

---

Keithley KPCI-ISO Series

# Using DriverLINX with Your Hardware

**KEITHLEY**

Information in this document is subject to change without notice. The software described in this document is furnished under a license agreement. The software may be used or copied only in accordance with the terms of the agreement.

SCIENTIFIC SOFTWARE TOOLS, INC. SHALL NOT BE LIABLE FOR ANY SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RELATED TO THE USE OF THIS PRODUCT. THIS PRODUCT IS NOT DESIGNED WITH COMPONENTS OF A LEVEL OF RELIABILITY SUITABLE FOR USE IN LIFE SUPPORT OR CRITICAL APPLICATIONS.

This document may not, in whole or in part, be copied, photocopied, reproduced, translated or reduced to any electronic medium or machine readable form without prior written consent from Scientific Software Tools, Inc.

Keithley KPCI-ISO Series: Using DriverLINX with your Hardware  
© Copyright 2002, Scientific Software Tools, Inc.  
All rights reserved.

SST 39-0202-2

DriverLINX, SSTNET, and LabOBJX are registered trademarks and DriverLINX/VB is a trademark of Scientific Software Tools, Inc.

All Keithley product names are trademarks or registered of Keithley Instruments, Inc. Microsoft and Windows are registered trademarks and Visual C++ and Visual Basic are trademarks of Microsoft Corporation.

Borland is a registered trademark and Borland C++ and Delphi are trademarks of Borland International, Inc.

All other brand and product names are trademarks or registered trademarks of their respective companies.

# Contents

<b>Preface</b>	<b>7</b>
About DriverLINX.....	7
About This User's Guide .....	7
Conventions Used in This Manual .....	9
<b>Configuring the KPCI-ISO Series</b>	<b>11</b>
Introduction.....	11
Configure DriverLINX Device Dialog.....	11
Device Subsystem Page .....	13
Digital Input Subsystem Page .....	15
Digital Output Subsystem Page .....	17
<b>Programming the KPCI-ISO Series</b>	<b>19</b>
Selecting an API.....	19
Choosing DriverLINX .....	19
Choosing Hardware Direct I/O .....	20
Choosing ActiveX Direct I/O .....	21
Installing an API .....	21
Installing DriverLINX .....	22
Installing the Hardware Direct I/O Interface.....	22
Installing the ActiveX Direct I/O Interface.....	23
Programming an API.....	24
Programming DriverLINX.....	24
Programming the Hardware Direct I/O Interface.....	24
Programming the ActiveX Direct I/O Interface .....	24
<b>Accessing KPCI-ISO Series Features using DriverLINX</b>	<b>29</b>
Introduction.....	29
KPCI-ISO Series Hardware Features.....	30
Differences Between the PCI and ISA Models .....	31
<b>Using DriverLINX with the KPCI-ISO Series</b>	<b>35</b>
Introduction.....	35
DriverLINX Hardware Model for the KPCI-ISO Series.....	35
DriverLINX Subsystems.....	35
DriverLINX Modes .....	36
DriverLINX Operations and Events .....	37
Logical Channels .....	38
Buffers .....	38
Connecting Signals to the KPCI-ISO Series .....	39
Digital Input Subsystem Signals .....	39

Digital Output Subsystem Signals.....	39
Counter/Timer Subsystem Signals.....	40

## **Using the Device Subsystem 41**

Introduction.....	41
Device Modes.....	42
Device Operations.....	42

## **Using the Digital Input Subsystem 43**

Introduction.....	43
Digital Input Modes.....	44
Digital Input Operations.....	44
Digital Port Configuration.....	45
Digital Input Timing Events.....	46
None or Null Event.....	46
Rate Event.....	46
Rate Generator: External Clocking.....	47
Digital Event.....	48
Digital Input Start Events.....	49
None or Null Event.....	49
Command Event.....	49
Post-Trigger Sampling with a Digital Event.....	49
Digital Input Stop Events.....	50
None or Null Event.....	51
Command Event.....	51
Terminal Count Event.....	51
Pre-Trigger Sampling with a Digital Event.....	51
Digital Input Channels.....	52
Digital Input Logical Channels.....	54
Single Channel Digital Input.....	55
Multi-channel Digital Input Range.....	55
Multi-channel Digital Input List.....	56
Extended Logical Channel Addressing.....	57
Bit-Masked Addressing.....	58
Digital Input Buffers.....	58
Digital Input Messages.....	59

## **Using the Digital Output Subsystem 61**

Introduction.....	61
Digital Output Modes.....	62
Digital Output Operations.....	62
Digital Output Timing Events.....	63
None or Null Event.....	63
Rate Event.....	63
Rate Generator: External Clocking.....	63
Digital Event.....	64
Digital Output Start Events.....	66
None or Null Event.....	66
Command Event.....	66
Post-Trigger Sampling with a Digital Event.....	66
Digital Output Stop Events.....	67
None or Null Event.....	68

Command Event.....	68
Terminal Count Event.....	68
Pre-Trigger Sampling with a Digital Event.....	68
Digital Output Channels.....	69
Digital Output Logical Channels .....	70
Single Channel Digital Output .....	70
Multi-channel Digital Output Range.....	71
Multi-channel Digital Output List.....	72
Extended Logical Channel Addressing .....	73
Bit-Masked Addressing .....	74
Digital Output Buffers.....	74
Digital Output Messages .....	75
<b>Using the Counter/Timer Subsystem</b>	<b>76</b>
Introduction.....	76
Counter/Timer Channels .....	76
Pacer Clock.....	76
Counter/Timer Messages.....	77
<b>Uninstalling DriverLINX</b>	<b>79</b>
How do I uninstall DriverLINX? .....	79
<b>Troubleshooting</b>	<b>83</b>
Solving Problems .....	83
Solving Problems Recognizing and Installing Drivers.....	83
Solving Problems Configuring the Drivers .....	84
Solving Problems Loading Drivers .....	84
Generating a DriverLINX Configuration Report .....	87
What is in the Report? .....	87
How do I Generate the Report? .....	87
<b>Glossary of Terms</b>	<b>89</b>



# Preface

---

## About DriverLINX

Welcome to DriverLINX® for Microsoft® Windows™, the high-performance real-time data-acquisition device drivers for Windows application development.

DriverLINX is a language- and hardware-independent application-programming interface designed to support hardware manufacturers' high-speed analog, digital, and counter/timer data-acquisition boards in Windows. DriverLINX is a multi-user and multitasking data-acquisition resource manager providing more than 100 services for foreground and background data acquisition tasks.

Included with your DriverLINX package are the following items:

- The DriverLINX API DLLs and drivers supporting your data-acquisition hardware
- AIO Panel, multifunction instrument for testing analog and digital I/O features of your Keithley hardware
- Source code for the sample programs
- The DriverLINX Application Programming Interface files for your compiler
- DriverLINX On-line Help System
- *DriverLINX Digital I/O Programming Guide*
- *DriverLINX Technical Reference Manual*
- Supplemental Documentation on DriverLINX and your data acquisition hardware

---

## About This User's Guide

The purpose of this manual is to help you quickly learn how to configure and use the hardware features of Keithley's KPCI-ISO Series with DriverLINX.

- For help installing and configuring your hardware and DriverLINX, please see the manual that accompanied your hardware.

- For more information on the DriverLINX API, please see the *DriverLINX Technical Reference Manual* and the *DriverLINX Digital I/O Programming Guide*.
- For additional help programming your board, please examine the source code examples on the Distribution Media.

This manual contains the following chapters:

### **Configuring the KPCI-ISO Series**

Describes configuring a KPCI-ISO board using the *Configure DriverLINX Device* dialog box.

### **Programming KPCI-ISO Series Hardware**

Describes the available programming options for the KPCI-ISO Series.

### **Accessing KPCI-ISO Features Using DriverLINX**

Describes how DriverLINX supports hardware features of the KPCI-ISO Series.

### **Using DriverLINX with the KPCI-ISO Series**

Describes the subsystems, modes, and operations that DriverLINX supports for the KPCI-ISO Series.

### **Using the Device Subsystem**

Describes using DriverLINX to initialize KPCI-ISO hardware.

### **Using the Digital Input Subsystem**

Describes using DriverLINX to perform KPCI-ISO digital input.

### **Using the Digital Output Subsystem**

Shows using DriverLINX to perform KPCI-ISO digital output.

### **Using the Counter/Timer Subsystem**

Describes DriverLINX's support for timing operations with the KPCI-ISO Series.

### **Uninstalling DriverLINX**

Describes how to remove DriverLINX from your computer.

### **Troubleshooting**

Describes potential installation and configuration problems and their solutions.



---

## Conventions Used in This Manual

The following notational conventions are used in this manual:

- Itemized lists are identified by a round bullet (•).
- Numbered lists indicate a step-by-step procedure.
- DriverLINX Application Programming Interface and Windows macro and function names are set in bold when mentioned in the text.
- **DriverLINX** indicates the exported function name of the device driver DLL while DriverLINX indicates the product as a whole.
- DriverLINX Application Programming Interface identifiers, menu items, and Dialog Box names are italicized when mentioned in the text.
- *Italics* are used for emphasis.
- Source code and data structure examples are displayed in Courier typeface and bounded by a box with a single line.

Code

- Tables of information are bounded by a box with a double line.

Tables

*Concept*

- Important concepts and notes are printed in the left margin.



# Configuring the KPCI-ISO Series

---

## Introduction

The installation program provides general instructions for installing and configuring DriverLINX. This manual explains the steps and special features that apply only to Keithley's KPCI-ISO Series boards.

Installing and configuring DriverLINX for the Keithley KPCI-ISO Series requires three steps:

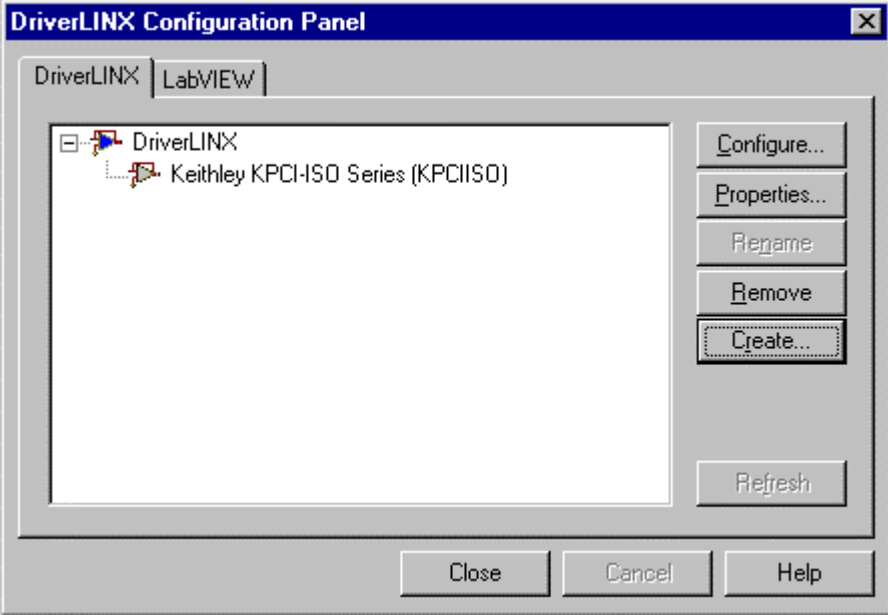
1. **To install DriverLINX**, insert the DriverLINX CD-ROM into your drive and follow the on-screen instructions.
2. **To install your KPCI-ISO hardware**, read and follow the instructions in the Keithley *KPCI-ISO PCI Bus Digital I/O Board User's Manual*.
3. **To configure DriverLINX**, use the *DriverLINX Configuration Panel* and consult the on-line help. Also see "Configure DriverLINX Device Dialog" on page 11 for configuration options specific to a Keithley KPCI-ISO board.

---

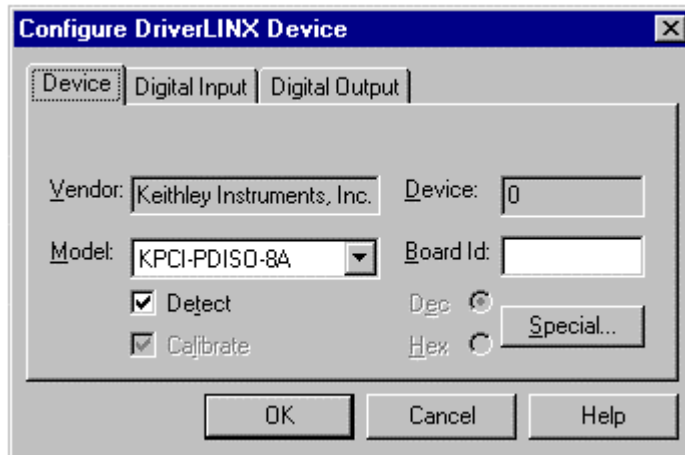
## Configure DriverLINX Device Dialog

DriverLINX uses a standardized configuration protocol for all data-acquisition hardware. It associates a board and any configuration options that you specify with a numbered Logical Device. Under Windows 2000 and XP, the driver creates a default configuration for new a board with the next available number. Under previous versions of Windows, you must use the DriverLINX Configuration Panel to create a Logical Device for each board. After a Logical Device is created, you can reconfigure it with any options available for the specific hardware model.

When you click the *Configure* button in the *DriverLINX Configuration Panel*, DriverLINX displays the *Configure DriverLINX Device* dialog. The following sections describe your choices for configuring DriverLINX to work with a Keithley KPCI-ISO Series model.



## Device Subsystem Page



Use the Device Subsystem page to tell DriverLINX the Model and Board Id of your KPCI-ISO Series board. This page also shows the Vendor and Device number and provides the *Special...* button to access the Walking-Bit Test dialog.

### ***Vendor***

The *Vendor* property displays “Keithley Instruments, Inc.” It is a read-only property.

### ***Device***

The *Device* property designates the number, or “name,” of the Logical Device you are configuring. It is a read-only property. To change it, first save (**OK**) or quit (**Cancel**) the current configuration. Then rename the Logical Device using the *DriverLINX Configuration Panel*.

### ***Model***

The *Model* property selects the hardware model of the board you’re configuring. This driver supports models KPCI-PDISO-8A and KPCI-PIO32IOA.

### **Windows 98/Me, 2000/XP**

Windows 98/Me and 2000/XP automatically determine the model of your board so DriverLINX disables Model selection.

### **Windows NT**

Select your board’s model from the list.

### ***Board Id***

The Board Id property associates this Logical Device with a specific board. DriverLINX automatically enters the board’s serial number in this field. DriverLINX uses the board’s serial number to uniquely recognize boards if you have installed multiple boards of the same model into your computer.

### **Windows 98/Me, 2000/XP**

Windows 98/Me and 2000/XP automatically determine which board to associate with this Logical Device. DriverLINX enters the serial number of the board when it starts the configuration.

## Windows NT

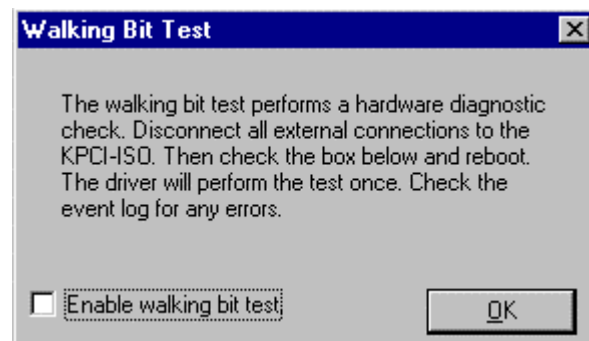
Under Windows NT, Board Id is initially blank. DriverLINX will use the Model to match this Logical Device to the first available board and then enter that board's serial number.

## Detect

The *Detect* property enables and disables DriverLINX's hardware detection and testing algorithms. For maximum system reliability, always leave this check box marked.

## Special...

The *Special...* button displays a hardware test dialog. DriverLINX can perform a Walking-Bit Test when it next loads this Logical Device. The Walking-Bit Test checks your board for internal input/output line damage.



To perform the Walking-Bit Test:

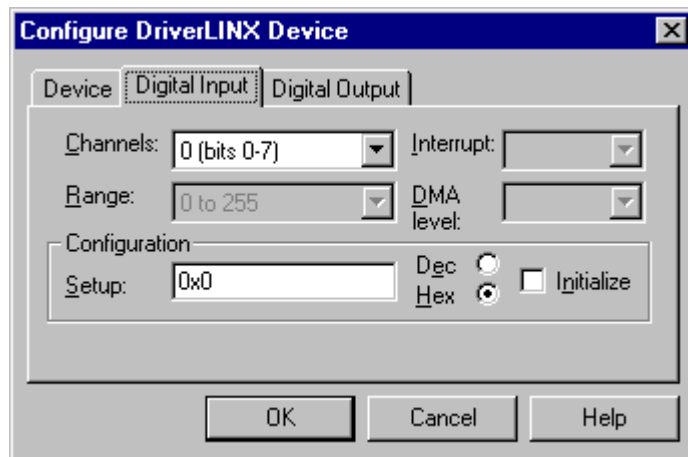
1. Enable the test using the check box.
  2. Make sure that Detect is checked on the Device Subsystem Page (see page 13).
  3. Remove the cable from your board.
  4. Restart your computer.
- If the driver finds a damaged line, it makes an entry in the Event Log and does not load the Logical Device.
  - If the test passes, the driver loads the Logical Device and clears the check box so the test does not run again.

---

**Important:** You must remove the cable from your board for a successful test.

---

## Digital Input Subsystem Page



Use the Digital Input Subsystem page to tell DriverLINX the Configuration Setup of the digital input channels. You can also use this page to view the Channel number and Range of each digital input channel.

By default, the hardware configures the digital input ports in unlatched mode. You can use this page to change the initial configuration of the input ports to latched mode.

### **Note:**

Applications can change a channel's configuration dynamically; See "Digital Port Configuration" on page 45.

### **Channels**

The Channels property allows you to select a Logical Channel for configuring or viewing the channel's range. Only the digital input/output channels support programmable configuration. The KPCI-ISO Series has External Clock and External Trigger channels that have fixed, input configurations. See "Digital Input Subsystem Signals" on page 39 for a list of channels for your board.

### **Range**

The Range property specifies the digital range for the selected Logical Channel. This is a read-only property.

### **Interrupt**

Windows automatically determines the interrupt channel, if any, for the KPCI-ISO Series board. DriverLINX disables this property.

### **DMA level**

The KPCI-ISO Series does not use system DMA channels. DriverLINX disables this property and displays it as blank.

## Configuration Setup

The *Configuration Setup* property allows you to statically configure all digital input ports as latched (strobed mode) or unlatched (basic mode).

