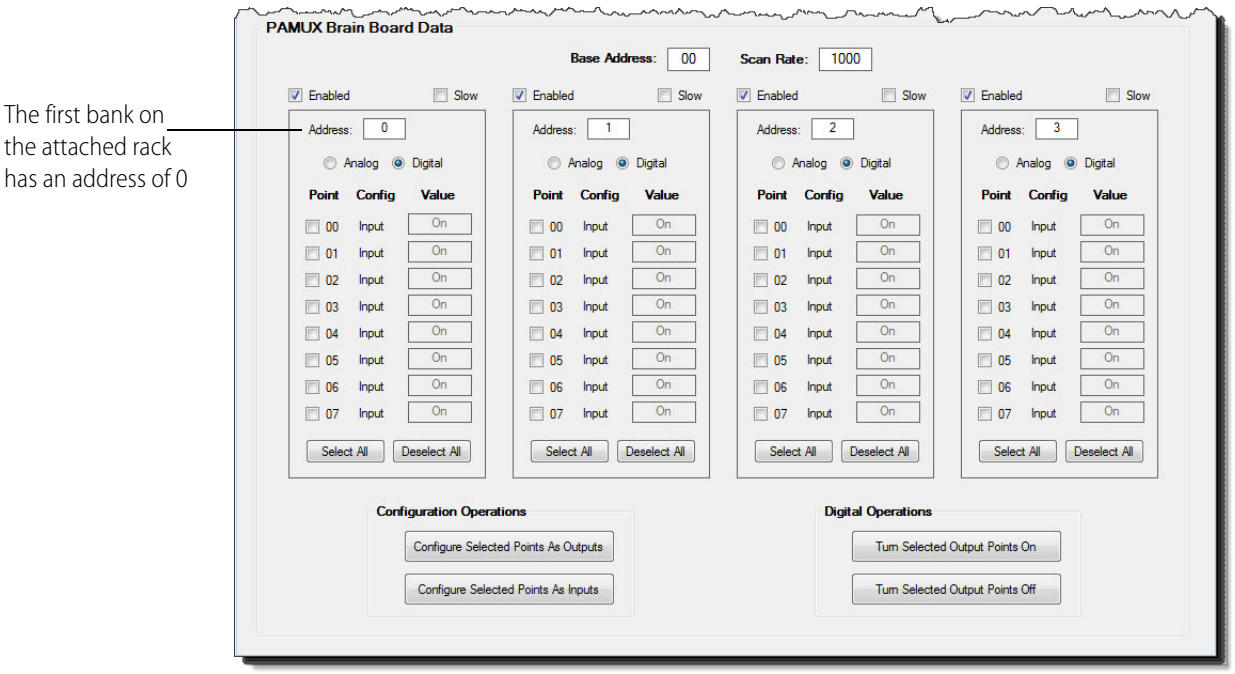


5. *Optional Step—Flash LEDs: This step is optional because it requires you to open your computer and see the LEDs on the card.*
   a. With your computer open so that you can see the LEDs, choose the card, and then click the Flash LEDs button.
   b. Notice the result on the card's LEDs.

      The program flashes the LEDs five times. If no flashing occurs, you may have selected the wrong card. Select the next card in the list and click the Flash LEDs button again.
6. Click OK to close the dialog box. In PamScan, click Open Adapter Card.

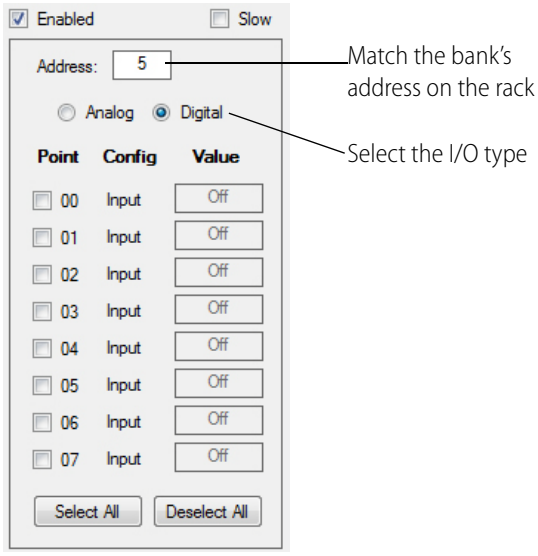


This screen shows all four banks available on a 32-channel Pamux board. Each bank has eight I/O points and has a unique address. For example, the banks shown here have addresses 0, 1, 2, and 3 on the attached I/O mounting rack.



7. Choose a bank on the Pamux mounting rack that has I/O installed. Then in the PamScan window, make sure the address matches the bank's address on the mounting rack.

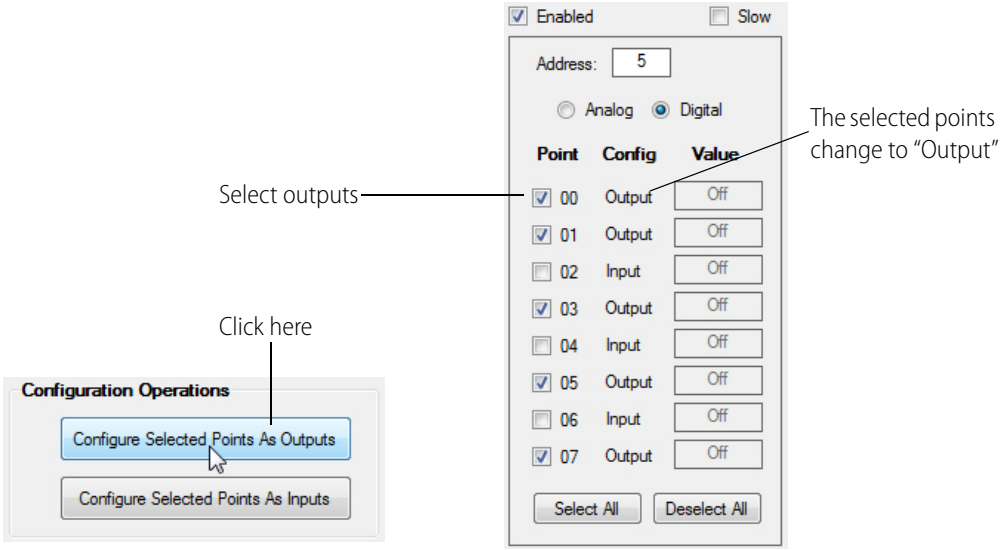
In the following example, 5 is entered as the address to match the address on the mounting rack:



