Keithley KPCI-PIO Series

# Using DriverLINX with Your Hardware

KEITHLEY

SST 17-1200-1

# Contents

# 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
- Learn DriverLINX, an interactive learning and demonstration program for DriverLINX that includes a Digital Storage Oscilloscope
- Source code for the sample programs
- The DriverLINX Application Programming Interface files for your compiler
- DriverLINX On-line Help System
- *DriverLINX 4.0 Installation and Configuration Guide*
- *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-PIO board with DriverLINX.

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

*DriverLINX 4.0 Installation and Configuration Guide* for your version of Windows.

- 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 Disks.

This manual contains the following chapters:

### Configuring the KPCI-PIO

Describes configuring the KPCI-PIO using the *Configure DriverLINX Device* dialog box.

### Programming KPCI-PIO Hardware

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

### Accessing KPCI-PIO Features Using DriverLINX

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

### Using DriverLINX with the KPCI-PIO Series

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

### Using the Device Subsystem

Describes using DriverLINX to initialize KPCI-PIO hardware.

### Using the Digital Input Subsystem

Describes using DriverLINX to perform KPCI-PIO digital input.

### Using the Digital Output Subsystem

Shows using DriverLINX to perform KPCI-PIO digital output.

### Using the Counter/Timer Subsystem

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

## Introduction

The *DriverLINX Installation and Configuration Guide* provides detailed instructions for installing and configuring DriverLINX for any Keithley data-acquisition product. This manual explains the steps and special features that apply only to Keithley's KPCI-PIO Series boards.

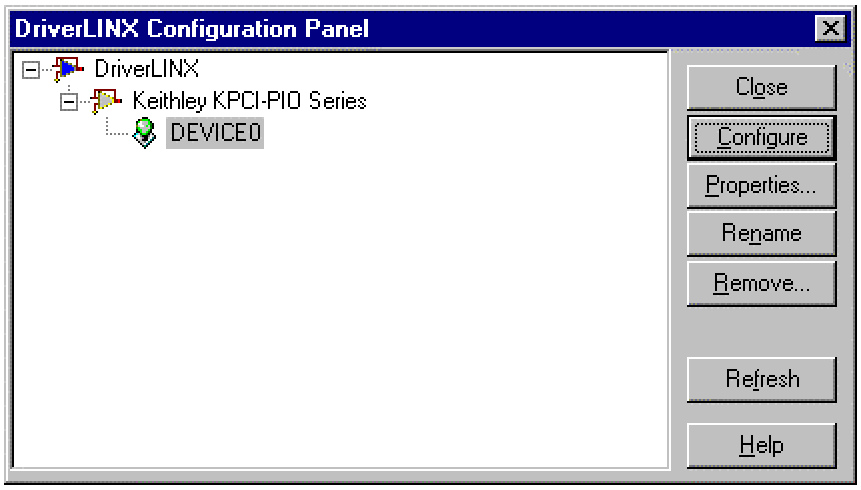
Installing and configuring DriverLINX for the Keithley KPCI-PIO board requires three steps:

1. **To install your KPCI-PIO hardware**, read and follow the instructions in the hardware manual. The "Hardware Installation" chapter of the *DriverLINX Installation and Configuration Guide* also contains useful general information about installing hardware in computers.
2. **To install DriverLINX**, follow the general procedure outlined in the "Installing DriverLINX" chapter of the *DriverLINX Installation and Configuration Guide*.
3. **To configure DriverLINX**, use the *DriverLINX Configuration Panel* and follow the procedure outlined in the "Configuring DriverLINX" chapter of the *DriverLINX Installation and Configuration Guide*. Also see "Configure DriverLINX Device Dialog" on page 11 for configuration options specific to a Keithley KPCI-PIO board.