- Unlatched (Basic mode)—Supports simple input/output without control or status signals. Outputs are latched and inputs are unlatched. Unlatched inputs return the state of the input lines at the time the software reads the digital input port.
- Latched (Strobed mode)—Supports input/output with an external control signal. Inputs and outputs are latched. The latched inputs return the state of the input lines at the time of a strobe signal on the INT\_REQ line. The default active edge, or polarity, of the strobe signal is the rising edge. Dynamic reconfiguration using a Dio Setup event can change the active edge to either rising or falling. (See “Dynamically Configuring KPCI-ISO Digital Input Ports” on page 45.)

DriverLINX supports two methods for statically configuring a digital I/O port.

- The simplified method configures all the digital I/O lines of a port for input or output in basic mode.
- The advanced method allows you to configure groups of digital I/O lines for input or output in either basic or strobed mode.

## Mode Configuration

DriverLINX mode configuration mimics the control register of the Intel 8255 digital I/O chip because most digital I/O hardware supports at least a subset of the 8255's modes. The KPCI-ISO Series supports latched or unlatched digital inputs. On model KPCI-PIO32IOA, both digital input ports operate in the same latching mode. The KPCI-ISO Series supports two mode settings: basic and strobed. Unlike the Intel 8255, the KPCI-ISO mode setting applies to all input channels. In strobed mode, the KPCI-ISO Series supports selectable latching on either the rising or falling edge on INT\_REQ.

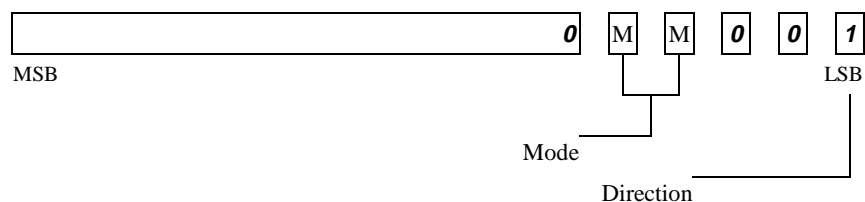
### Notes:

- Only channel 0 accepts a mode setup.
- On model KPCI-PIO32IOA, both input channels operate in the same mode.

To configure the mode settings of a device:

1. Select Channel 0 in the drop-down channel list.
2. Check the *Initialize* box.
3. Enter a *Setup* value in the following format:

### KPCI-ISO Series Mode Configuration





- **Mode Field**—

- 00 = Basic; no latching (emulated Intel 8255 Mode 0)

Basic mode supports simple input/output without control or status signals. Outputs are latched and inputs are unlatched. Unlatched inputs return the state of the input lines at the time the software reads them.

- 01 = Latching on rising-edge strobe (emulated Intel 8255 Mode 1)

Strobed mode supports input/output with an external rising-edge input-latching signal. Inputs and outputs are latched. The latched inputs return the state of the input lines at the time of a strobe signal on the INT\_REQ line.

- 11 = Latching on falling-edge strobe (modified Intel 8255 Mode 1)

Strobed mode supports input/output with an external falling-edge input-latching signal. Inputs and outputs are latched. The latched inputs return the state of the input lines at the time of a strobe signal on the INT\_REQ line.

- **Direction Field**—

- 1—configures the port for input

### ***Initialize***

Checking the *Initialize* check box instructs DriverLINX to use the *Configuration Setup* property to configure the digital I/O ports. Check *Initialize* to put the configuration setup into effect.

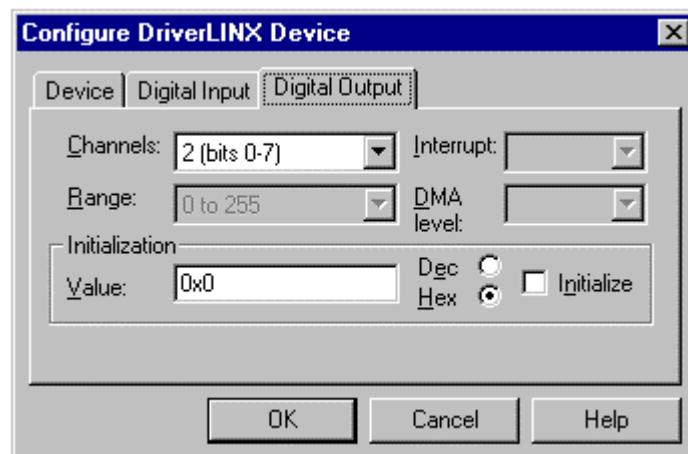
### ***Dec***

This check box converts the *Configuration Setup* property to decimal.

### ***Hex***

This check box converts the *Configuration Setup* property to hexadecimal.

## **Digital Output Subsystem Page**



Use the Digital Output subsystem page to change the default digital output port initialization values.

### ***Channels***

The *Channels* property allows you to select a Logical Channel for initialization or viewing the channel's range. KPCI-ISO Series models have either one or two digital output channels. The Digital Output subsystem shares these channels with the Digital Input subsystem, where they serve as read-backs of the last output value.

### ***Range***

The *Range* property specifies the supported digital output range for the selected Logical Channel. This is a read-only property.

### ***Interrupt***

Windows automatically determines the interrupt channel for the KPCI-ISO Series board. DriverLINX disables this property.

### ***DMA level***

The KPCI-ISO Series does not use system DMA channels. DriverLINX disables this property and displays it as blank.

### ***Initialization Value***

The *Initialization Value* property specifies the digital output value DriverLINX will write to the selected Logical Channel on hardware initialization. DriverLINX only writes this value if you enable the *Initialize* check box. By default, DriverLINX uses the hardware-defined initialization values if the *Initialize* check box is not checked. For the KPCI-ISO Series, the default digital output value is zero.

### ***Initialize***

Checking the *Initialize* check box instructs DriverLINX to use the *Initialization Value* property, rather than the default value, for digital output port initialization.

### ***Dec***

This check box converts the *Initialization Value* property to decimal.

### ***Hex***

This check box converts the *Initialization Value* property to hexadecimal.

# Programming the KPCI-ISO Series

---

## Selecting an API

The KPCI-ISO Series supports three different device driver interfaces to best match your programming needs. The supported driver interfaces are:

- **DriverLINX**—An interface that is hardware and operating system independent and supports multitasking, multithreading applications.
- **Hardware Direct I/O**—An interface for Windows 95/98 (or Windows NT/2K/XP with a special direct I/O driver) that supports single-tasking, single-threaded access directly to the hardware registers.
- **ActiveX Direct I/O**—An interface that is operating system independent and supports single-tasking, single-threaded access to an Intel 8255-like function call interface.

Keithley recommends the using the DriverLINX interface for maximum portability and versatility, but the other interfaces are useful in special circumstances. Please review the following sections on the advantages and disadvantages of each interface.

## Choosing DriverLINX

Keithley highly recommends selecting the DriverLINX interface for your applications. The DriverLINX interface has the following advantages:

- **Hardware independence**—DriverLINX supports ISA, PCMCIA, and PCI digital I/O boards with a common interface as well as digital ports on analog I/O and counter/timer boards.
- **Operating system independence**—DriverLINX supports Windows 98/Me and Windows NT/2000/XP with a common interface.
- **Shared hardware access**—DriverLINX allows multiple processes or threads to cooperatively share hardware resources.
- **Multitasking, multithreading support**—DriverLINX provides the synchronization and coordination for multitasking, multithreading applications to safely access shared hardware resources.

- **Portability**—DriverLINX supports older and newer hardware with a common interface for 32-bit applications on Windows 98/Me and Windows NT.
- **Versatility**—DriverLINX supports a wide variety of programming styles and languages including C/C++, Visual Basic, Delphi, and LabVIEW.
- **Background I/O**—DriverLINX supports background I/O using interrupts or other techniques so your application can overlap processing and data acquisition.
- **Full hardware support**—DriverLINX supports *all* the hardware's features.

The DriverLINX interface has a few disadvantages compared to the alternative interfaces:

- **Software modifications**—Existing applications will require rewriting to use DriverLINX.
- **No direct hardware access**—Before Windows NT, Microsoft discouraged, but nevertheless allowed, direct application program access to hardware.
- **Speed of single-value I/O**—Applications, especially those ported from DOS, that depend on rapid software polling of I/O ports will notice that the operating system time cost for multitasking, multithreading synchronization is higher than direct hardware access.

While writing your application to use a multi-product, portable API is the best long-term solution, you should consider the alternative APIs *only* for special purpose or short-term needs.

## Choosing Hardware Direct I/O

The Direct Hardware I/O Interface allow access to the KPCI-ISO hardware registers. The registers on these boards mimic the layout of Intel 8255-based products, such as the PIO24 and PIO96. This interface has the following advantages:

- **Direct I/O support**—This interface supports existing applications that write directly to the digital I/O's hardware registers using the x86 CPU's **inb** and **outb** instructions.
- **Intel 8255 emulation**—This interface accesses the hardware's I/O address map, which implement a subset of the programming protocols of Intel's 8255 chip.
- **Fast hardware access**—This interface provides the fastest possible access to the hardware registers.
- **MS-DOS window support**—This interface will host 16- and 32-bit console applications running in an MS-DOS window.

The Hardware Direct I/O Interface has a few, significant disadvantages:

- **Requires a special driver under Windows NT/2000/XP**—Only Windows 98/Me allows direct access to I/O ports. Windows NT/2000/XP does not support direct I/O without using a special port I/O driver available from Keithley's Web site at [www.keithley.com](http://www.keithley.com).

- **Non-exclusive hardware access**—This interface does not synchronize or coordinate hardware access among threads or processes.
- **Incompatible with DriverLINX**—Applications cannot use this interface with DriverLINX either in the same or another application.
- **Supports subset of hardware features**—This interface does not support using interrupts.

Keithley does not recommend using this interface for developing new applications.

## Choosing ActiveX Direct I/O

The ActiveX Direct I/O interface uses the methods of an ActiveX Automation object to access the KPCI-ISO hardware. This interface has the following advantages:

- **Operating system independence**—Both Windows 98/Me and Windows NT/2000/XP support this interface.
- **Intel 8255 emulation**—This interface simulates the I/O address map and programming protocols of Intel's 8255 chip.
- **Fast hardware access**—This interface provides faster access to the hardware registers than is available with the DriverLINX interface.
- **Versatility**—Most Windows compilers and scripting languages support ActiveX Automation objects.
- **Speed of single-value I/O**—The I/O performance of this interface is almost as fast as the Hardware Direct I/O interface when using early binding to the interface.
- **Board-relative addressing**—This interface automatically determines the actual hardware I/O address unlike the Hardware Direct I/O interface.
- **ISA digital I/O support**—This interface also supports Keithley's ISA digital I/O products.

The ActiveX Direct I/O interface also has disadvantages:

- **Non-exclusive hardware access**—This interface does not synchronize or coordinate hardware access among threads or processes.
- **Incompatible with DriverLINX**—Applications cannot use this interface with DriverLINX either in the same or another application.
- **Supports subset of hardware features**—This interface does not support using interrupts.
- **ActiveX interface**—Using Automation objects in some C++ compilers is more difficult than in Visual Basic.

---

## Installing an API

To install a KPCI-ISO programming interface, select one of the following topics for more information.

## Installing DriverLINX

The normal installation process automatically installs both the DriverLINX and Hardware Direct I/O APIs. Follow the instructions in the DriverLINX CD installer.

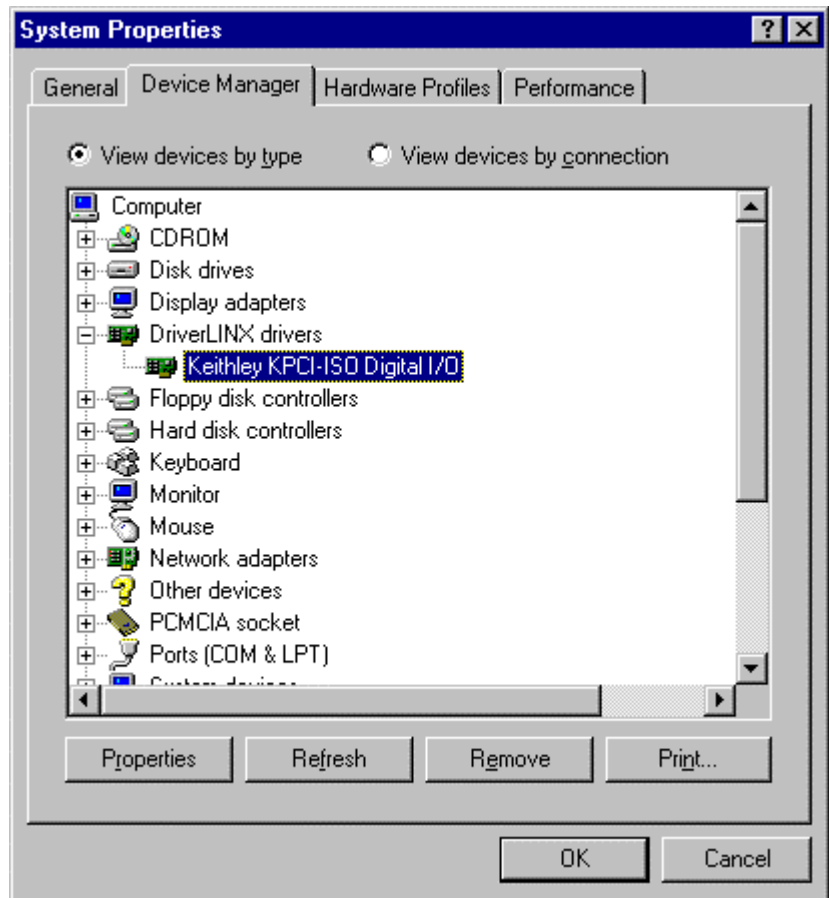
## Installing the Hardware Direct I/O Interface

To use the Hardware Direct I/O interface, you do not need to install a product-specific driver, but you must determine the I/O address that the operating system assigned to the KPCI-ISO board. Also, under Windows NT/2000/XP, you must install a separate port I/O driver from [www.keithley.com](http://www.keithley.com) that allows access to I/O ports.

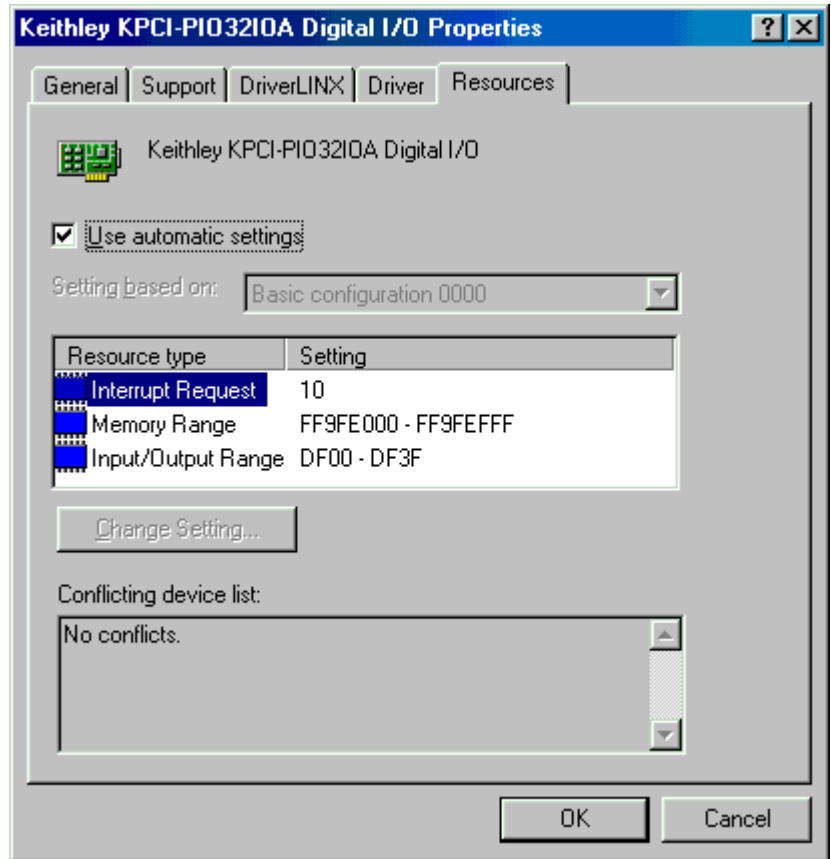
### *Determining the KPCI-ISO Board's I/O Address*

Windows automatically selects an available I/O address for the board. If you wish to check this address:

1. From Windows Control Panel, click the System icon and then select the Device Manager tab.
2. Click on the “+” next to DriverLINX drivers to expand the list of installed drivers.
3. Click on the driver you want to inspect and then click Properties.



4. Select the Resources tab to see the current I/O address assignment.



5. Note the 64-byte (40 hexadecimal) Input/Output Range. The base address for the data and control registers is the beginning of this range. In the example above, the base address is DF00.
6. Close the Device Manager windows.

## Installing the ActiveX Direct I/O Interface

The normal installation process automatically installs both the DriverLINX and Direct I/O APIs. Follow the instructions in the DriverLINX CD installer.

### *Installing the ActiveX Direct I/O Driver*

The normal DriverLINX installation automatically installs the ActiveX Direct I/O Driver. This driver is a registered ActiveX object that resides in KDigIO.DLL.

This driver requires that you install the DriverLINX kernel drivers as it uses their services. Follow the normal DriverLINX installation and configuration procedures to install these kernel drivers.

### *Configuring the ActiveX Direct I/O Driver*

The ActiveX Direct I/O Driver requires that you configure a DriverLINX Logical Device using the DriverLINX Configuration Panel

utility. The ActiveX Direct I/O driver does not require any separate configuration.

---

## Programming an API

To use a KPCI-ISO Series programming interface, select one of the following topics for more information.

### Programming DriverLINX

The DriverLINX API is extensively documented in this manual and in the *Digital I/O Programming Guide* and the *DriverLINX or DriverLINX/VB Technical Reference Manuals*.

### Programming the Hardware Direct I/O Interface

This interface uses only the x86 CPU instructions, **inb** and **outb**, for 8-bit I/O. See “Determining the KPCI-ISO Board’s I/O Address” on page 22 to determine the I/O address range for your KPCI-ISO board.

For more information on register-level programming for the KPCI-ISO, see the *KPCI-ISO PCI Digital I/O Board User’s Manual*.

Keithley does not recommend this interface for new applications.

### Programming the ActiveX Direct I/O Interface

The ActiveX Direct I/O Interface is an ActiveX Automation object hosting one interface. This control is compatible with Microsoft Visual C++ and Visual Basic as well as other ActiveX-hosting languages.