8. Select the I/O type, either Analog or Digital.

9. For *output* modules: select each point that is an output, then click Configure Selected Points As Outputs.

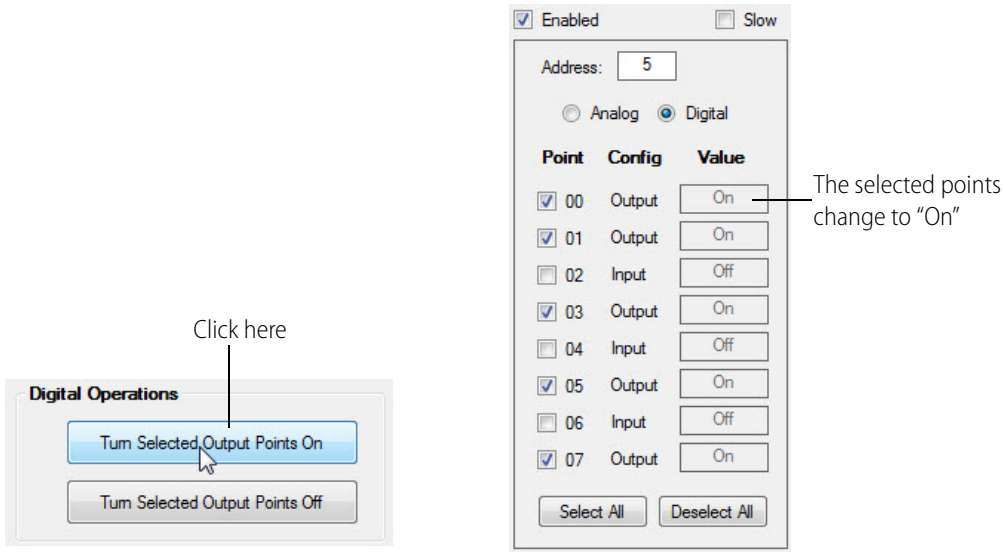
Notice that the selected points change to "Output."



For *input* modules: when an input changes state, its value will be updated in the display.

10. With the same output points selected, click Turn Selected Output Points On.

Notice that the selected output points change from "Off" to "On."



11. Continue to experiment as you like with selecting points, configuring points as inputs or outputs, and turning points on and off.

Developing an I/O Application

Before you begin:

- Add the PCIAC51.dll as a reference to your project.

- Near the top of your source code file, add the line: `using Opto22.Adapters.Pamux`

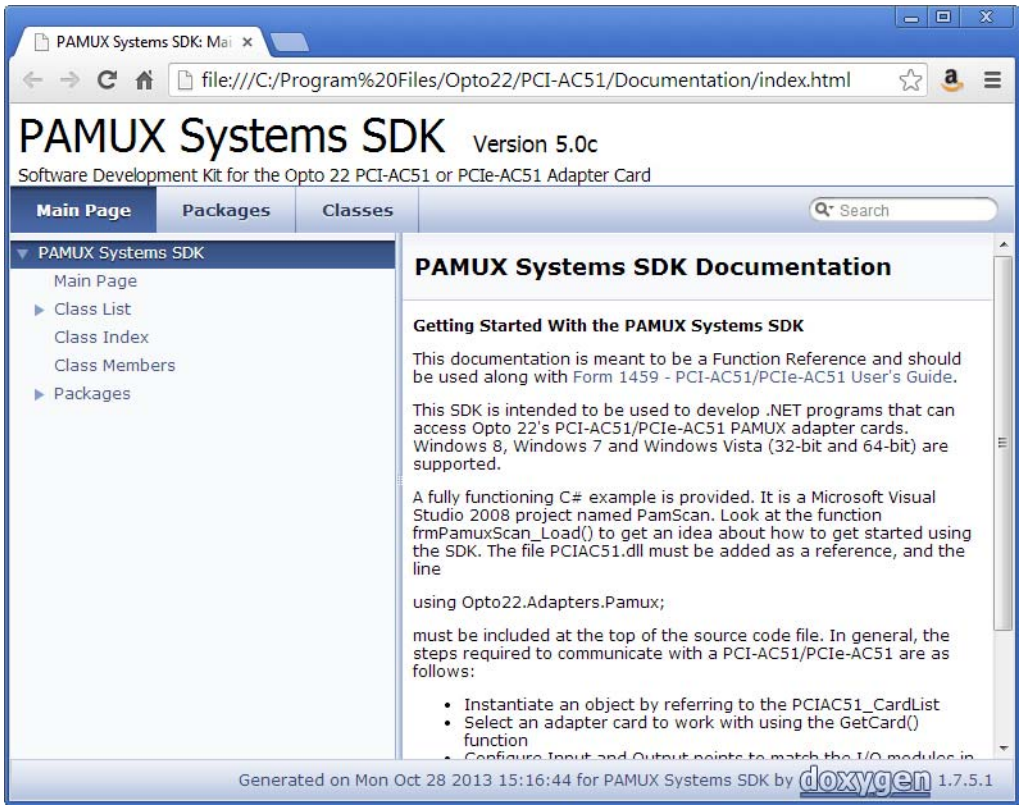
Your application should include the following basic steps.