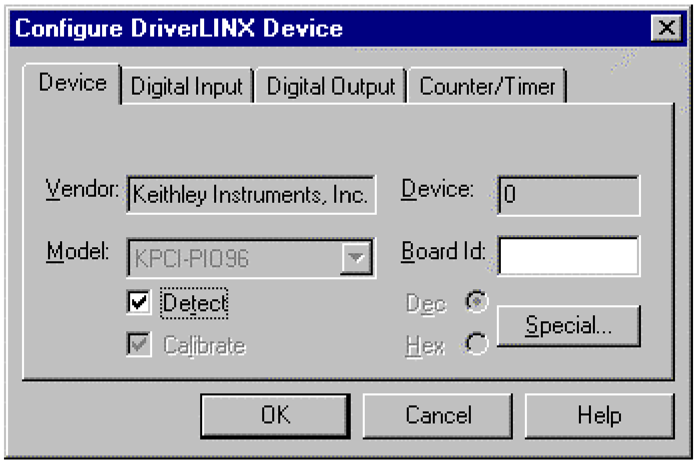
## Configure DriverLINX Device Dialog

DriverLINX uses a standardized configuration protocol for all data-acquisition hardware. Even though Windows automatically assigns resources for the KPCI-PIO, you must still follow the configuration process to assign a DriverLINX Logical Device number to the board and configure board-specific features.

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 the Keithley KPCI-PIO model.


DriverLINX Configuration Panel
DriverLINX
Keithley KPCI-PIO Series
DEVICE0
Close
Configure
Properties...
Rename
Remove...
Refresh
Help

## Device Subsystem Page



Vendor: Keithley Instruments, Inc.

Device: 0

Model: KPCI-PIO96

Board Id:

[✓] Detect

[✓] Calibrate

Dec [✓] Hex

Use the Device Subsystem page to tell DriverLINX the Model and Board Id of your KPCI-PIO 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 both model KPCI-PIO24 and model KPCI-PIO96.

#### Windows 95

Windows 95 automatically determines 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 KPCI-PIO'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 95

Windows 95 automatically determines which board to associate with this Logical Device. DriverLINX enters the serial number of the board when it starts the configuration.

### Windows NT

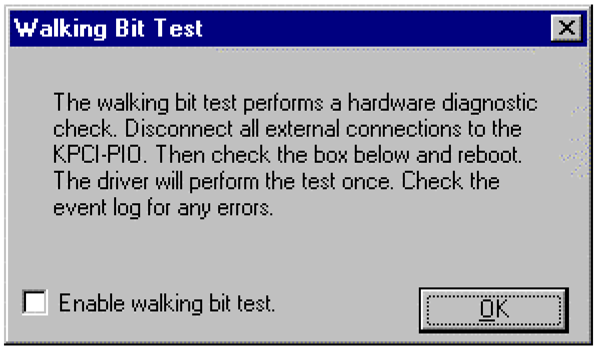
Under Windows NT, Board Id is initially blank. DriverLINX will use the Model to match this Logical Device to the first available board and then enter that board's serial number.

## 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 KPCI-PIO board for internal input/output line damage.


Walking Bit Test

The walking bit test performs a hardware diagnostic check. Disconnect all external connections to the KPCI-PIO. Then check the box below and reboot. The driver will perform the test once. Check the event log for any errors.

To perform the Walking-Bit Test:

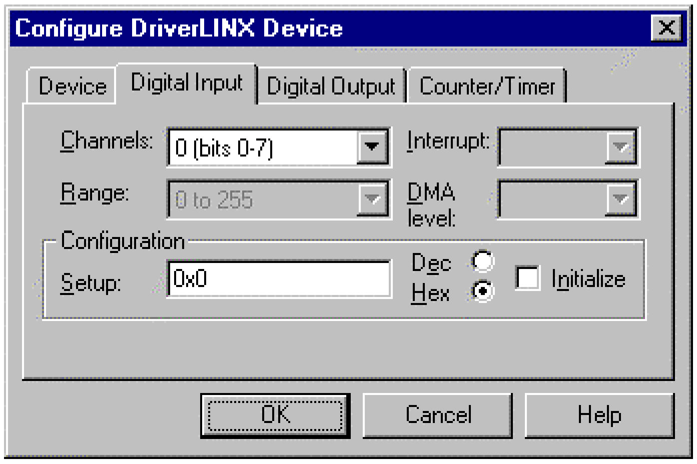
1. Enable the test using the check box.
2. Make sure that Detect is checked on the Device Subsystem Page (see page 13).
3. Remove the cable from your board.
4. Restart your computer.