The methods of the control’s interface, IKDigitalIo, emulate the 8-bit port I/O behavior of an Intel 8255 chip. Consult an Intel hardware manual or the programming section of the *KPCI-ISO PCI Digital I/O Board User’s Manual*.

#### ***IKDigitalIo Intel 8255 Interface***

The following syntax descriptions are shown in C/C++. For Visual Basic, use the Object Browser to see the VB syntax. Also see “Using the ActiveX Direct I/O Driver in Visual Basic” on page 25.

- HRESULT OpenDevice(BSTR Driver, long Device);  
  
Opens a digital I/O device.  
Driver is “KPCISIO” to open a KPCI-ISO Series board.  
Device is the DriverLINX Logical Device number of the KPCI-ISO Series board to open.
- HRESULT CloseDevice();  
  
Closes a previously opened device.  
The number of CloseDevice calls must match the number of OpenDevice calls.
- HRESULT Read(short Offset, unsigned char \* Result);



Reads an 8-bit value from the board.  
Offset is the value of the register to read relative the base I/O address.  
Result is the value read from the register.

- HRESULT Write(short Offset, unsigned char Value);

Writes an 8-bit value to the board.  
Offset is the value of the register to write relative the base I/O address.  
Value is the output to write to the register.

- HRESULT OpenCount(long \*Count);

Returns the number of times a client has called OpenDevice for this Logical Device.

Count is the address of the value for the result.

- HRESULT Device(long \*LogicalDevice);

Returns the Logical Device number if the device is open. Otherwise it returns -1.

LogicalDevice is the address of the value for the result.

- HRESULT Driver(BSTR \*Name);

Returns the DriverLINX class driver name if any device is open.

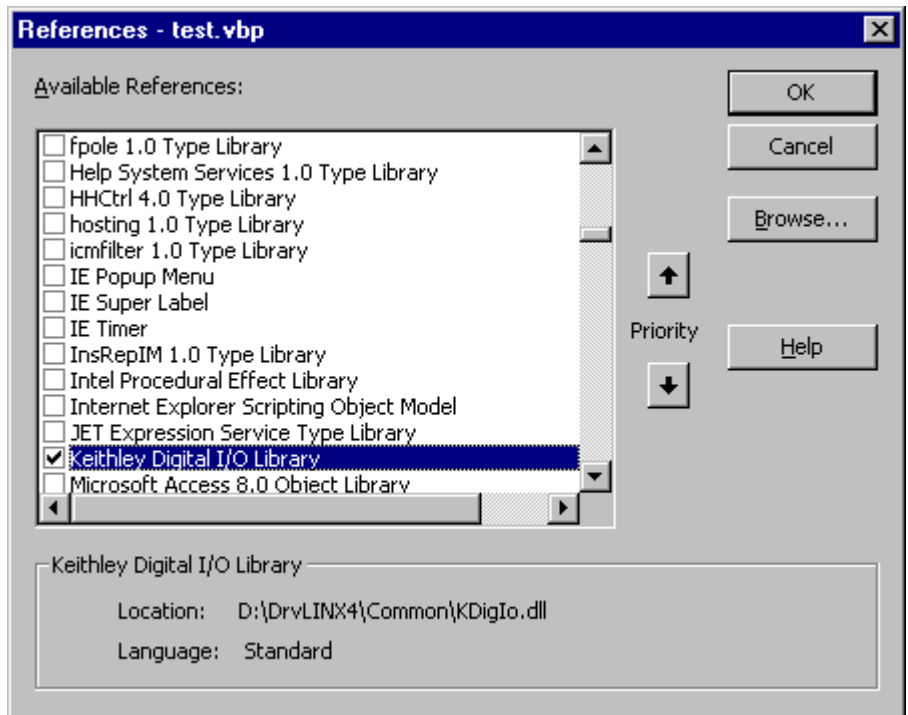
Otherwise it returns an empty string.

Name is the address of a string value for the result.

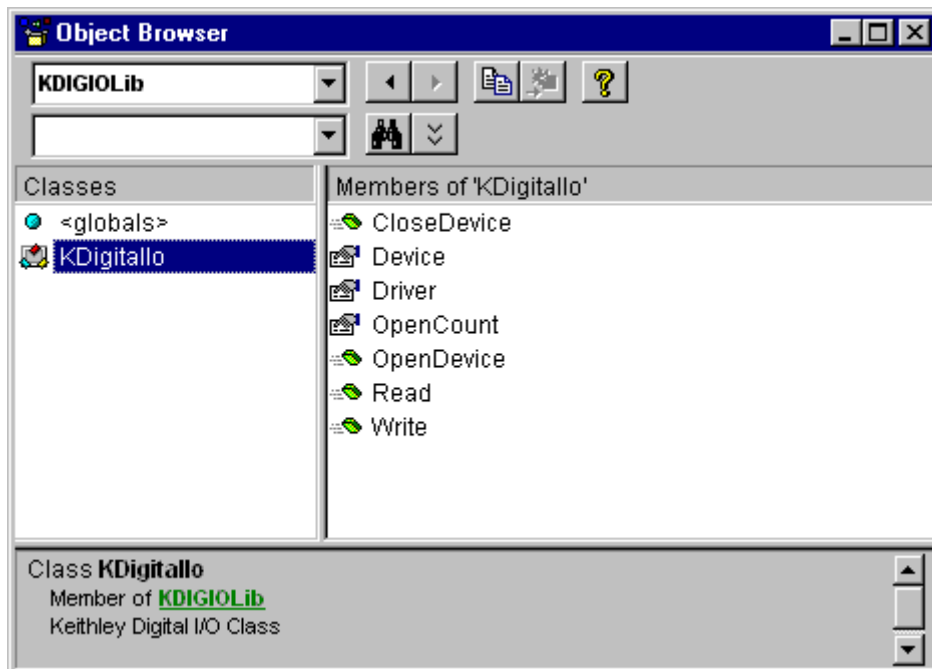
### ***Using the ActiveX Direct I/O Driver in Visual Basic***

The following instructions are for Version 5 of Microsoft Visual Basic. If you have a different version, please consult the Visual Basic documentation.

1. From VB's "Project" menu, select "References..."
2. Scroll through the "Available References", looking for "Keithley Digital I/O Library".

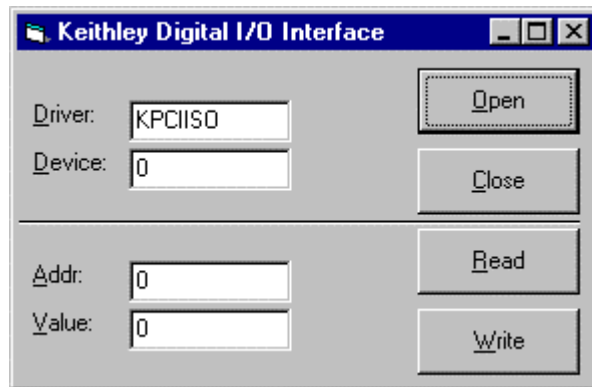


3. Check “Keithley Digital I/O Library” and then click OK.
4. From VB’s “View” menu, select “Object Browser” to display a list of libraries.
5. Search for “KDIGIOLib”.



6. The Object Browser shows the VB syntax for the IKDigitalIo (see “IKDigitalIo Intel 8255 Interface” on page 24) class.

## Creating a Simple Visual Basic Application



1. Create a Visual Basic form with the controls show above.
2. Add the following source code to declare, open, close, read, and write a device.

```
KPIO New KDigitalIo

cmdClose_Click()
    KPIO.CloseDevice

cmdOpen_Click()
    KPIO.OpenDevice txtDriver, txtDevice

cmdRead_Click()
    txtValue = Hex(KPIO.Read(txtAddr))

cmdWrite_Click()
    KPIO.Write txtAddr, txtValue
```



# Accessing KPCI-ISO Series Features using DriverLINX

---

## Introduction

DriverLINX accesses the hardware features of many different types of data-acquisition boards using a board-independent hardware model. The KPCI-ISO Subsystem chapters, which follow, explain the implementation of DriverLINX's hardware model for the KPCI-ISO Series. This chapter is for engineers who are more familiar with the board's features than DriverLINX's model. It surveys the Keithley KPCI-ISO hardware features to show how they fit into DriverLINX's hardware model.

This chapter also discusses the differences between the KPCI-ISO Series boards and the ISA boards they may replace. Direct access techniques—important to non-DriverLINX applications—differ between the PCI bus series and the ISA bus series. Consideration of this issue may be important when attempting to port applications between boards.

# KPCI-ISO Series Hardware Features

This section discusses the following hardware features of the KPCI-ISO Series, referencing related topics in this manual and other DriverLINX documentation:

- Data Lines
- Control Lines on the KPCI-ISO Series
- Input Latching on the KPCI-ISO Series

## Data Lines

Feature	Description	Reference/Notes
IP0-7 OP0-7 IP8-15 OP8-15	The KPCI-ISO Series models have one or two input port and one or two output ports. DriverLINX maps these ports as Logical Channels in the Digital Input and Digital Output Subsystems.	“Connecting Signals to the KPCI-” on page 39.
	DriverLINX also allows access to groups of 1, 2, 4, 8, 16 or 32 lines in single or consecutive ports.	“Extended Logical Channel Addressing” on page 57.

## Control Lines on the KPCI-ISO Series

Feature	Description	Reference/Notes
INT_REQ	Interrupt Request—the INT_REQ signal generates an interrupt on either the rising edge or falling edge. DriverLINX can use this interrupt for timing or as a start or stop trigger.	DriverLINX can use this interrupt for timing or as a start or stop trigger: <ul style="list-style-type: none"><li>• Timing signal —“Digital Input Timing Events” on page 46.</li><li>• Start trigger —“Digital Input Start Events” on page 49.</li><li>• Stop trigger—“Digital Input Stop Events” on page 50.</li></ul>
INT_ENN	Interrupt Enable—the INT_ENN signal is an active-low input that enables the INT_REQ line.	Because software cannot sense or control the INT_ENN signal, it has no representation in DriverLINX’s hardware model.

## Input Latching on the KPCI-ISO

Feature	Description	Reference/Notes
INT LATCH	The KPCI-ISO can optionally latch the input buffers when it receives an INT_REQ signal.	
	DriverLINX enables input buffer latching depending on the Service Request mode and clock source, and the configuration of the board's input ports:	Set the task's Mode property in the Service Request: <ul style="list-style-type: none"> <li>ActiveX Control—"Service Request Control - Request Group" in the <i>DriverLINX/VB Technical Reference Manual</i>.</li> <li>C/C++—"Service Requests - Request Group" in the <i>DriverLINX Technical Reference Manual</i>.</li> </ul>
	<ul style="list-style-type: none"> <li>Polled Mode Port configuration: <ul style="list-style-type: none"> <li>Basic mode (Mode 0)—<b>unlatched input</b></li> <li>Strobed or Strobed_NE mode (Mode 1)—<b>latched input</b></li> </ul> </li> </ul>	Configure the board using one of the following methods: <ul style="list-style-type: none"> <li>Static configuration using the DriverLINX Configuration Panel— "Configuring the " on page 11.</li> <li>Dynamic configuration using a <i>Configure Service Request</i>—"Configuring a Digital Port at Run Time" in the <i>DriverLINX Digital I/O Programming Guide</i>.</li> </ul>
	<ul style="list-style-type: none"> <li>Interrupt Mode Clock source: <ul style="list-style-type: none"> <li>INTERNAL (system timer)—<b>unlatched mode</b></li> <li>EXTERNAL, EXTERNAL+, EXTERNAL- —<b>latched mode</b></li> </ul> </li> </ul>	Set the Rate Event's clock source property in the Service Request: <ul style="list-style-type: none"> <li>ActiveX Control—"Service Request Control - Events Group - Rate Events" in the <i>DriverLINX/VB Technical Reference Manual</i>.</li> <li>C/C++—"Service Requests - Events Group - Rate Events" in the <i>DriverLINX Technical Reference Manual</i>.</li> </ul>

## Differences Between the PCI and ISA Models

The KPCI-ISO Series models are related to two models from the ISA PIO Series. This section lists the differences between the two series:

- INT\_REQ on the KPCI-ISO Series
- Bus Interface
- Intel 8255 Mode Emulation

### INT\_REQ on the KPCI-ISO Series

Feature	Description	Reference/Notes
INT POLARITY	The KPCI-ISO Series supports rising or falling edge interrupts on its INT_REQ line. The ISA	On the KPCI-ISO Series, the rising edge is the default active edge. Applications can change the

	models do not support interrupts.	interrupt polarity using a Dio Setup event or by choosing between External+ and External- clock source in a timing event.
INT LATCH	The KPCI-ISO can optionally latch the input buffers when it receives an INT_REQ signal. The ISA PDISO-8 and PIO-32IO cannot latch the input buffers.	This added feature on the KPCI-ISO allows it to perform strobed input. See Intel 8255 Mode Emulation—Mode 1, below.



## Bus Interface

Feature	Description	Reference/Notes
PCI ISA	Keithley's KPCI-ISO models have a PCI bus interface, while the PIO models have an ISA bus interface.  On the PCI bus, boards can share interrupts. On the ISA bus, boards and applications usually cannot share interrupts.	DriverLINX applications access boards through a hardware-independent interface so they are portable between boards on different buses.  Applications that access boards directly are not portable.

## Intel 8255 Mode Emulation

Feature	Description	Reference/Notes
Mode 0	Basic I/O—All models in both series support Mode 0.	DriverLINX sets the mode through static or dynamic configuration.  Static configuration using the DriverLINX Configuration Panel— “Configuring the ” on page 11.
Mode 1 rising or falling edge	Strobed I/O—The KPCI-ISO Series emulates strobed mode through input buffer latching with its INT_REQ line. The ISA PDISO-8 and PIO-32IO do not support this mode.	
Mode 2	Bidirectional I/O—Only the ISA PIO96 allows this mode.	



# Using DriverLINX with the KPCI-ISO Series

---

## Introduction

This chapter describes how to set up and use KPCI-ISO hardware features with DriverLINX. The descriptions here use the *Edit Service Request* dialogs for language and API independence.

Other DriverLINX documentation can help you with language specifics and general DriverLINX programming:

- For the correct syntax with the language you're using, please see the *DriverLINX Technical Reference Manual* or the *DriverLINX/VB Technical Reference Manual*.
- For DriverLINX examples in your programming language, please see the source code examples in the subdirectories of your DriverLINX installation directory or on the original Distribution Media.
- For a detailed reference on digital input/output programming, please see the *Digital I/O Programming Guide*.

---

## DriverLINX Hardware Model for the KPCI-ISO Series

By design, DriverLINX provides a portable, hardware-independent API for data-acquisition boards while still allowing applications to access unique or proprietary hardware features of specific products. To achieve this goal, DriverLINX maps a hardware-independent, or abstract, data-acquisition model onto KPCI-ISO Series hardware capabilities.

The following sections describe how DriverLINX implements KPCI-ISO Series hardware features as Subsystems, Modes, Operations, Events, Logical Channels, Buffers, and Messages.

### DriverLINX Subsystems

The KPCI-ISO Series supports four of DriverLINX's Logical Subsystems:

1. **Device**—refers to the KPCI-ISO board as a whole.

2. **Digital Input**—refers to the 8-bit digital input ports as well as the 1-bit digital input control signal, INT\_REQ, and read back of the digital output ports.
3. **Digital Output**—refers to the 8-bit digital output ports.
4. **Counter/Timer**—refers to the clock channel for pacing input/output tasks.

## DriverLINX Modes

Applications use modes in Service Requests to advise DriverLINX on their preferred hardware data transfer technique. The DriverLINX modes fall into two general classes:

- **Foreground or synchronous modes**—The calling application does not regain control until DriverLINX completes the Service Request. DriverLINX supports this mode for simple, single value or single scan I/O operations or software housekeeping functions that DriverLINX can complete without a significant delay.
- **Background or asynchronous modes**—The calling application regains control as soon as DriverLINX initiates the task. The calling application must synchronize with the data-acquisition task using status polling or DriverLINX’s messages (preferred). DriverLINX supports this mode for buffered data transfers or for commands that require a significant time to complete.

DriverLINX supports the following three modes with the KPCI-ISO for its commands (Service Requests).

- **Polled Mode**—This is a foreground or synchronous operation. DriverLINX supports this mode for simple, single-value or single scan I/O operations that the data-acquisition board can complete without significant delay.
- **Interrupt Mode**—This is a background or asynchronous operation. DriverLINX transfers data between the computer’s memory and the data-acquisition board using hardware interrupts and programmed I/O transfers.
- **Other Mode**—This is a foreground or synchronous operation. DriverLINX supports this mode for initialization, configuration, calibration, data conversion, and timebase operations.

The following table summarizes the data acquisition modes that DriverLINX supports for each subsystem with the Keithley KPCI-ISO.

Subsystem	Polled	Interrupt	DMA	Other
Digital Input	√	√		√
Digital Output	√	√		√
Counter/Timer				√
Device				√

*KPCI-ISO Supported DriverLINX Modes*

## DriverLINX Operations and Events

Applications construct DriverLINX data-acquisition tasks by combining a small number of DriverLINX operations and events in many possible ways. The following table summarizes the operations and events that DriverLINX supports for the Keithley KPCI-ISO. Later sections for each DriverLINX subsystem will describe the operations and events in more detail.

**Note:** All subsystems allow the *MESSAGE* operation, which is not shown in the table. DriverLINX allows any Mode setting for the *MESSAGE* operation.

Subsystem	Operation	Events		
		Timing	Start	Stop
Mode				
<b>Digital Input</b>				
Polled	Start	null	null, cmd, dig*	null, cmd, TC
Interrupt	Start, Stop, Status	dig**, rate	cmd, dig	cmd, TC, dig
Other	Initialize, Config	diosetup		
<b>Digital Output</b>				
Polled	Start	null	null, cmd, dig*	null, cmd, TC
Interrupt	Start, Stop, Status	dig, rate	cmd, dig	cmd, TC, dig
Other	Initialize			
<b>Counter/Timer</b>				
Other	Initialize			
<b>Device</b>				
Other	Initialize, Configure, Capabilities			

\* In polled mode, a digital start event sets up bit-masked input/output.

### *Allowed Operations and Events for KPCI-ISO Subsystems and Modes*

The following list explains the Event abbreviations in the preceding table:

- **null**—Null or None Event. Use when a Service Request does not require an event
- **cmd**—Command Event. Use when DriverLINX starts or stops a task on software command
- **TC**—Terminal Count Event. Use when DriverLINX processes all data buffers once

- **rate**—Rate Event. Use to specify how DriverLINX paces or clocks data transfer
- **diosetup**—DIO Setup Event. Use to specify to direction or mode of a digital input/output port.