1. Instantiate the PCIAC51_CardList object by accessing the public property TheCardList, for example: `pciac51cards=PCIAC51_CardList.TheCardList`
2. Call GetCard ( ) on the PCIAC51_CardList object, for example: `myCard=pciac51Cards.GetCard (0)`
3. Inspect the error code of the GetCard ( ) function.
4. Configure input and output points using the WriteConfig ( ) function. Inspect the error code from the configuration function.
5. Start an application loop that continuously reads or writes points. At the same time, continue to inspect the error codes from the PCIAC51.dll.
6. When the application loop is complete, call ReleaseCard ( ) to close the card properly.

Function Reference for Windows 8, Windows 7, and Vista

The function reference for the Windows 8, Windows 7, and Vista SDK is available on your computer once you have installed the software (see "Running the SDK Setup Program" on page 9).

To open the function reference, go to the Start menu on your computer and select Start > All Programs > Opto 22 > PCI-AC51 > PCI-AC51 and PCIe-AC51 Function Reference. If your browser blocks the ActiveX controls, make sure to unblock them so that you can see the navigation pane.



Porting Programs Developed with the Older SDK (Using OptoPM32.dll) to Windows 8, Windows 7, and Vista

Programs developed previously with the older OptoPM32.dll SDK will not work with Windows 8, Windows 7, or Vista due to an incompatibility at the driver level of these operating systems. If you want to convert your existing program to work on Windows 8, Windows 7, or Vista, follow these steps:

1. Copy the files from the existing application to the Windows 8, Windows 7, or Vista computer.
2. Install the Pamux Systems SDK on the same computer (see "Running the SDK Setup Program" on page 9).

   The setup program will install the appropriate files and driver for the target operating system.
3. Use the files in the directory listed below based on the target operating system:
   - For 64-bit Windows 8, Windows 7, or Vista
     C:\Program Files\Opto22\PCI-AC51\Legacy-OptoPM32\FilesFor32BitAppsOn64BitSystems
   - For 32-bit Windows 8, Windows 7, or Vista
     C:\Program Files\Opto22\PCI-AC51\Legacy-OptoPM32\FilesFor32BitAppsOn32BitSystems
4. (For Development) If you want to compile the old program on Windows 8, Windows 7, or Vista, use the files as follows. If you just want to run the old program, skip to the next step.
   - OptoPM32.h – Include this file in your existing application.
   - OptoPM32.lib – Link this file into your existing application.
5. (For Deployment) To run the old program on Windows 8, Windows 7, or Vista, use the files as follows.
   - OptoPM32.dll – Put this file in the same directory as your application.
   - wdapi1140.dll – Put this file in the same directory as your application.

   NOTE for VB Applications: Be sure to register any OCX files. For example, to register the file, COMCT232.OCX, open a command prompt with "run as administrator" permissions. Navigate to the directory where COMCT232.OCX resides, type `regsvr32 COMCT32.OCX` *and then press Enter. If successful, a dialog box will notify you.*

Using the SDK on Windows 2000/XP

Use this section if you have Windows 2000/XP. If you are using the SDK for Windows 8, Windows 7, or Vista, see "Using the SDK on Windows 8, Windows 7, and Vista" on page 11.

This section shows you what's in this version of the Pamux Systems SDK and how to use it to simplify application development with the PCI-AC51 or PCIe-AC51 adapter card.

NOTE: If you are migrating current applications to use with the newer PCI or PCIe card, also see Appendix C: Converting Applications to a Newer Card.

This version of the SDK runs in 32-bit Windows XP as well the 32-bit version of Windows 2000. The SDK supports Microsoft Visual Basic (version 6) and Microsoft Visual C++ (version 6).

SDK Contents

The Pamux Systems SDK includes a driver, sample applications, and tools to help you develop applications for the PCI-AC51 or PCIe-AC51 adapter card and integrate it with your system. The driver saves you time and effort by providing a simplified interface between input/output modules and application programs written in Microsoft Visual C++ and Visual Basic.

In addition to the driver, the SDK includes sample Visual Basic and Visual C++ applications as well as tools to locate, test, and diagnose the adapter card. All examples include source code.

Pamux Functions

Required Function Calls

For many applications, only four Pamux functions are required:

1. Open a PCI-AC51 or PCIe-AC51 to get a handle.
2. Configure outputs.
3. Read and write to I/O.
4. Close the PCI-AC51 or PCIe-AC51 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-AC51 or PCIe-AC51 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 hHandle` - Handle to a PCI-AC51 or PCIe-AC51 card. Handles are acquired usingPamuxPCICardOpen().

`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 29.

Developing an 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 SDK under \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 SDK under \Vc\VC Project Includes.

Function Reference for Windows 2000/XP

The functions listed in this section include parameters and descriptions. All descriptions that mention the PCI-AC51 also apply to the PCIe-AC51.