## Logical Channels

DriverLINX designates the individually addressable hardware channels for each subsystem as “Logical Channels”. Generally, the zero-based Logical Channel numbering sequence closely follows the hardware channel numbering scheme.

In some cases, however, DriverLINX assigns Logical Channel numbers to hardware features that users do not commonly think of as “channels.” For instance, DriverLINX commonly models external hardware clock input lines, external hardware trigger input lines, and external interrupts as 1-bit digital Logical Channels. In other cases, DriverLINX models subsystem-specific features, such as internal pacer clocks, as members of a more general purpose set of counter/timer channels.

For more information about how DriverLINX assigns Logical Channel numbers, see the notes for each supported subsystem.

## Buffers

Applications usually use data buffers to exchange data between the application and the data-acquisition hardware. When using data buffers, observe the following about DriverLINX’s data buffers:

- DriverLINX supports data-acquisition tasks with 1 to 255 data buffers per task.
- DriverLINX imposes no size limits on a single buffer, although the operating system or some hardware products may have size restrictions.
- User applications must allow DriverLINX to allocate all data buffers to guarantee application portability to different hardware and operating systems and to ensure that the hardware can physically access the buffer memory.
- User applications usually do not have concurrent or immediate access to the in-use data buffer while DriverLINX is executing a data-acquisition task.

---

## Connecting Signals to the KPCI-ISO Series

The *KPCI-ISO PCI Bus Digital I/O Board User's Manual* describes the data and control signals for the KPCI-ISO Series and the connector pinouts for these signals (from the Windows Start menu, open *Programs/DriverLINX/Keithley/KPCI-ISO Hardware Manual* to access). This section summarizes how DriverLINX logically numbers the I/O data signals and how DriverLINX uses several of these control signals for external clock and trigger inputs.

### Digital Input Subsystem Signals

The KPCI-ISO Series Digital Input subsystem has one or two 8-bit digital input ports and one control input which DriverLINX models as a 1-bit logical digital input port. In addition, the series allow reading back the last value written to the digital output ports. DriverLINX maps these signals to Logical Channels as shown in the following tables.

#### KPCI-PDISO-8A

Logical Channel	DriverLINX Function	External Connector
0	Digital Input Port	IP0 – IP7
1	Digital Output Port Read Back	OP0 – OP7
2	External interrupt as logical trigger	INT_REQ
3	External interrupt as logical external clock	INT_REQ

#### KPCI-PIO32IOA

Logical Channel	DriverLINX Function	External Connector
0	Digital Input Port	IP0 – IP7
1	Digital Input Port	IP8 – IP15
2	Digital Output Port Read Back	OP0 – OP7
3	Digital Output Port Read Back	OP8 – OP15
4	External interrupt as logical trigger	INT_REQ
5	External interrupt as logical external clock	INT_REQ

### Digital Output Subsystem Signals

The Digital Output subsystem has three or twelve 8-bit digital input/output ports. DriverLINX maps these signals to Logical Channels as shown in the following tables:

#### KPCI-PDISO-8A

Logical Channel	DriverLINX Function	External Connector
-----------------	---------------------	--------------------

0	Digital Input Port placeholder (usage results in an error)	N/A
1	Digital Output Port Read Back	OP0 – OP7

#### KPCI-PIO32IOA

Logical Channel	DriverLINX Function	External Connector
0	Digital Input Port placeholder (usage results in an error)	N/A
1	Digital Input Port placeholder (usage results in an error)	N/A
2	Digital Output Port Read Back	OP0 – OP7
3	Digital Output Port Read Back	OP8 – OP15

## Counter/Timer Subsystem Signals

For the KPCI-ISO Series, DriverLINX supports a virtual timer channel to model using the interrupt request line, INT\_REQ, as an external clock. DriverLINX maps the timer to Logical Channel 0. The virtual timer channel provides program interchangeability with other DriverLINX drivers that support hardware with externally clocked physical timer channels.

Applications can select either rising-edge or falling-edge external clocking with the KPCI-ISO Series.

Logical Channel	Timer	Connector Name
0	Virtual timer	none

*How DriverLINX maps counter/timer hardware channels to Logical Channels*

Applications cannot use this timer independently from tasks on an input/output subsystem.



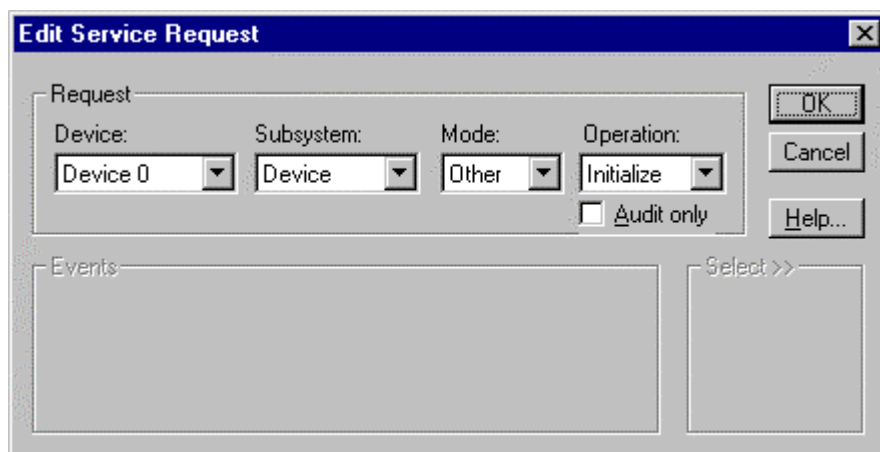
# Using the Device Subsystem

---

## Introduction

This chapter describes how DriverLINX implements Device Subsystem features for the KPCI-ISO Series. Each section covers a group of Service Request properties, using the *Edit Service Request* dialog as a reference. This chapter shows which DriverLINX features the KPCI-ISO driver implements and gives notes on their usage. For detailed explanations or examples of DriverLINX features, in general, see one of the following references:

- *DriverLINX/VB Technical Reference Manual (ActiveX Control)*
- *DriverLINX Technical Reference Manual (C/C++)*
- *DriverLINX Digital I/O Programming Guide*



A Service Request for the Device Subsystem uses the following properties:

Request Group:

- Modes—Specify how DriverLINX sets up data transfers and synchronizes execution with the application.
- Operations—Specify what service DriverLINX performs.

---

## Device Modes

The Service Request Mode specifies how DriverLINX sets up data transfers and synchronizes execution with the application.

The KPCI-ISO Device Subsystem supports the following mode:

- **Other**—For subsystem initialization and configuration.

---

## Device Operations

The Service Request Operation specifies what service DriverLINX performs.

The KPCI-ISO Device Subsystem supports the following DriverLINX operations:

*If another process is using the same data-acquisition board, DriverLINX will prevent Device Initialization from interfering with data-acquisition tasks of another process.*

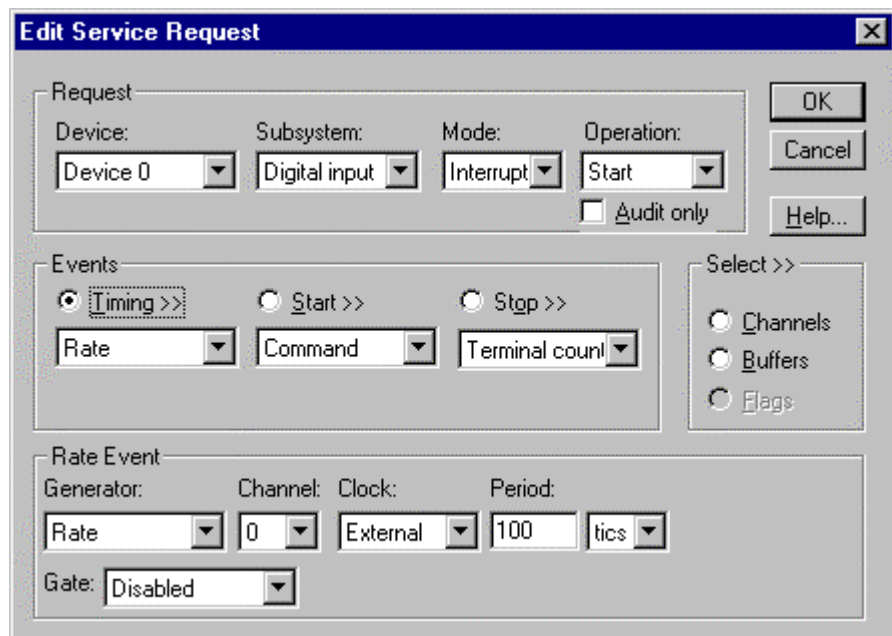
- **Initialize**—DriverLINX aborts all data-acquisition tasks for every subsystem controlled by the current process. DriverLINX then performs an initialization for each supported subsystem.
- **Configure**—DriverLINX displays the *Configure DriverLINX Device* dialog for the current Logical Device. Use the *DriverLINX Configuration Panel* rather than this operation to configure DriverLINX.
- **Capabilities**—DriverLINX provides hardware-specific and configuration information as a Logical Device Descriptor database.

# Using the Digital Input Subsystem

## Introduction

This chapter describes how DriverLINX implements Digital Input Subsystem features for the KPCI-ISO Series. Each section covers a group of Service Request properties or messages, using the *Edit Service Request* dialog as a reference. This chapter shows which DriverLINX features the KPCI-ISO driver implements and gives notes on their usage. For detailed explanations or examples of DriverLINX features, in general, see one of the following references:

- *DriverLINX/VB Technical Reference Manual (ActiveX Control)*
- *DriverLINX Technical Reference Manual (C/C++)*
- *DriverLINX Digital I/O Programming Guide*



A Service Request uses the following properties and messages:

Request Group:

- **Modes**—Specify how DriverLINX sets up data transfers and synchronizes execution with the application.
- **Operations**—Specify what service DriverLINX performs.

Events Group:

- **Timing Events**—Specify how DriverLINX and the board pace sampling.
- **Start Events**—Specify when sampling begins.
- **Stop Events**—Specify when sampling ends.

Select Group:

- **Channels**—Specify which channels, or ports, to sample.
- **Buffers**—Specify the size and number of sample storage buffers.

Service Request Messages:

- **Messages**—ActiveX control events or Windows messages that DriverLINX sends to synchronize execution with the application.

---

## Digital Input Modes

The Service Request Mode specifies how DriverLINX sets up data transfers and synchronizes execution with the application.

The KPCI-ISO Digital Input Subsystem supports the following modes:

- **Polled**—For single value or single scan digital input samples.
- **Interrupt**—For buffered transfers using programmed I/O.
- **Other**—For subsystem initialization and data conversion.

---

## Digital Input Operations

The Service Request Operation specifies what service DriverLINX performs.

The KPCI-ISO Digital Input Subsystem supports the following DriverLINX operations:

- **Initialize**—Aborts any active digital input tasks started by the application. The Initialize operation does not abort tasks started by other applications.
- **Configure**—Sets the mode and direction of digital input/output ports.
- **Start**—Initiates a data-acquisition the task specified in the Service Request.
- **Status**—Reports the buffer position of the next sample that DriverLINX will process.

- **Stop**—Terminates a digital input data-acquisition task.
- **Message**—Displays a dialog box containing the last Service Request error message, if any.

Most of these operations function similarly in all drivers. Only the Configure operation has parameters specific to the KPCI-ISO Series.

## Digital Port Configuration

The KPCI-ISO Series boards have one or two digital input ports. They can be configured as latched or unlatched using two methods:

- Users can statically configure the latching mode of the input ports using the *DriverLINX Configuration Panel*. (See “Configuring the ” on page 11.)
- The KPCI-ISO Series also supports dynamic configuration, enabling applications to configure the latching mode of the input ports with DIO Setup Events.

### *Dynamically Configuring KPCI-ISO Digital Input Ports*

When the driver loads a Logical Device, it configures the digital input/output ports with the settings in the DriverLINX Configuration Dialog. Applications can dynamically configure ports to any desired settings. The new settings persist until any application reconfigures a port. If only a single application is using the device, Device Initialization resets the hardware configuration back to the static settings defined during driver configuration.

Configuration specifies two settings for a digital input/output port:

- **Direction**—input or output.
- **Mode**—basic or strobed.

Basic mode supports simple input/output without control or status signals. Outputs are latched and inputs are unlatched. Unlatched inputs return the state of the input lines at the time the software reads them.

Strobed mode supports input/output with an external control signal. Inputs and outputs are latched. Latched inputs return the state of the input lines at the time of a strobe signal on the INT\_REQ line.

#### **Notes:**

- Strobed mode is available only on the KPCI-ISO.
- All inputs the KPCI-PIO32IOA operate in the same mode. Configuring any input in strobed configures all inputs in strobed mode.

To change a port’s configuration, setup the Service Request as follows:

Device	Subsystem	Mode	Operation
<Logical Device>	Digital Input or Digital Output	OTHER	CONFIGURE

- To configure as an input channel, set the subsystem to digital input. To configure as an output channel, set the subsystem to digital output.

Specify a Digital Setup Event for the timing event as follows:

Channel	Mode
<Logical Channel>	DIO_BASIC or DIO_STROBED or DIO_STROBED_NE

- Specify the channel you want to reconfigure as the Logical Channel in the Digital Setup Event.
- Specify one of the following Mode values in the Digital Setup Event:
  - DIO\_BASIC—Inputs unlatched, outputs latched.
  - DIO\_STROBED—Inputs latched by a positive (rising) edge on the INT\_REQ signal, output latched.
  - DIO\_STROBED\_NE—Inputs latched by a negative (falling) edge on the INT\_REQ signal, outputs latched.

See “Configuring a Digital Port at Run Time” in the *DriverLINX Digital I/O Programming Guide* for examples using a DIO Setup Event.

---

## Digital Input Timing Events

Timing Events specify how DriverLINX and the board pace, or clock, Digital Input sampling. DriverLINX uses the Timing Event to program when the KPCI-ISO reads the next digital input sample from the port.

The KPCI-ISO supports the following Timing Events:

- **None**—Input does not require pacing as DriverLINX is reading only a single value or simultaneous scan.
- **Rate**—The KPCI-ISO supports only fixed rate digital input using an external input signal (INT\_REQ).
- **Digital**—DriverLINX uses an external digital input signal (INT\_REQ) to pace the acquisition of each sample. Digital timing is available only on the KPCI-ISO.

### None or Null Event

The Null Event specifies that the task does not need a clock to determine when to read the next sample.

### Rate Event

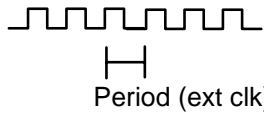
The KPCI-ISO supports one type of Rate Event for digital input:

- **Rate Generator**—Generates a rate clock with equal (internal clocking) or unknown (external clocking) time intervals between tics.



## Rate Generator: External Clocking

An externally clocked Rate Generator produces a rate clock with unknown time intervals between tics.



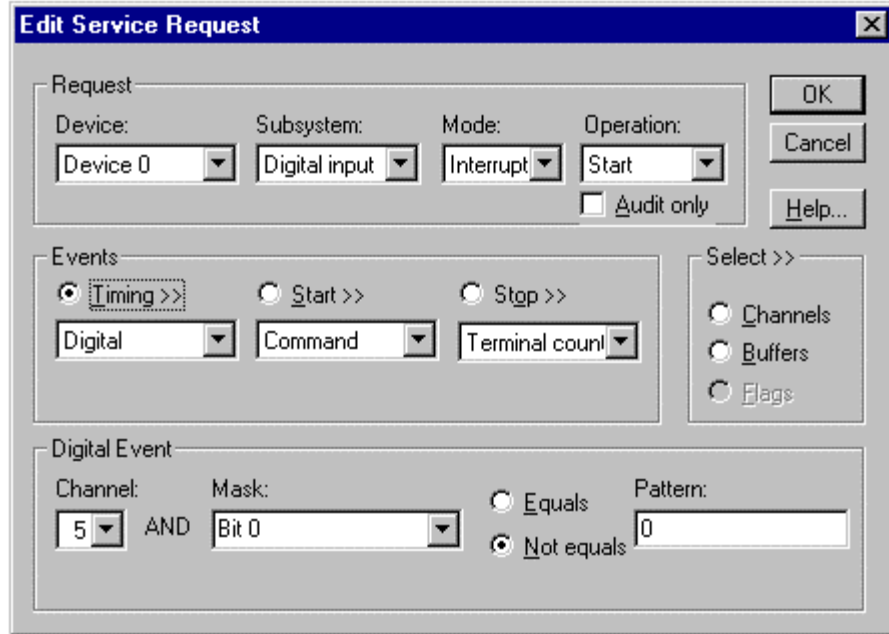
Use an externally clocked rate generator when you want to synchronize digital input samples with a recurrent external signal.

*How to set up the KPCI-ISO for fixed rate sampling using an external clock.*

- Specify external clocking using a **Rate Generator** on *Logical Channel 0* with an **External**, **External+** or **External-** Clock source. External and External+ both specify sampling on the rising, or positive, edge of the external clock signal. External- specifies sampling on the falling, or negative, edge of the external clock signal.
- Users should connect the external clock signal to the INT\_REQ line.
- The *Period* value does not affect the external clock frequency, but DriverLINX requires a valid hardware value in case the application requests a timebase operation and to optimize data transfer between the driver and the application.

## Digital Event

DriverLINX supports Digital Events as aliases for externally clocked Rate Generators. Use this technique for compatibility with data-acquisition products that only support external clock sources.



How to set up the KPCI-ISO for external rate sampling using a digital event.

Digital Timing Events contain *mask*, *pattern*, and *match* fields. The mask is logically ANDed with the digital input data on the Logical Channel and then compared against the *pattern* for a match/mismatch.

*A task cannot use a channel for a digital start or stop event while using it for a digital timing event.*

- Specify external clocking using *Logical Channel 3* (KPCI-PDISO8A) or **5** (KPCI-PIO32IOA). For hardware-independence, you can specify the hardware external clock channel by the symbolic constant, *DI\_EXTCLK*.
- Users should connect the external clock signal to the INT\_REQ line. Do not forget to ground the INT\_ENN line to enable the INT\_REQ line.
- Specify the *Mask* property with a single bit set to indicate that DriverLINX should only compare a 1-bit digital input value against the *Pattern* property.
- Specify the *Match* property as **Not equals**.
- Specify the *Pattern* property as **0** for a rising, or positive, edge clock ( $\neq 0$ ), or **1** for a falling, or negative, edge clock ( $\neq 1$ ).



---

# Digital Input Start Events

Start Events specify when the KPCI-ISO hardware starts reading digital input data.