PCI-AC51 or PCIe-AC51 Operations

Function Type	Function	Parameter Type	Parameters	Description
long	PamuxPCICardOpen	long long long	* phHandle BoardID ResetLevel	Opens access to a PCI-AC51 or PCIe-AC51 card. phHandle gets a handle. BoardID is the PCI-AC51 ID. The first PCI-AC51 is at ID 0. ResetLevel: 1 = high reset; 0 = low reset. Set the reset level to match the reset level configured on the Pamux brains. This function also performs a Pamux bus reset.
long	PamuxPCICardOpenNoReset	long long long	* phHandle BoardID ResetLevel	Opens access to a PCI-AC51 or PCIe-AC51 card. phHandle gets a handle. BoardID is the PCI-AC51 ID. The first PCI-AC51 is at ID 0. ResetLevel: 1 = high reset; 0 = low reset. Set the reset level to match the reset level configured on the Pamux brains. This function does not reset the Pamux bus.
long	PamuxPCICardClose	long	hHandle	Releases the handle to the card and turns on LEDs 1 and 2.
long	PamuxCardReset	long	hHandle	Resets the card and resets the Pamux bus.
long	PamuxReadType	long	hHandle Bank *pType	Accesses the board ID for the specified bank. **Note:** Not all Pamux boards support the self-identification feature.
long	PamuxReadTypeNoRead	long	hHandle Bank *pType	Accesses the board IP for the last Pamux Board Read (does not access the Pamux Bus). **Note:** Not all Pamux boards support the self-identification feature.

* 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	hHandle 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	hHandle Bank OutputMask	
long	PamuxDigBank32Config	long long long	hHandle Bank OutputMask	
long	PamuxDigBankRead	long long long	int hHandle 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 hHandle Bank * pData	
long	PamuxDigBank32Read	long long long	int hHandle Bank * pData	
long	PamuxDigBankWrite	long long long	hHandle 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	hHandle Bank Data	
long	PamuxDigBank32Write	long long long	hHandle 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	hHandle 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	hHandle 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	hHandle 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	hHandle * pBank * pPoint * pIoPort	Provides the Pamux metrics needed: the PCI-AC51 or PCIe-AC51 handle, the bank number, and the point number.
long	PamuxDirectRead	long long long	hHandle Bank *Data	Reads one byte (eight bits) from the specified I/O port.
void	PamuxDirectWrite	long long long	hHandle 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	hHandle 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	hHandle 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	hHandle 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	hHandle 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	hHandle Bank Point * pData	Reads the value of an analog point.
long	PamuxAnaPointWrite	long long long long	hHandle 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	hHandle 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	hHandle 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	hHandle 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.

Bit Operations

Function Type	Function	Parameter Type	Parameter	Description
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.

Pack/Unpack Utility Operations

Function Type	Function	Parameter Type	Parameter	Description
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.

Scaling Operations

Function Type	Function	Parameter Type	Parameter	Description
long	PamuxUtilScaleI2l	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	PamuxUtilScaleF2l	float	X1	
		float	X2	
		long	Y1	
		long	Y2	
		float	Xin	

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

Pamux Bus Reset Line

This section explains the terminology and functioning of the Pamux bus reset line and its interaction between the PCI-AC51 or PCIe-AC51 card and the Pamux brains.

Purpose of the Reset Line

The purpose of the reset line on the Pamux bus is to provide a way to reset all the brains in the system to their default power up state. Digital brains will turn OFF the digital output modules and configure them as inputs. Analog brains will set all analog outputs to zero scale and configure them as inputs.

See the *Pamux User's Guide* (form 0726) for more details, including interaction with the watchdog feature: http://www.opto22.com/documents/0726_Pamux_Manual.pdf

Reset Line Voltage

The pinout of the Pamux bus (50-pin ribbon cable) is listed in Chapter 4 ("Programming without the Pamux driver") of the Pamux Users Guide. The reset line is pin 49 of the ribbon cable. All even-numbered pins are connected to logic ground. The state of the reset line can be verified by measuring the voltage between pin 49 and any even pin.

Terminology

"LOW" and "HIGH"

The reset line is considered to be "LOW" when the voltage on the reset line is LOW (~ 0 VDC).

The reset line is considered to be "HIGH" when the voltage on the reset line is HIGH (~ 3.1 to 5.0 VDC).

Brain Reset Configuration: "Active HIGH" and "Active LOW"

Each Pamux brain has a jumper to configure whether the brain will reset when the reset line is HIGH or LOW. This is referred to as reset "Active HIGH" or reset "Active LOW".

If the brains are jumpered for reset "Active HIGH", then the system will be "in reset" (the brains will be reset) when the reset line is HIGH. The brains will function normally (not be reset) when the reset line is LOW.

Similarly, if the brains are jumpered for reset "Active LOW", then the system will be "in reset" (the brains will be reset) when the reset line is LOW. The brains will function normally (not be reset) when the reset line is HIGH.

You must determine whether you want your Pamux system to operate with reset Active HIGH or reset Active LOW. Then consistently set the reset jumper on each brain in the system to match your choice; all brains must use the same reset configuration. One consideration is how you want your system to function if the computer loses power. Without power to the computer, the PCI-AC51 or PCIe-AC51 cannot drive the reset line high, so by default it will be low. You would want to analyze your system needs to determine what is necessary for your system to automatically go into failsafe mode.

Special Precautions for the Software Developer

No exclusive access—The Pamux Systems SDK permits only a single handle to a card. 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 cards simultaneously.

A: Error Codes

Error Messages for Windows 8, Windows 7, and Vista

If one of the function calls returns false, it is an indication that an error or warning has occurred. To inspect the message, use the lastError property on either the PCIAC51_CardList object or the PCIAC51_Card object, depending on what function was called.

All error or warning messages are also logged in the Application area of the Event Log. The messages will be grouped under the application name, "Opto 22 PCI-AC51."

Error Codes for Windows 2000/XP

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

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 may be 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 allo-cate	The PCI-AC51 or PCIe-AC51 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.