The KPCI-ISO supports the following Start Events for digital input:

- **None**—Use this event when the DriverLINX operation does not require a Start Event.
- **Command**—DriverLINX starts the task on software command, i.e., as soon as DriverLINX finishes programming the KPCI-ISO hardware for the task.
- **Digital**—The KPCI-ISO starts acquiring digital input samples when a digital Logical Channel input satisfies the digital condition specified in the Start Event.

## None or Null Event

The Null Event specifies that the task does not need a Start Event to begin the task.

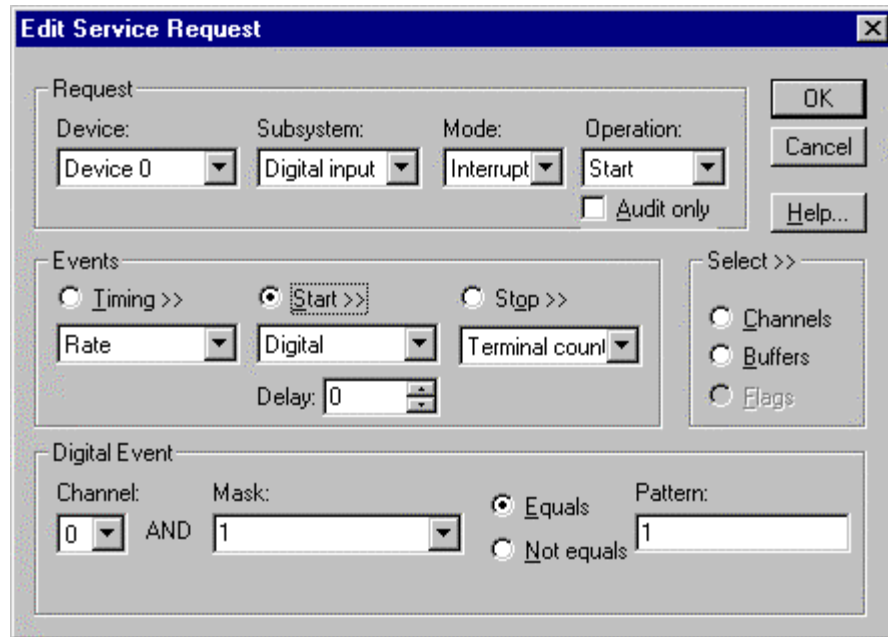
## Command Event

The Command Event starts data acquisition as soon as DriverLINX has completed programming the KPCI-ISO hardware with the task parameters.

## Post-Trigger Sampling with a Digital Event

The KPCI-ISO can acquire digital input samples *after* a digital trigger condition. Use post-triggering when you want to synchronize the start of data acquisition with an external signal.

DriverLINX supports digital triggering on all digital input lines. However, the KPCI-ISO has only one digital input, INT\_REQ, that can detect a trigger precisely when it occurs. For all other digital inputs, DriverLINX uses a software comparison to detect the trigger at the digital input port during the interrupt service routine. DriverLINX can reliably detect triggers only if either the triggers either have pulse durations longer than the interrupt latency (unlatched mode) or the application has previously configured the digital input trigger port as latched.



How to set up the KPCI-ISO for post-triggered digital input.

A Digital Start Event requires an external timing source because the KPCI-ISO Series does not have a hardware clock.

A task cannot use a channel for a digital start or stop event while using it for a digital timing event.

Digital Start Events contain *mask*, *pattern*, and *match* fields. The mask is logically AND with the digital input data on the Logical Channel and then compared against the *pattern* for a match/mismatch.

- Specify the *Logical Channel*.
- Specify the *Mask* property to indicate that DriverLINX should only compare selected bits in the digital input value against the *Pattern* property.
- Specify the *Match* property as **Equals** to trigger on an exact match of the digital pattern to the masked digital input. Match on **Not Equals** triggers on any difference between the masked digital input and the *Pattern*.
- Specify the *Pattern* property to compare with the masked digital input value.
- Specify the *Delay* property as the number of timing pulses (0 to  $2^{32} - 1$ ) to ignore after the trigger before starting data acquisition. The *Delay* property must be an integral multiple of the number of channels in the channel list.

## Digital Input Stop Events

Stop Events specify when the KPCI-ISO hardware stops reading digital input data.

The KPCI-ISO supports the following Stop Events for digital input:

- **None**—Use this event when the DriverLINX operation does not require a Stop Event.

- **Command**—DriverLINX stops the task on software command, i.e., when the application issues a Service Request with a *Stop* operation.
- **Terminal count**—DriverLINX stops the task after the KPCI-ISO hardware has filled all the data buffers once.
- **Digital**—The KPCI-ISO stops the task when the hardware detects that the digital Logical Channel input satisfies the digital condition specified in the Stop Event.

## None or Null Event

The Null Event specifies that the task does not need a Stop Event to end the task. This event may be used with single-value polled input tasks.

## Command Event

The Command Event stops data acquisition when the user application changes the *Operation* property in the Service Request to *Stop* and resubmits the Service Request to DriverLINX.

Sample timing depends on the task mode. In interrupt mode, DriverLINX samples at each timing event until the *Stop* operation. In polled mode, an application can repeatedly initiate a single scan with a *Status* operation before terminating the task with a *Stop* operation.

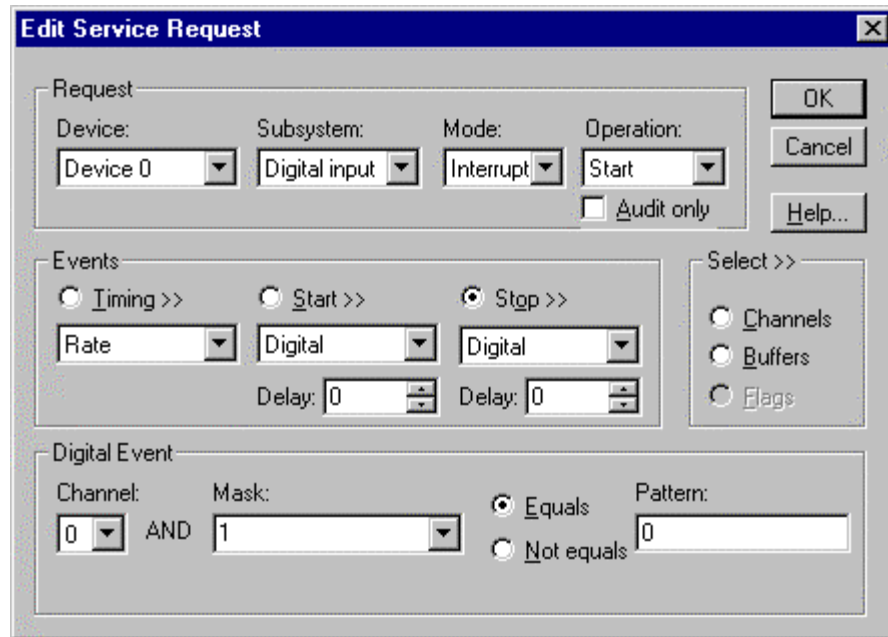
## Terminal Count Event

The Terminal Count Event stops data acquisition after DriverLINX has read the digital input data into all the data buffers *once*. Use terminal count when you want to read a fixed amount of data.

## Pre-Trigger Sampling with a Digital Event

The KPCI-ISO can acquire digital input samples *until* the hardware detects a digital trigger condition. Use pre-triggering when you want to synchronize the end of data acquisition with an external signal.

DriverLINX supports digital triggering on all digital input lines. However, the KPCI-ISO has only one digital input, INT\_REQ, that can detect a trigger precisely when it occurs. For all other digital inputs, DriverLINX uses a software comparison to detect the trigger at the digital input port during the interrupt service routine. DriverLINX can reliably detect triggers only if either the triggers either have pulse durations longer than the interrupt latency (unlatched mode) or the application has previously configured the digital input trigger port as latched.



How to set up the KPCI-ISO for pre-triggered digital input.

*A Digital Stop Event requires an external timing source because the KPCI-ISO does not have a hardware clock.*

*A task cannot use a channel for a digital start or stop event while using it for a digital timing event.*

Digital Stop Events contain *mask*, *pattern*, and *match* fields. The mask is logically AND with the digital input data on the Logical Channel and then compared against the *pattern* for a match/mismatch.

- Specify the *Logical Channel*.
- Specify the *Mask* property to indicate that DriverLINX should only compare selected bits in the digital input value against the *Pattern* property.
- Specify the *Match* property as **Equals** to trigger on an exact match of the digital pattern to the masked digital input. Match on **Not Equals** triggers on any difference between the masked digital input and the *Pattern*.
- Specify the *Pattern* property to compare with the masked digital input value.
- Specify the *Delay* property as the number of timing pulses (0 to  $2^{32} - 1$ ) to ignore after the trigger before stopping data acquisition. The *Delay* property must be an integral multiple of the number of channels in the channel list.

## Digital Input Channels

DriverLINX allows applications to specify the digital channels using three techniques:

- **Start Channel**—Acquire data using a single channel.
- **Start/Stop Channel Range**—Acquire data using a consecutive range of channels.

- **Channel List**—Acquire digital data using a list of channels.

In addition, DriverLINX provides two methods to address one or more digital I/O lines:

- **Extended Logical Channel Addressing**—Acquire data from 1, 2, 4, 8, 16 or 32 bits using a single Logical Channel.
- **Bit-Masked Addressing**—Read from or write data to specific bits of a digital port.

## Digital Input Logical Channels

The KPCI-ISO Series Digital Input subsystem has one or two 8-bit digital input ports and one control input which DriverLINX models as a 1-bit logical digital input port. In addition, the series allow reading back the last value written to the digital output ports. DriverLINX maps these signals to Logical Channels as shown in the following tables.

### KPCI-PDISO-8A

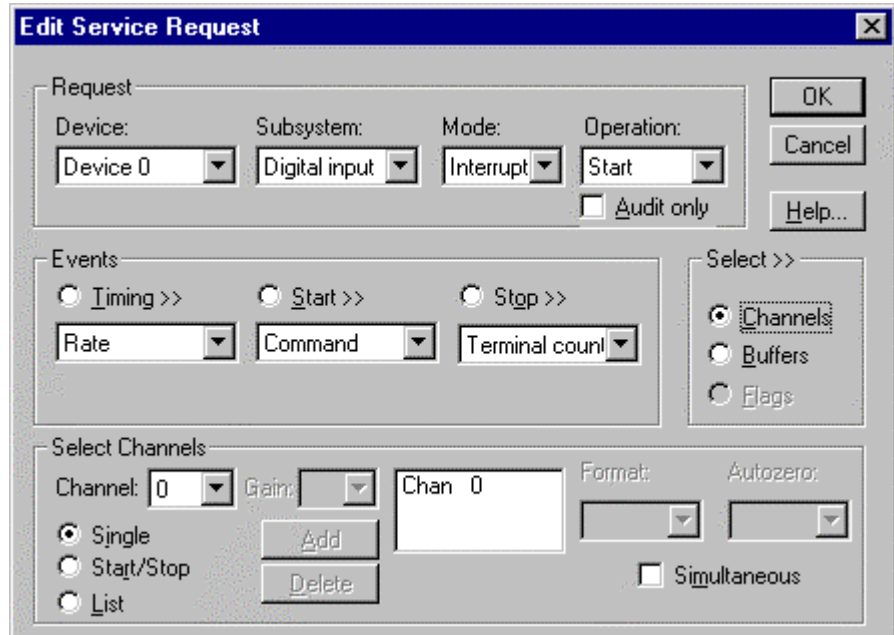
Logical Channel	DriverLINX Function	External Connector
0	Digital Input Port	IP0 – IP7
1	Digital Output Port Read Back	OP0 – OP7
2	External interrupt as logical trigger	INT_REQ
3	External interrupt as logical external clock	INT_REQ

### KPCI-PIO32IOA

Logical Channel	DriverLINX Function	External Connector
0	Digital Input Port	IP0 – IP7
1	Digital Input Port	IP8 – IP15
2	Digital Output Port Read Back	OP0 – OP7
3	Digital Output Port Read Back	OP8 – OP15
4	External interrupt as logical trigger	INT_REQ
5	External interrupt as logical external clock	INT_REQ

## Single Channel Digital Input

In Single Channel mode, a KPCI-ISO board acquires data from one channel.

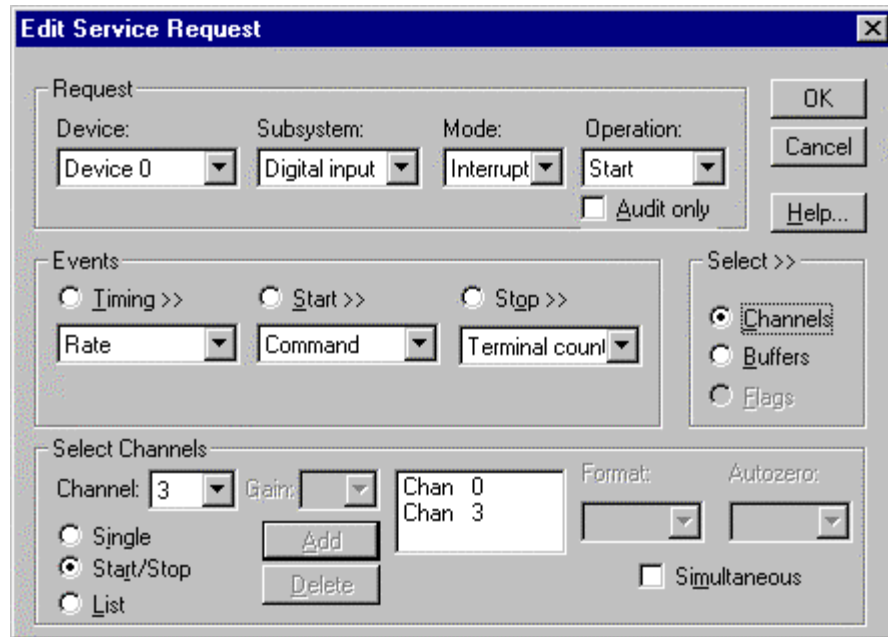


*How to set up the KPCI-ISO to read from a single channel*

## Multi-channel Digital Input Range

In Multi-channel Range mode, the KPCI-ISO acquires data from a consecutive range of digital channels.

- Specifying a Stop Channel less than the Start Channel instructs DriverLINX to use the following sequence:  
Start Channel  
...  
Logical Channel  $n-1$   
Logical Channel 0  
...  
Stop Channel  
where  $n$  is the number of Logical Channels in the subsystem.
- Specifying a Stop Channel equal to the Start Channel is the same as single channel mode.
- Specifying *Simultaneous*, instructs DriverLINX to sample all the selected channels at each timing event. Otherwise, DriverLINX acquires one sample from the next channel in sequence at each timing event.

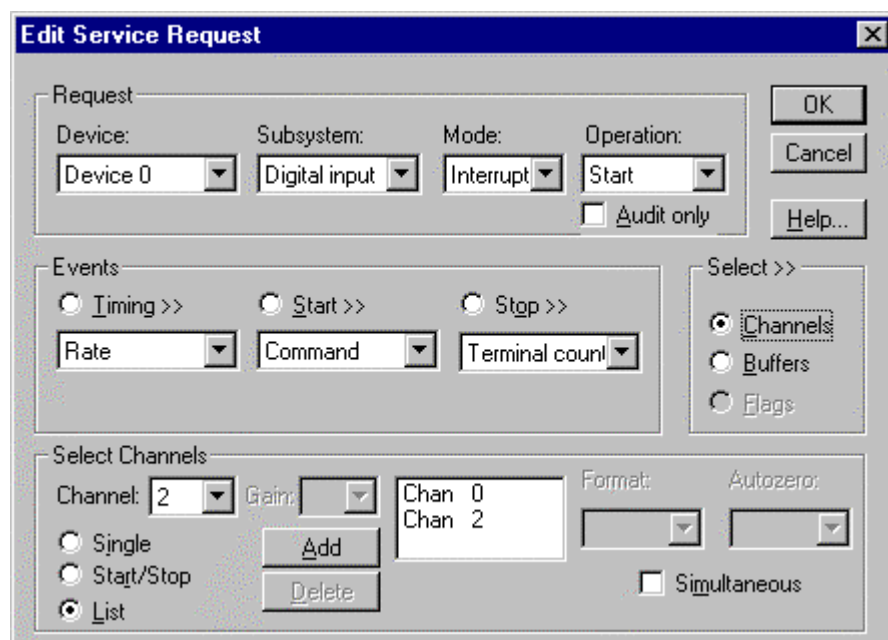


How to set up the KPCI-ISO to read from a range of channels

## Multi-channel Digital Input List

In Multi-channel List mode, the KPCI-ISO board acquires data from a random list of digital channels.

- Specifying *Simultaneous*, instructs DriverLINX to sample all the selected channels at each timing event. Otherwise, DriverLINX acquires one sample from the next channel in the list at each timing event.



How to set up the KPCI-ISO to read from a list of channels



## Extended Logical Channel Addressing

DriverLINX supports an extension to Logical Channel addressing allowing applications to combine adjacent Logical Channels into a single, wider channel or split a Logical Channel into smaller addressable parts. For instance, applications can address individual bits of a digital port or read and write multiple ports as a single value.

To use the Logical Channel addressing extensions, form a 16-bit Logical Channel address by combining the channel number of an addressable unit with a size field as follows:

<b>0</b>	<b>Size Code</b>	<b>Channel Number</b>													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSB															LSB

The following table lists the 3-bit size codes available for the KPCI-ISO Series:

Size Code	Unit	Bits
0	native	varies with hardware
1	bit	1
2	half nibble	2
3	nibble	4
4	byte	8
5	word	16

*"Native" units refer to the hardware-defined digital channel size. For the KPCI-ISO Series, the native size is an 8-bit byte.*

When using extended Logical Channel addressing, DriverLINX groups digital bits in units defined by the size code and then assigns consecutive channel numbers starting from zero.

For KPCI-ISO models, the Digital Input Channels have the following channel addresses for each size code:

### KPCI-PDISO-8A

Unit	Channels	Address (dec)	Address (hex)
native	0 .. 1	0 .. 1	0000 .. 0001
bit	0 .. 15	4096 .. 4111	1000 .. 100F
half nibble	0 .. 7	8192 .. 8199	2000 .. 2007
nibble	0 .. 3	12288 .. 12291	3000 .. 3003
byte	0 .. 1	16384 .. 16385	4000 .. 4001
word	0 .. 0	20480 .. 20480	5000 .. 5000

### KPCI-PIO32IOA

Unit	Channels	Address (dec)	Address (hex)
native	0 .. 3	0 .. 3	0000 .. 0003
bit	0 .. 31	4096 .. 4127	1000 .. 101F
half nibble	0 .. 15	8192 .. 8207	2000 .. 200F
nibble	0 .. 7	12288 .. 12295	3000 .. 3007
byte	0 .. 3	16384 .. 16387	4000 .. 4003
word	0 .. 1	20480 .. 20481	5000 .. 5001
dword	0 .. 0	24576 .. 24576	6000 .. 6000

**Notes:**

- The first half of each of the following ranges is for reading back values output to the second half of the range. Using the Logical Channels in the first half in the Digital Output Subsystem is an error.
- DriverLINX supports single-value transfers of any model-supported width.
- DriverLINX supports multi-value transfers only using data widths smaller than or equal to the hardware native size.