Code Decimal	Code Hex	Description	Remedy
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 (0xAC51) could not be found. Remember that board IDs start from zero and end at "n-1" (where n is the number of boards).
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-AC51 or PCIe-AC51 card, the handle's data structure is marked as open. The OptoPM32.DLL does not support multiple access handles to individual cards.
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 SDK layer installed.
8386	0x20C2	The device layer file version is too old for the SDK.	May happen if a SDK with an older WinDriver version is installed onto the system. Update all SDK installations with the latest SDKs.
8387	0x20C3	OptoPM32 does not support this function.	An unsupported PCI-AC51 and PCIe-AC51 function is called from the application. The card does not support the identify type and does not support a reset level function.

Code Decimal	Code Hex	Description	Remedy
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 SDK. 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.

B: Requirements, Specifications, & LEDs

This appendix provides system requirements, card specifications, and information on card LEDs.

System Requirements

Here's what you need to install and run the software provided for the PCI-AC51 and PCIe-AC51:

- A computer with x86 compatible processor, 1 GHz or higher. The most recent version of BIOS must be installed.
- Microsoft Windows 8 Professional (32-bit or 64-bit), Microsoft Windows 7 Professional (32-bit or 64-bit), Windows Vista® Business (32-bit), or Windows XP Professional (32-bit, with Service Pack 2 or higher)

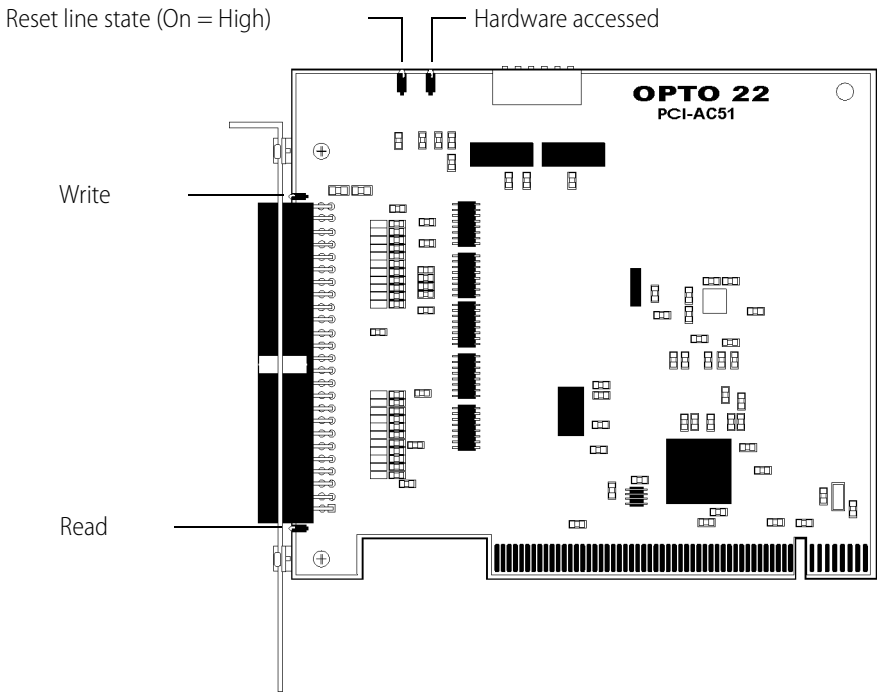
Specifications

	PCI-AC51	PCIe-AC51
Power requirements (provided by PC bus)	5 VDC @ 500 mA 3.3 VDC @ 300 mA	12 VDC @ 50 mA 3.3 VDC @ 500 mA
Computer compatibility	32 or 64-bit PCI bus	PCI Express x1 or higher
SDK compatibility	Microsoft Windows 2000/XP (supports Visual Basic 6 and Visual C++ 6) Windows 8, Windows 7, and Vista, 32-bit and 64-bit (supports C#, VB.NET, other .NET languages, VB, Visual C++)	
PCI compliance	PCI Specification Revision 2.2	PCI Express Specification Revision 1.1*
Opto 22 brain compatibility	B4, B5, B6, SNAP-B4, SNAP-B6	B4, B5, B6, SNAP-B4, SNAP-B6
LEDs	Four, indicating board access, reset level, read, and write	Four, indicating board access, reset level, read, and write
Operating temperature	0 to 70 °C	0 to 60 °C
Storage temperature	-30 to 85 °C	-30 to 85 °C

* Since PCI Express is backwards compatible, the card is also compatible with PCIe 2 and 3 slots.

LEDs

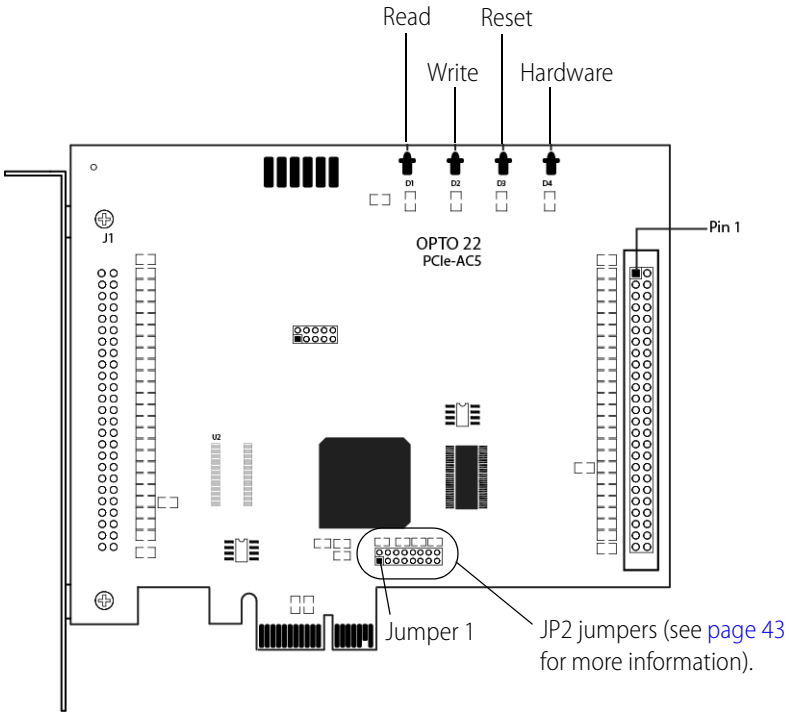
PCI-AC51



- **Reset** Indicates the state of the reset line, which can be either High or Low (On = High).
- **Hardware** Blinks when the PCI-AC51 is accessed.
- **Write** Blinks when a Pamux write is executed.
- **Read** Blinks when a Pamux read is executed.