## Bit-Masked Addressing

In addition to Extended Logical Channel Addressing, DriverLINX also allows applications to access specific bits in a single channel. This technique is more convenient when an application needs to access bits that are not grouped together.

Bit masking allows an application or thread to access specific bits in a port without affecting bits controlled by other applications.

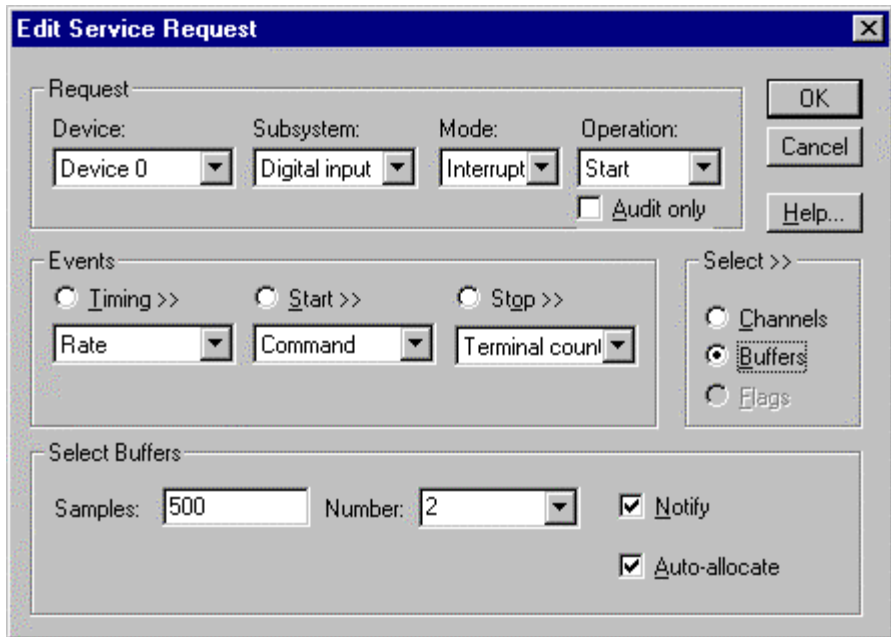
For more information, see “Reading or Writing Specific Digital Bits” in the *Digital I/O Programming Guide*.

---

## Digital Input Buffers

DriverLINX supports both single-value digital input and buffered digital input.

- **For single-value input**, specify the Number of buffers as **0** and the buffer size as **0**.
- **For buffered input**, specify the Number of buffers from **1** to **255** and the buffer size as needed.



*How to set up the KPCI-ISO to read digital samples using data buffers*

An individual DriverLINX buffer may have any size as long as the buffer length holds an integral number of channel scans, i.e., the number of digital input channels you're acquiring. This restriction enforces the requirement that a buffer has the same number of samples for each channel.

## Digital Input Messages

For digital input operations, DriverLINX can report the following messages to the application:

DriverLINX Message	Explanation
Service Start	DriverLINX has started the data acquisition task.
Service Done	DriverLINX has completed the data acquisition task.
Buffer Filled	DriverLINX has filled a data buffer with digital input samples.
Start Event	DriverLINX has processed the interrupt for a hardware start event. <i>Applicable only to the ExtTrg channel on the KPCI-ISO Series.</i>
Stop Event	DriverLINX has processed the interrupt for a hardware stop event. <i>Applicable only to the ExtTrg channel on the KPCI-ISO Series.</i>
Data Lost	DriverLINX has detected a digital input data overrun condition.
Critical Error	DriverLINX has encountered an unexpected hardware or software condition.

*DriverLINX Event message for digital input*

For details on using DriverLINX messages in your application see the appropriate reference manual:

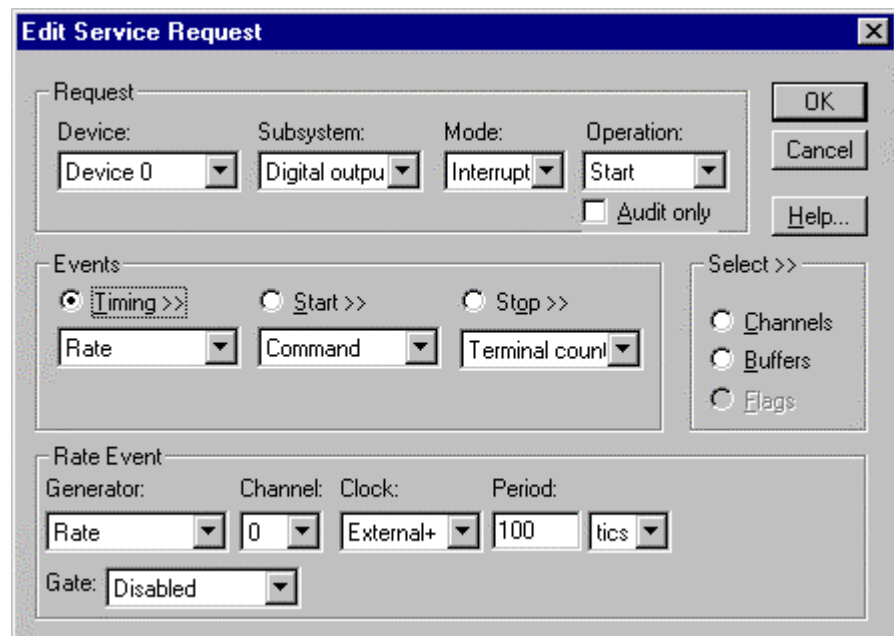
- ActiveX Control—“Overview of DriverLINX/VB Control Events” in the *DriverLINX/VB Technical Reference Manual*.
- C/C++—“DriverLINX Messages - Overview” in the *DriverLINX Technical Reference Manual*.

# Using the Digital Output Subsystem

## Introduction

This chapter describes how DriverLINX implements Digital Output Subsystem features for the KPCI-ISO Series. Each section covers a group of Service Request properties or messages, using the *Edit Service Request* dialog as a reference. This chapter shows which DriverLINX features the KPCI-ISO driver implements and gives notes on their usage. For detailed explanations or examples of DriverLINX features, in general, see one of the following references:

- *DriverLINX/VB Technical Reference Manual (ActiveX Control)*
- *DriverLINX Technical Reference Manual (C/C++)*
- *DriverLINX Digital I/O Programming Guide*



A Service Request uses the following properties and messages:

Request Group:

- **Modes**—Specify how DriverLINX sets up data transfers and synchronizes execution with the application.
- **Operations**—Specify what service DriverLINX performs.

Events Group:

- **Timing Events**—Specify how DriverLINX and the board pace sampling.
- **Start Events**—Specify when sampling begins.
- **Stop Events**—Specify when sampling ends.

Select Group:

- **Channels**—Specify which channels, or ports, to sample.
- **Buffers**—Specify the size and number of sample storage buffers.

Service Request Messages:

- **Messages**—ActiveX control events or Windows messages that DriverLINX sends to synchronize execution with the application.

---

## Digital Output Modes

The Service Request Mode specifies how DriverLINX sets up data transfers and synchronizes execution with the application.

The KPCI-ISO Series Digital Output Subsystem supports the following modes:

- **Polled**—For single value or single scan digital output samples.
- **Interrupt**—For buffered transfers using programmed I/O.
- **Other**—For subsystem initialization and data conversion.

---

## Digital Output Operations

The Service Request Operation specifies what service DriverLINX performs.

The KPCI-ISO Series Digital Output Subsystem supports the following DriverLINX operations:

- **Initialize**—Aborts any active digital output tasks started by the application. The Initialize operation does not abort tasks started by other applications.
- **Start**—Initiates a data-acquisition the task specified in the Service Request.
- **Status**—Reports the buffer position of the next sample that DriverLINX will process.
- **Stop**—Terminates a digital output data-acquisition task.

- **Message**—Displays a dialog box containing the last Service Request error message, if any.

## Digital Output Timing Events

Timing Events specify how DriverLINX and the board pace, or clock, Digital Output sampling. DriverLINX uses the Timing Event to program when the KPCI-ISO writes the next digital output sample to the port.

The KPCI-ISO supports the following Timing Events:

- **None**—The task requires no pacing as DriverLINX is writing only a single value or simultaneous scan.
- **Rate**—The KPCI-ISO Series only supports fixed rate digital output using either a software system clock or an external input signal (INT\_REQ).
- **Digital**—DriverLINX uses an external digital input signal (INT\_REQ) to pace the transfer of each sample.

### None or Null Event

The Null Event specifies that the task does not need a clock to determine when to write the next sample.

### Rate Event

The KPCI-ISO Series supports one type of Rate Event for digital output:

- **Rate Generator**—Generates a rate clock with equal (internal clocking) or unknown (external clocking) time intervals between tics.



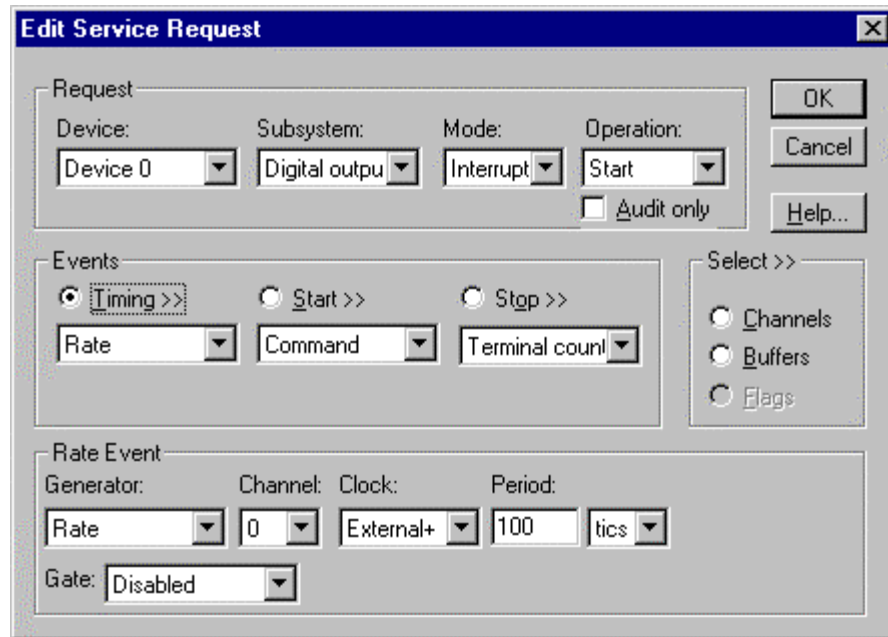
### Rate Generator: External Clocking

An externally clocked Rate Generator produces a rate clock with unknown time intervals between tics.



Period (ext clk)

Use an externally clocked rate generator when you want to synchronize digital output samples with a recurrent external signal.



How to set up the KPCI-ISO for rate sampling using an external clock.

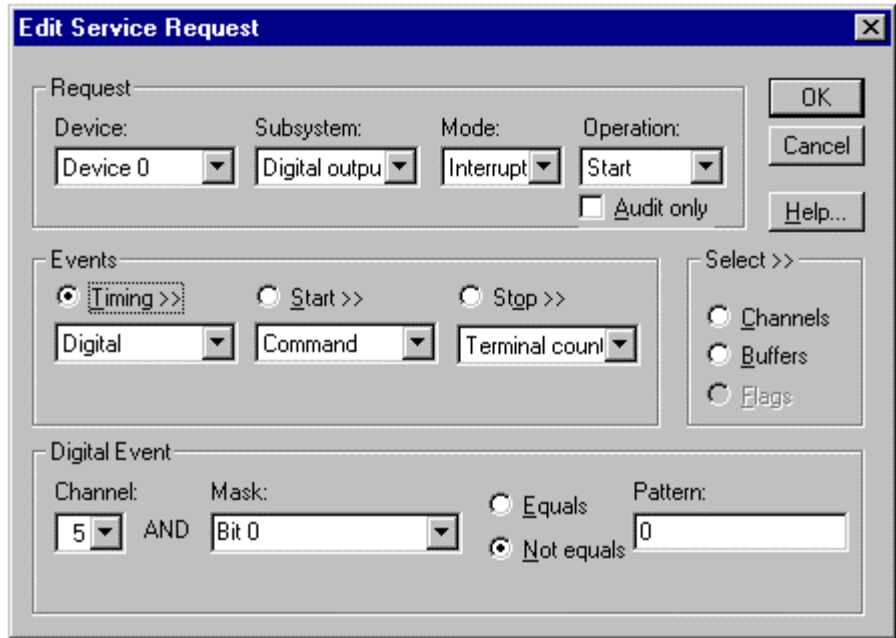
*A task cannot use the INT\_REQ line for a digital start or stop event while using it as an external clock source.*

- Specify external clocking using a **Rate Generator** on *Logical Channel 0* with an **External**, **External+**, or **External-** Clock source. External and External+ both specify sampling on the rising, or positive, edge of the external clock signal. External- specifies sampling on the falling, or negative, edge of the external clock signal.
- Users should connect the external clock signal to the INT\_REQ line.
- The *Period* value does not affect the external clock frequency, but DriverLINX requires a valid hardware value in case the application requests a timebase operation and to optimize data transfer between the driver and the application.

## Digital Event

DriverLINX supports Digital Events as aliases for externally clocked Rate Generators. Use this technique for compatibility with data-acquisition products that only support external clock sources.





How to set up the KPCI-ISO for external rate sampling using a digital event.

Digital Timing Events contain *mask*, *pattern*, and *match* fields. The mask is logically AND with the digital input data on the Logical Channel and then compared against the *pattern* for a match/mismatch.

*A task cannot use a channel for a digital start or stop event while using it for a digital timing event.*

- Specify external clocking using *Logical Channel 3* (KPCI-PDISO8A) or **5** (KPCI-PIO32IOA). For hardware-independence, you can specify the hardware external trigger channel by the symbolic constant, *DI\_EXTCLK*.
- Users should connect the external clock signal to the INT\_REQ line.
- Specify the *Mask* property with a single bit set to indicate that DriverLINX should only compare a 1-bit digital input value against the *Pattern* property.
- Specify the *Match* property as **Not equals**.
- Specify the *Pattern* property as **0** for a rising, or positive, edge clock ( $\neq 0$ ), or **1** for a falling, or negative, edge clock ( $\neq 1$ ).

---

# Digital Output Start Events

Start Events specify when the KPCI-ISO hardware starts writing digital output data.

The KPCI-ISO supports the following Start Events for digital output:

- **None**—Use this event when the DriverLINX operation does not require a Start Event.
- **Command**—DriverLINX starts the task on software command, i.e., as soon as DriverLINX finishes programming the KPCI-ISO hardware for the task.
- **Digital**—The KPCI-ISO starts writing digital output samples when a digital Logical Channel input satisfies the digital condition specified in the Start Event.

## None or Null Event

The Null Event specifies that the task does not need a Start Event to begin the task.

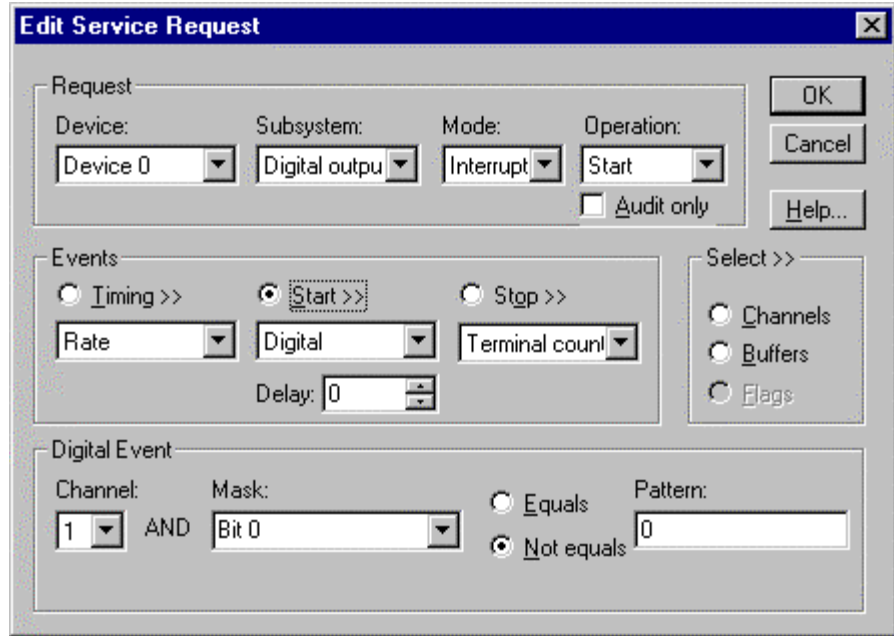
## Command Event

The Command Event starts data acquisition as soon as DriverLINX has completed programming the KPCI-ISO hardware with the task parameters.

## Post-Trigger Sampling with a Digital Event

The KPCI-ISO can write digital output samples *after* a digital trigger condition. Use post-triggering when you want to synchronize the start of data acquisition with an external signal.

DriverLINX supports digital triggering on all digital input lines. However, the KPCI-ISO has only one digital input, INT\_REQ, that can detect a trigger precisely when it occurs. For all other digital inputs, DriverLINX uses a software comparison to detect the trigger at the digital input port during the interrupt service routine. DriverLINX can reliably detect triggers only if either the triggers either have pulse durations longer than the interrupt latency (unlatched mode) or the application has previously configured the digital input trigger port as latched.



How to set up the KPCI-ISO for post-triggered digital output.

*A Digital Start Event requires an external timing source because the KPCI-ISO does not have a hardware clock.*

*A task cannot use a channel for a digital start or stop event while using it for a digital timing event.*

Digital Start Events contain *mask*, *pattern*, and *match* fields. The mask is logically AND with the digital input data on the Logical Channel and then compared against the *pattern* for a match/mismatch.

- Specify the *Logical Channel*.
- Specify the *Mask* property to indicate that DriverLINX should only compare selected bits in the digital input value against the *Pattern* property.
- Specify the *Match* property as **Equals** to trigger on an exact match of the digital pattern to the masked digital input. Match on **Not Equals** triggers on any difference between the masked digital input and the *Pattern*.
- Specify the *Pattern* property to compare with the masked digital input value.
- Specify the *Delay* property as any number of timing pulses from 0 to  $(2^{32} - 1)$ .