NOTE: If the PCI-AC51 is rapidly or continually accessed, the blinking indicators may appear to stick on (will not blink). This is normal.

PCIe-AC51



- **Read** Blinks when a Pamux read is executed.
- **Write** Blinks when a Pamux write is executed.
- **Reset** Indicates the state of the reset line, which can be either High or Low (On = High).
- **Hardware** Blinks when the PCIe-AC51 is accessed.

NOTE: If the PCIe-AC51 is rapidly or continually accessed, the blinking indicators may appear to stick on (will not blink). This is normal.

C: Converting Applications to a Newer Card

This section describes how to convert applications that used the AC28 and PCI-AC28 adapter cards.

Applications that Used the OptoPMUX.DLL for the AC5

This version of the 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 see the *PCI-AC5 and AC5 User's Guide*, Opto 22 form #1211.

Converting Applications that Use Inp( ) and Outp( )

This PCI SDK library provides a consistent application code model for Windows 2000 and Windows XP operating systems. Inp() and Outp() function calls at the user level are unsupported because of the Windows hardware abstraction layer.

The PCI-AC51 and PCIe-AC51 interface 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. The AC28 relied on jumper settings for configuration; these jumpers are not included on the PCI-AC51 or the PCIe-AC51.

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.

D: PCI-AC51 and PCIe-AC51 Technical Reference

OPTO 22

This technical reference is provided for those who are either technically curious or are interested in authoring a device driver for an unsupported operating system or a Windows kernel-mode driver.

The information provided is *as is*; you use this information at your own risk. Opto 22 does not support any kernel layer or drivers other than those we provide.

In this appendix:

Architecture

The PCI-AC51 and PCIe-AC51 cards contain an Altera FPGA (U8 on the PCI-AC51 and U1 on the PCIe-AC51). The Altera FPGA implements the PCI or PCI Express interface to the host computer, regulating data transactions between the host computer and the adapter card while also providing adapter card configuration information to the host computer's BIOS. The FPGA also implements Opto 22's Pamux state machine, which drives the Pamux bus cycle and provides the register architecture visible to the software application.

Adapter Card Identification

PCI Vendor ID, 0x148a

PCI Device ID, 0xac51

Type "PCI Simple Communications Controller"

Adapter Card Base Address Configuration

Base Address Register 1 (BAR1) is memory mapped. It is configured as a 16 consecutive byte-only array. Only the first 7 bytes are used. Accessing unmapped registers may cause the computer to bus lock.

BAR1 Register Table

An X indicates that data is ignored.

Offset	Direction	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x00	R	Reset Status Register	Reset Busy	X	X	X	X	X	X	Reset Level
0x00	W	Reset Control Register	X	X	X	X	X	X	X	Reset Level
0x01	R/W	Bank Address Register	X	X	Pamux A5	Pamux A4	Pamux A3	Pamux A2	Pamux A1	Pamux A0
0x02	R/W	Data Register	Pamux D7	Pamux D6	Pamux D5	Pamux D4	Pamux D3	Pamux D2	Pamux D1	Pamux D0
0x03	R	Start Read Register	1	0	1	0	1	0	1	1
0x03	W	Start Write Register	X	X	X	X	X	X	X	X
0x04	R	Status Register	X	Read Status	Write Status	0	0	Pamux ID1	Pamux ID0	Pamux Busy
0x05	R	Switch/Jumper Register*	0	0	S6/J6	S5/J5	S4/J4	S3/J3	S2/J2	S1/J1
0x06	R	Version Register	D7	D6	D5	D4	D3	D2	D1	D0

* PCI-AC51 has switches. PCIe-AC51 has jumpers. See page 35 for jumper location.

Register Description

Reset Status Register (0x00) The Reset Status Register reports the reset state (Reset Level, bit 0) and the status of a 150 ms timer (Reset Busy, bit 7). The timer provides the application a way to ensure a minimum reset duration for all Pamux brains. The timer is reset (and the Reset Busy Bit set to 1) when register 0x00 (Reset Control Register) is written. When the duration expires, the Reset Busy Bit is set to 0.

Reset Control Register (0x00) Bit 0 drives the reset signal on the Pamux bus. Writing a 1 asserts the reset signal to the "high level." Writing a 0 asserts the reset signal to the "low level."

Bank Address Register (0x01) The Pamux address for the next Pamux read/write cycle is stored in this register. This register must not be modified while the Busy Bit of the Status Register is set to 1.

Data Register (0x02) Do not access this register while a Pamux operation is in progress.

Writing: The data to be written in the next Pamux write cycle should be stored in this register before the Pamux write cycle is finished.

Reading: After the Pamux read cycle is complete, this register holds the data that was read from the Pamux brain.

Start Read Register (0x03) A read initiates a Pamux read bus cycle. Reading from this register initiates a Pamux read. The bank Address Register must be configured before reading from this register. A value of 0xAB is always returned in this register and can be ignored. The data read from the Pamux brain is returned in the Data Register. Do not access this register while a Pamux operation is in progress.

Start Write Register (0x03) Writing to this register initiates a Pamux write cycle. The Bank Address Register and the Data Register must be configured before writing to this register. Do not access this register while a Pamux operation is in progress.

Status Register (0x04) This register contains the Busy bit (busy = 1), to indicate if a Pamux bus cycle is in progress. The ID bits hold the value of the last brain communicated with. These bits are valid only after the Busy bit is 0. The Write Status bit is 1 if the current or last Pamux bus cycle was a Pamux write cycle. The Read Status bit is 1 if the current or last Pamux bus cycle was a Pamux read.

Switch/Jumper Register (0x05) Provides state of the switches on the PCI-AC51 or state of the jumpers on the PCIe-AC51. Custom driver applications may use S1 through S4 (or J1 through J4) in any desired fashion. S5/J5 and S6/J6 may be verified for proper hardware settings.

Version Register (0x06) Returns the version of the state machine architecture.

Powerup Conditions

An X indicates that data is ignored.

Offset	Direction	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x00	R/W	Reset Control Register	X	X	X	X	X	X	X	0
0x01	R/W	Bank Address Register	X	X	X	X	X	X	X	X
0x02	R/W	Data Register	X	X	X	X	X	X	X	X
0x04	R	Status Register	X	0	0	0	0	X	X	0
0x05	R	Switch/Jumper Register*	X	X	S6/J6	S5/J5	S4/J4	S3/J3	S2/J2	S1/J1

* PCI-AC51 has switches. PCIe-AC51 has jumpers. See page 35 for jumper location.

Controlling the Reset Line

Controlling the Reset Line Using the Pamux Driver

When using the Pamux driver, it is necessary to tell the driver how the reset level is configured on the brains. This is done as part of the GetCard() function. Use one of the following values:

- eResetLevel.BrainsSetForResetActiveLow
- eResetLevel.BrainsSetForResetActiveHigh

You will then be able to reset the brains in the system or take the brains out of reset by using the following functions:

- ResetBrains()
  - Be sure to keep the brains in reset for at least 150 mSec to ensure all brains properly reset.
- ReturnToNormalAfterReset()

You can also reset the brains in the system and take the brains back out of reset using a single function, which does a 150 mSec rest:

- ResetBrainsAndReturnToNormal()

Controlling the Reset Line When Not Using the Pamux Driver

Bit 0 of the Reset Control Register on the PCI-AC51 and the PCIe-AC51 is used to control the reset line.

- When bit 0 is set to 0, the reset line will be HIGH and LED 1 will be ON.
- When bit 0 is set to 1, the reset line will be LOW and LED 1 will be OFF.

The reset line must be in the appropriate state (depending on the reset configuration on the brains) for a minimum of 150 mSec to ensure all brains reset.

Default Power-Up State of Reset Line

When a computer with a PCI-AC51 or PCIe-AC51 card installed has just been powered up, bit 0 of the reset control register will be in its default power-up condition, which is a bit value of 0. This will drive the reset line HIGH and turn LED 1 (reset line) ON.

Reset Configuration of Brains When Using PamScan

One of the parameters you must choose when using the PamScan utility is the reset level of the brains in your system. Set the option in PamScan to match how your Pamux brains are configured. PamScan needs to know the brain configuration so that it can determine whether to set the reset line HIGH or LOW in order to take the system out of reset. This allows PamScan to configure and control the brains and output modules.

Testing

The following table shows the results of using each of the possible reset settings on the brains and in the PamScan utility. After choosing the reset setting in PamScan, click the Connect button to establish communication with the PCI-AC51 or PCIe-AC51 card and brains.

Jumper 7 Setting on Brains	Pam Scan Configuration	Reset Line Voltage	State of Reset Line	Status
Set "Active Low"	Choose "Active Low"	3.48 VDC	HIGH	Brains allow control of outputs
Set "Active Low"	Choose "Active High"	0 VDC	LOW	Brains are reset
Set "Active High"	Choose "Active Low"	3.48 VDC	HIGH	Brains are reset
Set "Active High"	Choose "Active High"	0 VDC	LOW	Brains allow control of outputs

DIP Switches (PCI-AC51) and Jumpers (PCIe-AC51)

DIP switches or jumpers are provided on the adapter cards for custom implementation and some Pamux settings.

- The **PCI-AC51** has six DIP switches (the DS1 bank). A switch in the ON position reads as a zero and OFF reads as a one. The card is shipped with all switches in the ON position.
- The **PCIe-AC51** has eight jumpers (labeled JP2), of which only the first six are used. No jumper reads as a zero, and a jumper in place reads as a 1. The card is shipped with no jumpers. (Do not use the jumper group labeled J3. See the diagram on page 35 for jumper group location.)

DIP switches and jumpers may be modified on a system that is powered; however, the applications that use the PCI-AC51 or PCIe-AC51 must be stopped.

Switches or jumpers 1 through 4 are provided for custom implementation. One common use will be for identifying certain cards in a multiple card system. The settings of these switches or jumpers do not affect the Pamux operation.

Switches or jumpers 5 and 6 are used to set the Pamux strobe duration. Under most conditions, these do not have to be changed.

S5 (Switch)	Jumper 5	Strobe Duration
On	Out	2.7 µS*
Off	In	3.3 µS

*Factory default. Also the legacy AC28 (K Revision) Pamux bus timing.

S6 (Switch)	Jumper 6	Strobe Duration
On	Out	High Latency Mode**
Off	In	Low Latency Mode

** Factory Default

Strobe Duration

Legacy Pamux documentation indicates that the *on-time* specification of the read and write strobes were 2,000 ns. Through the lifecycle of Pamux hardware, this time was increased to 2,700ns (brain and AC28's with a Lattice chip).