## Digital Output Stop Events

Stop Events specify when the KPCI-ISO hardware stops writing digital output data.

The KPCI-ISO supports the following Stop Events for digital output:

- **None**—Use this event when the DriverLINX operation does not require a Stop Event.
- **Command**—DriverLINX stops the task on software command, i.e., when the application issues a Service Request with a *Stop* operation.

- **Terminal count**—DriverLINX stops the task after the KPCI-ISO hardware has filled all the data buffers once.
- **Digital**—The KPCI-ISO stops the task when the hardware detects the digital Logical Channel input satisfies the digital condition specified in the Stop Event.

## None or Null Event

The Null Event specifies that the task does not need a Stop Event to end the task.

## Command Event

The Command Event stops data acquisition when the user application changes the *Operation* property in the Service Request to *Stop* and resubmits the Service Request to DriverLINX.

Sample timing depends on the task mode. In interrupt mode, DriverLINX samples at each timing event until the *Stop* operation. In polled mode, an application can repeatedly initiate a single scan with a *Status* operation before terminating the task with a *Stop* operation.

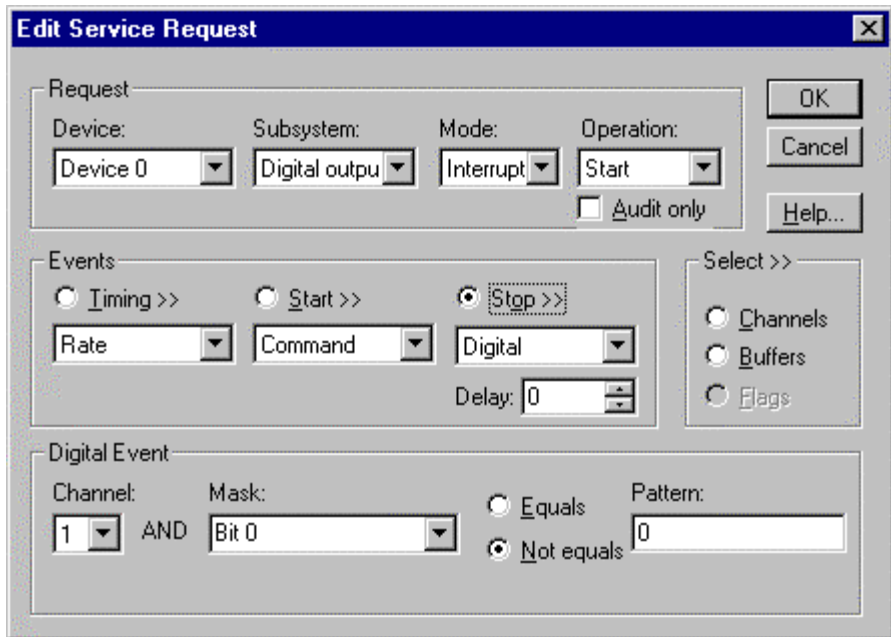
## Terminal Count Event

The Terminal Count Event stops data acquisition after DriverLINX has written the digital output data from all the data buffers *once*. Use terminal count when you want to write a fixed amount of data.

## Pre-Trigger Sampling with a Digital Event

The KPCI-ISO can write digital output samples *until* the hardware detects a digital trigger condition. Use pre-triggering when you want to synchronize the end of data transfer with an external signal.

DriverLINX supports digital triggering on all digital input lines. However, only the KPCI-ISO has a digital input, INT\_REQ, that can detect a trigger precisely when it occurs. For all other digital inputs, DriverLINX polls at the specified sample rate to detect the trigger. Polling can reliably detect triggers only if they have pulses at least as long as the sample period.



How to set up the KPCI-ISO for pre-triggered digital output.

Digital Start Events contain *mask*, *pattern*, and *match* fields. The mask is logically AND with the digital input data on the Logical Channel and then compared against the *pattern* for a match/mismatch.

- Specify the *Logical Channel*.
- Specify the *Mask* property to indicate that DriverLINX should only compare selected bits in the digital input value against the *Pattern* property.
- Specify the *Match* property as **Equals** to trigger on an exact match of the digital pattern to the masked digital input. Match on **Not Equals** triggers on any difference between the masked digital input and the *Pattern*.
- Specify the *Pattern* property to compare with the masked digital input value.
- Specify the *Delay* property as the number of timing pulses (0 to  $2^{32} - 1$ ) to ignore after the trigger before starting data acquisition. The *Delay* property must be an integral multiple of the number of channels in the channel list.

## Digital Output Channels

DriverLINX allows applications to specify the digital channels using three techniques:

- **Start Channel**—Write data using a single channel.
- **Start/Stop Channel Range**—Write data using a consecutive range of channels.
- **Channel List**—Write digital data using a list of channels.

In addition, DriverLINX provides two methods to address one or more digital I/O lines:

- **Extended Logical Channels Addressing**—Write data to 1, 2, 4, 8, 16, or 32 bits using a single Logical Channel.
- **Bit-Masked Addressing**—Write data to specific bits of an output port.

## Digital Output Logical Channels

The KPCI-ISO has one or two digital output ports that DriverLINX designates as Logical Channels. The last output values can be read back using the Digital Input Subsystem.

### KPCI-PDISO-8A

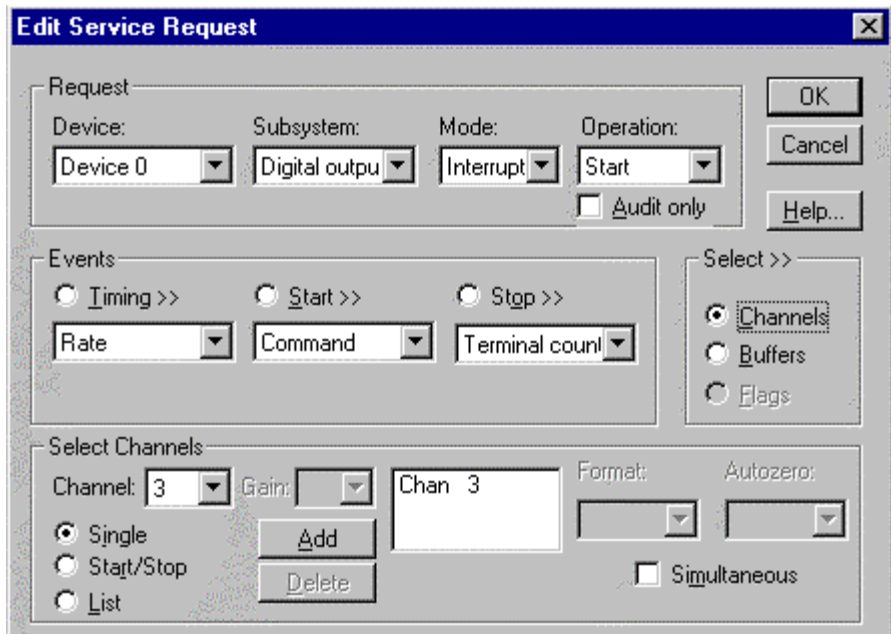
Logical Channel	DriverLINX Function	External Connector
0	Digital Input Port placeholder (usage results in an error)	N/A
1	Digital Output Port Read Back	OP0 – OP7

### KPCI-PIO32IOA

Logical Channel	DriverLINX Function	External Connector
0	Digital Input Port placeholder (usage results in an error)	N/A
1	Digital Input Port placeholder (usage results in an error)	N/A
2	Digital Output Port Read Back	OP0 – OP7
3	Digital Output Port Read Back	OP8 – OP15

## Single Channel Digital Output

In Single Channel mode, the KPCI-ISO board writes all data from one channel.

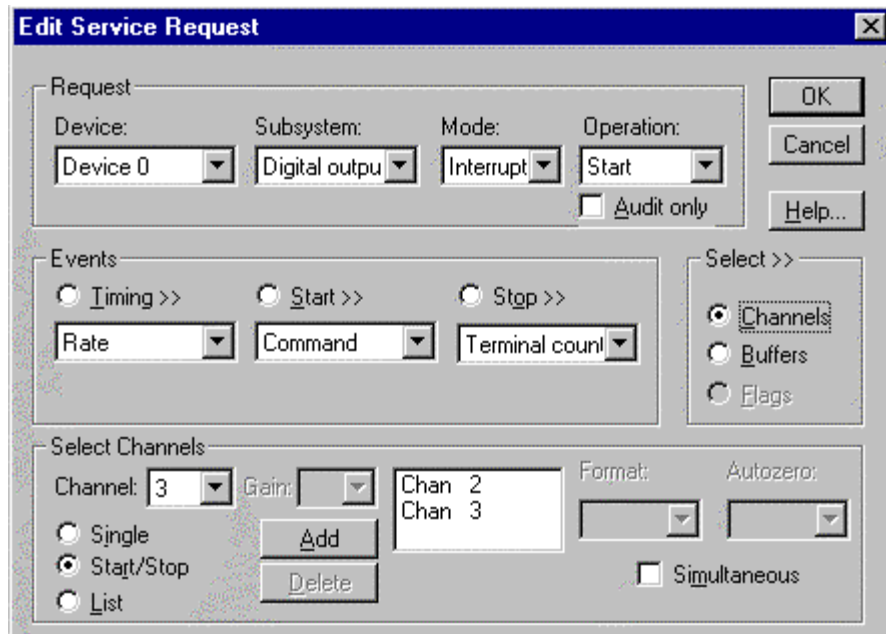


*How to set up the KPCI-ISO to write to a single channel*

## Multi-channel Digital Output Range

In Multi-channel Range mode, the KPCI-ISO board writes data from a consecutive range of digital channels.

- Specifying a Stop Channel equal to the Start Channel is the same as single channel mode.
- Specifying *Simultaneous*, instructs DriverLINX to write to all the selected channels at each timing event. Otherwise, DriverLINX writes one sample from the next channel in sequence at each timing event.

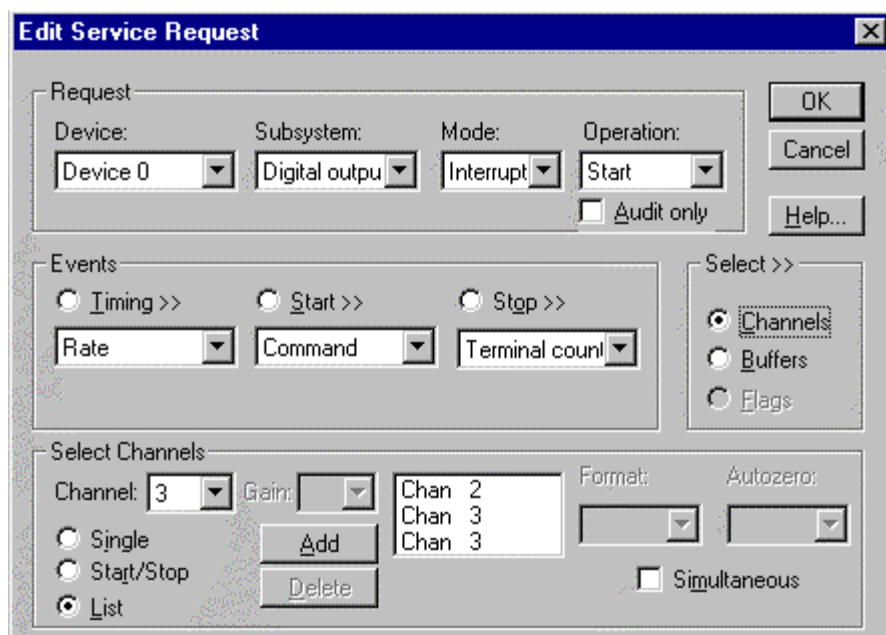


How to set up the KPCI-ISO to write to a range of channels

## Multi-channel Digital Output List

In Multi-channel List mode, the KPCI-ISO writes data from a random list of digital channels.

- Specifying *Simultaneous*, instructs DriverLINX to sample all the selected channels at each timing event. Otherwise, DriverLINX writes one sample from the next channel in sequence at each timing event.



How to set up the KPCI-ISO to write to a range of channels



## Extended Logical Channel Addressing

DriverLINX supports an extension to Logical Channel addressing allowing applications to combine adjacent Logical Channels into a single, wider channel or split a Logical Channel into smaller addressable parts. For instance, applications can address individual bits of a digital port or read and write multiple ports as a single value.

To use the Logical Channel addressing extensions, form a 16-bit Logical Channel address by combining the channel number of an addressable unit with a size field as follows:

<b>0</b>	<b>Size Code</b>				<b>Channel Number</b>											
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MSB															LSB

The following table lists the 3-bit size codes available for the KPCI-ISO Series:

Size Code	Unit	Bits
0	native	varies with hardware
1	bit	1
2	half nibble	2
3	nibble	4
4	byte	8
5	word	16

*"Native" units refer to the hardware-defined digital channel size. For the KPCI-ISO Series, the native size is an 8-bit byte.*

When using extended Logical Channel addressing, DriverLINX groups digital bits in units defined by the size code and then assigns consecutive channel numbers starting from zero.

For the KPCI-ISO models, the 8-bit digital input/output ports have the following channel addresses for each size code:

### KPCI-PDISO-8A

Unit	Channels	Address (dec)	Address (hex)
native	0 .. 1	0 .. 1	0000 .. 0001
bit	0 .. 15	4096 .. 4111	1000 .. 100F
half nibble	0 .. 7	8192 .. 8199	2000 .. 2007
nibble	0 .. 3	12288 .. 12291	3000 .. 3003
byte	0 .. 1	16384 .. 16385	4000 .. 4001
word	0 .. 0	20480 .. 20480	5000 .. 5000

### KPCI-PIO32IOA

Unit	Channels	Address (dec)	Address (hex)
native	0 .. 3	0 .. 3	0000 .. 0003
bit	0 .. 31	4096 .. 4127	1000 .. 101F
half nibble	0 .. 15	8192 .. 8207	2000 .. 200F
nibble	0 .. 7	12288 .. 12295	3000 .. 3007
byte	0 .. 3	16384 .. 16387	4000 .. 4003
word	0 .. 1	20480 .. 20481	5000 .. 5001
dword	0 .. 0	24576 .. 24576	6000 .. 6000

#### Notes:

- The first half of each of the following ranges is for reading back values output to the second half of the range. Using the Logical Channels in the first half in the Digital Output Subsystem is an error.
- DriverLINX supports single-value transfers of any model-supported width.
- DriverLINX supports multi-value transfers only using data widths smaller than or equal to the hardware native size.

## Bit-Masked Addressing

In addition to Extended Logical Channel Addressing, DriverLINX also allows applications to access specific bits in a single channel. This technique is more convenient when an application needs to access bits that are not grouped together.

Bit masking allows an application or thread to access specific bits in a port without affecting bits controlled by other applications.

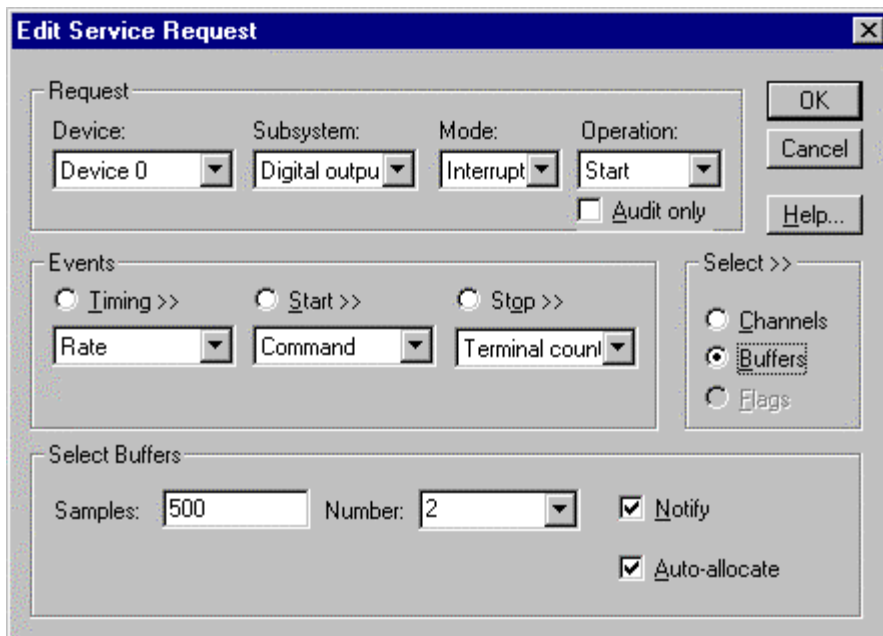
For more information, see “Reading or Writing Specific Digital Bits” in the *Digital I/O Programming Guide*.

---

## Digital Output Buffers

DriverLINX supports both single-value digital output and buffered digital output.

- **For single-value input**, specify the Number of buffers as **0** and the buffer size as **0**.
- **For buffered input**, specify the Number of buffers from **1** to **255** and the buffer size as desired.



*How to set up the KPCI-ISO to write digital samples using data buffers*

An individual DriverLINX buffer may have any size as long as the buffer length holds an integral number of channel scans, i.e., the number of digital output channels you're outputting. This restriction enforces the requirement that a buffer has the same number of samples for each channel.

## Digital Output Messages

For digital output operations, DriverLINX can report the following messages to the application:

DriverLINX Message	Explanation
Service Start	DriverLINX has started the data acquisition task.
Service Done	DriverLINX has completed the data acquisition task.
Buffer Filled	DriverLINX has emptied a data buffer of digital output samples.
Start Event	DriverLINX has processed the interrupt for a hardware start event. <i>Applicable only to the ExtTrg channel on the KPCI-ISO Series.</i>
Stop Event	DriverLINX has processed the interrupt for a hardware stop event. <i>Applicable only to the ExtTrg channel on the KPCI-ISO Series.</i>
Data Lost	DriverLINX has detected a digital output data overrun condition.
Critical Error	DriverLINX has encountered an unexpected hardware or software condition.

*DriverLINX Event message for digital output*

For details on using DriverLINX messages in your application see the appropriate reference manual:

- ActiveX Control—“Overview of DriverLINX/VB Control Events” in the *DriverLINX/VB Technical Reference Manual*.
- C/C++—“DriverLINX Messages - Overview” in the *DriverLINX Technical Reference Manual*.

# Using the Counter/Timer Subsystem

---

## Introduction

The Counter/Timer Subsystem for the KPCI-ISO has a single “virtual” counter/timer. DriverLINX provides this counter/timer through software, as the KPCI-ISO does not have a hardware clock. You can use the counter/timer to externally pace input/output subsystems, but it does not support independent counting or timing applications.