When S5 is on (PCI-AC51) or jumper 5 is not installed (PCIe-AC51), the card maintains the last AC28 (Rev K) timing specification, 2,700 ns. Switching to the 3,300 ns setting may improve long cable issues.

High and Low Latency Modes

The amount of idle time between Pamux bus cycles has never been specified. Historic Pamux hardware assumed that there was adequate software delay time between each bank access. Recent PCs are capable of performing back-to-back bus cycles with nearly no delay (as little as 60 ns).

The PCI-AC51 (state machine version 2) and PCIe-AC51 enforce idle time between bus cycles. In high latency mode (S6 on, or jumper 6 not installed), 3,600 ns of idle time is added between each bus cycle. This means the busy bit may be busy for as much as 7,700 ns. The additional delay time is required for many B6 analog brains. These brains are software driven and need the additional time to properly operate.

In some cases of digital only systems (systems that contain only B4, SNAP-B4, or B5 brains), S6 may be turned off (low latency mode) if the additional latency is undesirable.

Performing Pamux Operations

Issuing Pamux Reset

The PCI-AC51 and PCIe-AC51 provide a level oriented reset control feature.

1. Assert the Pamux Reset by writing a value to the Reset Control Register (BAR1, offset 0x00), bit 0. A zero sets the Pamux reset level to low. A one sets the Pamux reset level to high.
2. Poll the status register and inspect the state of Reset Busy. Wait for the bit to negate (return to zero). The Reset Busy enforces a 150 ms delay interval, the recommended reset duration for all Pamux hardware. Digital-only brains may reset with a much shorter interval.
3. Negate the Pamux Reset signal by writing a complementary value to bit 0 of the Reset Control Register.

NOTE: Reset is asynchronous to all Pamux operations. Reset may be asserted while the Pamux bus cycle is in operation (Status Register Busy Bit is asserted). If a Pamux read was in progress, the result is uncertain. If a write is in progress, the reset must be asserted well after the Pamux bus cycle is completed. If the reset is negated before the end of the current Pamux write, the written result is uncertain.

NOTE: A substantial reset period is required for analog or mixed brains. Typically, a Pamux analog brain will reset if the Reset is asserted for 150 ms. The Reset Busy bit in the Reset Status Register provides a 150 ms busy indication. Note this is simply an indicator. Additional changes to the reset state will immediately appear on the Pamux Reset signal and the 150 ms timer is reset upon each write to the Reset Control Register.

Reading the State Machine Version

Read the Version Register. All 8 bits of the register are significant to the state machine's implementation revision.

This register is independent of all Pamux operations. It may be read without impact to any current operations the state machine is performing.

Reading the Switch Values

The bit positions in the register correspond to the switches or jumpers on the board:

- PCI-AC51—A switch in the on position reads as a zero.
- PCIe-AC51—An uninstalled jumper reads as a zero.

Polled PCI-AC51 and PCIe-AC51 Operation

NOTE: All of the directions assume a byte-wide access to the card.

Performing a Pamux Bank Write

1. Wait until the Busy Bit in the Status Register is negated (zero).
2. Write the Pamux bank address to BAR1 offset 1.
3. Write the Pamux data to BAR1 offset 2.
4. A write to the Start Cycle Register (BAR1, offset 3) begins a Pamux bus cycle. The data written to the register is ignored.
5. Read the Status Register (BAR1, offset 4) and inspect the Busy Bit (Bit 0). If the bit is one, a Pamux cycle is in progress. Do not modify the Bank Address Register, Data Register, or the Start Cycle Register while the busy status is asserted. Wait until the Busy Bit negates.

6. To read the Pamux brain identification, read the Status Register (BAR1 offset 4) and mask bits 1 and 2. These bits are valid only after the Busy Bit is negated.

Performing a Pamux Bank Read

1. Wait until the Busy Bit in the Status register is negated (zero).
2. Write the Pamux bank address to the Bank Address Register (BAR1, offset 1).
3. Read the Start Read Register (BAR1, offset 3) to start a Pamux read bus cycle. Ignore the data read from this register (the returned data is hard coded as 0xab).
4. Read the Status Register (BAR1, offset 4) and inspect the Busy Bit (Bit 0). If the bit is one (asserted), a Pamux cycle is in progress. Wait until the Busy Bit negates (zero).
5. Read the Pamux data from the Data Register (BAR1, offset 2).

NOTE: Only specific Pamux brain versions support the ID function.

Notes for the SNAP-B6 (Digital Aspect)

The SNAP-B6's digital I/O is software driven. This is currently the only brain that uses software to emulate a hardware-only digital brain.

If operations between digital accesses are too fast, intermittent and unpredictable behavior may occur. If you should see this, here are workarounds and recommendations.

To correct this, please use the following recommendations:

1. Read and write banks to the SNAP-B6, only one bank at a time.
2. Allow a 2 millisecond delay before the next register operation occurs.
3. For PCIAC51.dll, use Set Slow Mode Delay( ) and Enable Slow Mode( ) to control the timing of operations. If you are using the OptoPM32.dll for the PCI-AC51 or PCIe-AC51, use the Pamux_Set_Slow_Bank_Range function.
4. If high-speed digital is required, use a SNAP-B4 and separate the analog and digital I/O.

Alternate Method

(OptoPM32.dll only) Create a text file named "o22_pamux_slow_bank.txt" in the folder where the application starts. There are two numbers in this file, as follows:

(starting bank number) (ending bank number)

So, if you have SNAP-B6's located say at address 32 and ending at bank 40 (decimal addresses), then you would want to set the numbers to be:

32 40

If you have multiple adapters, this delay applies to all PCI-AC51 or PCIe-AC51 adapters.