---

## Counter/Timer Channels

The following table lists the Counter/Timer Subsystem’s Logical Channels and shows their allowable clock sources, modes and gates.

Logical Channel	Clocks		Modes	Gates
	Source	Tic Period		
0	External	1 $\mu$ s (1 MHz)	Rate Gen	Disabled No Connect
	External+			
	External-			

*Counter/Timer Subsystem Logical Channels and Allowed Clocks, Modes and Gates*

### Pacer Clock

The Pacer Clock (Logical Channel 0) is a virtual software clock. DriverLINX uses this external clock to pace digital I/O as the KPCI-ISO Series does not have an

internal hardware timer. The Pacer Clock does not support independent counter/timer tasks.

### ***External Clocking***

The KPCI-ISO Series allows an external clock source to pace input/output tasks. DriverLINX defines the following external clock sources for the System Pacer Clock:

- **External, External+** —specify sampling on the rising, or positive, edge of the external clock signal.
- **External-** —specifies sampling on the falling, or negative, edge of the external clock signal.

### ***Clocking Modes***

The System Pacer Clock can only operate in the following mode:

- **Rate Generator**—specifies sampling of one channel or one scan at intervals based on an external clock source.

### ***Gating***

The system timer does not have a gate control. Select **Disabled** or **No Connect** for the *Gate Status* of Rate Events.

---

## **Counter/Timer Messages**

The KPCI-ISO supports counter/timer interrupts indirectly. You can set up an INTERRUPT mode digital input or output task with a single sample buffer. “Buffer Filled” messages notify your application of the counter/ timer interrupt.



# Uninstalling DriverLINX

---

## How do I uninstall DriverLINX?

DriverLINX consists of three separate component installations:

- DriverLINX for Keithley KPCI-ISO Series
- DriverLINX Programming Interfaces
- DriverLINX Documentation

You can uninstall the last two installations at any time without interfering with compiled applications that require DriverLINX drivers. To uninstall the latter components, run the “Add/Remove Programs” tool in the Windows Control Panel.

To uninstall DriverLINX drivers for the Keithley KPCI-ISO Series, you must

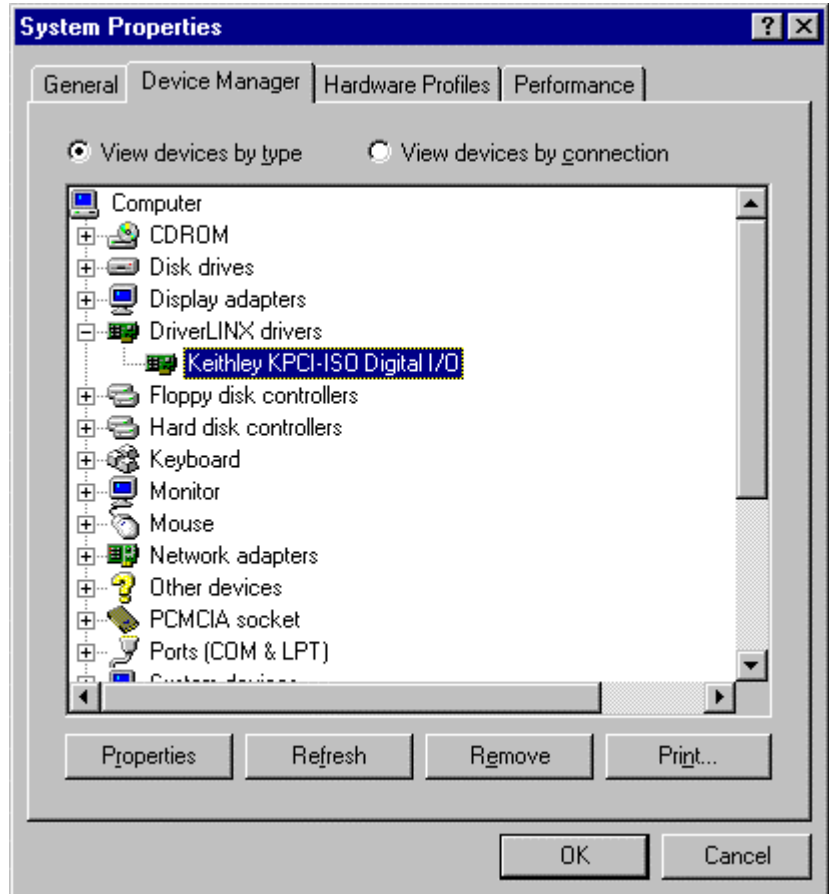
- Disable the DriverLINX driver.
- Shut down your computer to remove the hardware.
- Reboot your computer to unload the driver.
- Run the DriverLINX uninstall program.

### ***How to Disable a DriverLINX Driver in Windows NT***

1. From the Windows Start menu, select “Settings”, then “Control Panel”. Left click on the DriverLINX Configuration icon in the Control Panel.
2. Select the KPCI-ISO devices you want to disable.
3. Right click on each device and select “Disabled” on the popup menu.
4. Repeat steps 2-3 for each KPCI-ISO board that you are uninstalling.
5. Close the DriverLINX Configuration Panel.
6. When finished, shut down your computer and physically remove any installed KPCI-ISO hardware.
7. Reboot Windows.
8. To finish uninstalling, see “How to Remove DriverLINX for Keithley KPCI-ISO Series” on page 80.

## How to Disable a DriverLINX Driver in Windows 98/Me

1. From the Windows Start menu, select “Settings”, then “Control Panel”. Left click on the System icon in the Control Panel. Select the “Device Manager” tab in the System Properties dialog.
2. Left click the “+” icon next to “DriverLINX drivers” to display the installed Keithley KPCI-ISO devices.



3. Select the KPCI-ISO device you want to disable.
4. Click the “Remove” button.
5. In the “Confirm Device Removal” dialog, select “OK”.
6. Repeat steps 3-5 for each KPCI-ISO board or driver that you uninstalling.
7. When finished, click “Close”, shut down your computer, and physically remove any installed KPCI-ISO hardware.
8. Reboot Windows.
9. To finish uninstalling, see “How to Remove DriverLINX for Keithley KPCI-ISO Series” on page 80.

## How to Remove DriverLINX for Keithley KPCI-ISO Series

1. From the Windows Start menu, select “Settings”, then “Control Panel”. Left click on the Add/Remove Programs icon in the Control Panel.



2. Select “DriverLINX for Keithley KPCI-ISO Series” in the Add/Remove Programs Properties dialog.
3. Click the “Add/Remove...” button.
4. Answer “Yes” to “Are you sure you want to remove ‘DriverLINX for Keithley KPCI-ISO Series’ and all of its components?” in the Confirm File Deletion dialog.
5. The DriverLINX uninstall program will proceed.

---

The uninstall program will not remove the folder, “\DrvLINX4\System”. This folder contains copies of any \Windows\System files that the original DriverLINX installation updated.

---



# Troubleshooting

---

## Solving Problems

Correct operation of your KPCI-ISO hardware requires successful completion of three steps.

1. Windows recognizes KPCI-ISO hardware and installs drivers.
2. You configure the KPCI-ISO drivers using the DriverLINX Configuration Panel.
3. Windows loads the KPCI-ISO drivers into memory.

If you are having a problem installing or configuring your KPCI-ISO product, review the following notes. If these notes do not solve your problem, or your problem is not described, then contact technical support and fully describe your problem.

### Solving Problems Recognizing and Installing Drivers

Windows must recognize the KPCI-ISO hardware and then install KPCI-ISO drivers for the hardware.

- For Windows to automatically recognize the KPCI-ISO hardware, you must install a KPCI-ISO product into your machine and restart the computer.
- For Windows to install KPCI-ISO drivers, you must install the KPCI-ISO driver software on the DriverLINX CD.

If you installed the KPCI-ISO hardware before you installed the DriverLINX software, Windows 98/Me may install the KPCI-ISO hardware without drivers. To fix this, you must instruct Windows to reinstall the Plug and Play driver. Follow the notes below for instructions.

### ***Fixing Problems Installing Hardware Before Software on Windows 98/Me***

If you installed the KPCI-ISO hardware before installing the DriverLINX software, Windows 98/Me may install the hardware incorrectly.

1. Start Windows Device Manager.

2. Search for “DriverLINX drivers”. If you do not see it, go to step 4.
3. Click on the “+” icon next to “DriverLINX drivers” and search for “PCI Card”. If you find “PCI Card”, go to step 6, otherwise go to the next step.
4. Search for “Other devices”.
5. Click on the “+” icon next to “Other devices” and search for “PCI Card”.
6. Select “PCI Card”.
7. Click “Remove” in the Device Manager.
8. Highlight “Computer” in the Device Manager and click “Refresh”.
9. Windows should re-install your hardware under “DriverLINX drivers”.
10. Answer “Yes” to “Do you want to restart your computer now?”.
11. After rebooting your computer, you must configure DriverLINX. See “Configuring the KPCI-ISO Series” on page 11.

## Solving Problems Configuring the Drivers

Windows automatically assigns hardware resources for the KPCI-ISO Series, but you must still configure the KPCI-ISO drivers before using them. The DriverLINX configuration only requires that you assign each KPCI-ISO product a unique Logical Device number. And, on Windows NT, you must manually select the hardware model of your KPCI-ISO board. See “Configuring the KPCI-ISO Series” on page 11 for more information.

## Solving Problems Loading Drivers

Before the KPCI-ISO drivers can load, you must

1. Install the DriverLINX software.
2. Install the KPCI-ISO hardware into your computer.
3. Configure DriverLINX.
4. Reboot your computer.

If you have not completed the above steps, please do so before proceeding.

On Windows 2000/XP, Windows NT and Windows 98/Me, the operating system will automatically assign hardware resources to the KPCI-ISO cards. Automatic resource assignment can fail sometimes on

- Older PCI computers
- Computers with ISA cards installed
- Computers with no free hardware resources.

Sorting through all possibilities can be a challenge due to the sheer number of combinations of PCI hardware designs, PC plug-in boards, and versions of Windows. The following sections will help you gather information about why a driver may have failed to load. This information is essential for you or technical support to solve your problem.

## ***Did the DriverLINX Driver Load?***

1. Run “DriverLINX Configuration” from Windows Control Panel.
2. Select the “DriverLINX” tab.
3. Click the “+” icon next to DriverLINX to expand the list of drivers, if necessary.
4. Select “Keithley KPCI-ISO Series”. Click “+”, if necessary, to expand the list.
5. Select the line with the number of the Logical Device you configured. If the number does not exist on Windows NT or, on Windows 98/Me, you see only a hardware description, you did not configure the driver. See “Configuring the KPCI-ISO Series” on page 11.
6. Click the “Properties...” button and then select the “General” tab.
7. Do you see “Status: Device Loaded”? If not, did you reboot the computer after configuring? If not, reboot now and repeat the above steps.



8. If you rebooted the computer after configuring and Windows did not load your device, see “Checking for Device Errors” on page 85.

## ***Checking for Device Errors***

When a DriverLINX kernel driver cannot load, it usually writes an explanation into the system event log. You can view this log under Windows 98/Me or Windows NT using the DriverLINX Event Viewer.

Windows 98/Me maintains additional driver information in the Device Manager. Also see “Getting More Driver Information on Windows 98/Me” on page 86.

1. Run “DriverLINX Event Viewer” from the DriverLINX folder.
2. Click on the “+” icon next to “DriverLINX” in the left panel.
3. Select the abbreviation for your driver.
4. Does the first line in the right panel show a current error?
5. Double click on the error line to see more detail and an explanatory message.

6. If you cannot resolve the problem yourself, please provide this error information when contacting technical support.

### ***Getting More Driver Information on Windows 98/Me2000/XP***

Windows 98/Me/2000/XP reports additional information about device status using the Device Manager. To access this utility,

1. Right click on “My Computer” and then select “Properties”.
2. Select “Device Manager” and “View devices by type
3. Does “DriverLINX drivers” appear in the list? If not, see "Solving Problems Recognizing and Installing Drivers" on page 83.
4. Click the “+” next to “DriverLINX drivers”.
5. Does your KPCI-ISO product appear in the list? If not, see "Solving Problems Recognizing and Installing Drivers" on page 83.
6. Does the icon next to your KPCI-ISO product display an exclamation point (!)? If no, Windows has loaded your KPCI-ISO driver.
7. Select the line with the “!” and then click “Properties”.
8. The General tab will show the reason why the driver did not load.
9. The Resources tab will show if Windows detected an unresolvable hardware conflict.

### ***Getting More Driver Information on Windows NT***

On Windows NT, the only reasons that a driver does not load are

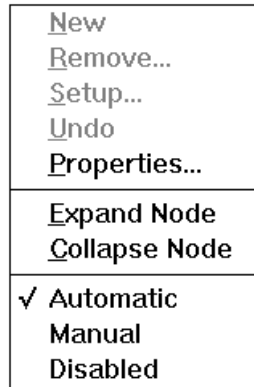
- You did not install the driver software.
- You did not correctly configure the driver.
- You changed the driver startup parameters.

An incorrectly configured driver will report the reasons that it failed to load into the Windows Event Log. See “Checking for Device Errors” on page 85 for more information.

On Windows NT, DriverLINX drivers load automatically during system boot. An administrator can change the startup command for any NT driver to either “manual” or “disabled”.

1. Run “DriverLINX Configuration” from Windows Control Panel.
2. Select the “DriverLINX” tab.
3. Click the “+” icon next to DriverLINX to expand the list of drivers, if necessary.
4. Select “Keithley KPCI-ISO Series”. Click “+”, if necessary, to expand the list.
5. Select the line with the number of the Logical Device that did not load.

6. Right click the mouse to see a popup menu.



7. Select “Automatic” to instruct Window to load the driver the next time you reboot.

---


## Generating a DriverLINX Configuration Report

Your DriverLINX installation includes a troubleshooting tool that generates a report of your DriverLINX configuration. If you call Technical Support, after reading “Solving Problems” on page 83, they may ask you to generate and e-mail this report to help you solve installation and configuration problems.

### What is in the Report?

The troubleshooting tool analyzes your computer to obtain information about DriverLINX and operating system software that would assist Technical Support in troubleshooting a problem you are having. It includes information on DriverLINX files, environment variables, registry entries, hardware and the operating system.

### How do I Generate the Report?

You can easily generate the report by clicking this shortcut . Once the troubleshooting tool generates the report, you will have the opportunity to review it and make deletions, if desired, before e-mailing it to Technical Support. If you do not have direct access to e-mail, you can save the report to a disk file and send a copy later. A Technical Support engineer will guide you through these steps when you are asked to send a report.





# Glossary of Terms

## **A/D**

Abbreviation for analog-to-digital, a process that converts a continuous analog signal into a discrete digital approximation of the analog signal.

## **ADC**

Abbreviation for analog-to-digital converter, the hardware that performs the A/D conversion process.

## **API**

Abbreviation for Application Programming Interface. An API defines the syntax of the data structures and functions of software services.

## **Basic I/O**

Intel 8255 Programmable Peripheral Interface chip Mode 0. Basic I/O supports simple input or output without control and status signals. Outputs are latched and inputs are not latched.

## **Bidirectional I/O**

Intel 8255 Programmable Peripheral Interface chip Mode 2. Bidirectional I/O supports bidirectional input and output with control and status signals. The 8-bit inputs and outputs are latched.

## **Buffer**

A block of memory used to receive data from a data-acquisition device or to write data to a data-acquisition device.

## **Bus mastering**

A hardware technique that allows a device on the PCI bus to initiate a direct data transfer with memory or another device. The logic for controlling the transfer resides on the PCI device, not the system board. See also “DMA”.

## Clocking

A periodic pulse or signal that data-acquisition hardware uses to read or write the next sample or block of samples. Also referred to as “pacing”.

## D/A

Abbreviation for digital-to-analog, a process that converts a discrete digital value into a continuous analog voltage representing that value.

## DAC

Abbreviation for digital-to-analog converter, the hardware that performs the D/A conversion process.

## DMA

Abbreviation for Direct Memory Access, a technique where the system board can transfer data between a device and memory without using the CPU. In the PC, a standard chip on the system board controls the transfer. See also “bus mastering”.

## Event

For DriverLINX, an event is the occurrence of a signal that clocks, starts, or stops a data-acquisition task.

## Event Log

An event log is a file where applications and device drivers record troubleshooting messages. If your Logical Device fails to load, check the event log to find out why. You can view event log entries using the DriverLINX Event Viewer or Windows NT Event Viewer.

## Gating

A signal that enables and disables another signal or data-acquisition task depending on the value of the gate signal.

## IRQ

Abbreviation for interrupt request. Peripheral hardware signals the CPU that it is ready to transfer data.

## ISA

Abbreviation for Industry Standard Architecture. A standard for the original IBM AT bus specification that defines the bus structure, CPU and support chip architecture, and the clock frequency of the ISA bus.

## **ISR**

Abbreviation for interrupt service routine, the software function inside a device driver that handles interrupt requests.

## **Logical Device**

DriverLINX's designation for a specific data-acquisition board inside your computer.

## **Messages**

In Windows and DriverLINX, a message notifies the application about the state of a process.

## **Modes**

DriverLINX data-acquisition techniques.

## **Nibble**

Four bits, or half of a Byte.

## **Operations**

Allowed DriverLINX data-acquisition commands.

## **Pacing**

A periodic pulse or signal that data-acquisition hardware uses to read or write the next sample or block of samples. Also referred to as "clocking".

## **PCI**

Abbreviation for Peripheral Component Interconnect. PCI refers to a specification for a high-speed common local bus on a system board.

## **PCMCIA**

Abbreviation for Personal Computer Memory Board International Association. PCMCIA established a specification for miniaturized boards and sockets for installing peripheral device in portable personal computers called *PC Board*. *PCMCIA* sometimes refers to PC Board.

## **Process**

Refers to the collection of data and code segments and hardware resources that the operating system assigns to one application.

## **Service Request**

A DriverLINX object or data structure that completely defines a data-acquisition task.

## **Single Scan**

One pass through a task's list or range of channels. Some boards can process all the channels simultaneously. On other boards, DriverLINX emulates a simultaneous scan as rapidly as the CPU will allow.

## **Strobed I/O**

Intel 8255 Programmable Peripheral Interface chip Mode 1. Strobed I/O supports input or output with control and status signals. The 8-bit inputs and outputs are latched.

## **Subsystem**

DriverLINX subdivides a general purpose data-acquisition device into six subsystems—Device, Analog Input, Analog Output, Digital Input, Digital Output, and Counter/Timer.

## **Triggering**

The technique of using a pulse or signal to start or stop a data-acquisition task.