➢ 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


Configure DriverLINX Device

Device | Digital Input | Digital Output | Counter/Timer

Channels: 0 (bits 0-7)
Interrupt:
Range: 0 to 255
DMA level:
Configuration
Setup: 0x0
Dec ( ) Hex (●)
☐ Initialize

OK | Cancel | Help

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

The initial, default configuration for all digital input/output channels as input port in unlatched mode. You can use this page to change the initial configuration of selected channels to output, and, on the KPCI-PIO24 change input channels to latched mode.

**Note:**

Applications can change a channel's configuration dynamically; See "Digital Port Configuration" on page 55.

### 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-PIO24 has External Clock and External Trigger channels that have fixed, input configurations. See "Digital Input Subsystem Signals" on page 47 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-PIO board. DriverLINX disables this property.

### DMA level

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

## Configuration Setup

*Use caution when configuring and connecting lines to the digital I/O ports. Connecting an input line to an output port, or vice versa, could damage the hardware.*

The *Configuration Setup* property allows you to statically configure each digital I/O port as input or output. The KPCI-PIO allows configuration of each port as either input or output. Model KPCI-PIO24 also allows configuration of 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 them.
- Latched (Strobed mode)—Supports input/output with an external control signal (available only on the KPCI-PIO24). 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 of the strobe signal—its polarity—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-PIO Digital Ports" on page 55.)

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

- The simplified method configures the whole port in basic mode.
- The advanced method allows you to configure groups of digital I/O lines in basic or strobed mode.

## Simplified Digital I/O Port Configuration

For simplified configuration:

1. Select the channel to configure in the drop-down channel list.
2. Check the *Initialize* box.
3. Enter one of the following values for the *Setup* property to configure the entire port:

- **1**—configures all lines in the port as input (unlatched), or
- **0**—configures all lines in the port as output.

## Advanced Digital I/O Port Configuration

With advanced configuration, you can set a port's mode or configure it in parts as input or output.

### Mode Configuration

Model KPCI-PIO24 supports two mode settings: basic and strobed. The KPCI-PIO96 supports only basic mode (see, "Simplified Digital I/O Port Configuration" above).

**Notes:**

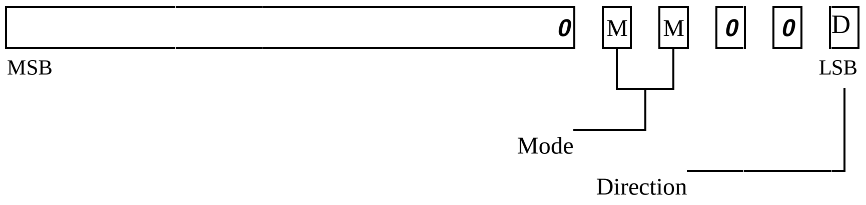
- For compatibility with other DriverLINX PIO drivers, only Ports A and B accept mode setups. For Port C, see "Input/Output Configuration" below.

- On the KPCI-PIO24, all inputs operate in the same mode. Configuring any input in strobed mode configures *all inputs* in strobed mode.

To configure the KPCI-PIO24 ports using mode settings:

1. Select the Port A or B channel to configure in the drop-down channel list.
2. Check the *Initialize* box.
3. Enter a *Setup* value in the following format:

***KPCI-PIO24 Mode Configuration***



- **Mode Field**—
  - 00 = Mode 0 (Basic)

  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 = Mode 1 (Strobed)

  Strobed mode, available on the KPCI-PIO24, 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.

- **Direction Field**—
  - 0—configures the port for output
  - 1—configures the port for input

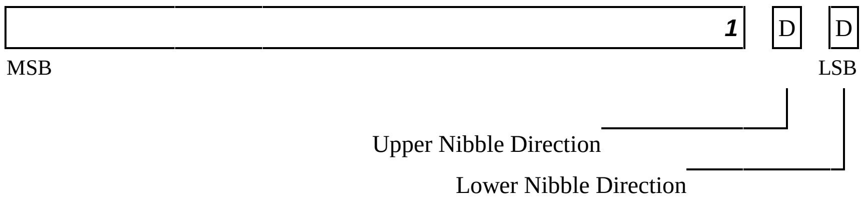
## Input/Output Configuration

Both models support configuration of each Port C nibble.

To configure the KPCI-PIO Port C channel nibbles individually:

1. Select the Port C channel to configure in the drop-down channel list.
2. Check the *Initialize* box.
3. Enter a *Setup* value in the following format:

***Nibble Configuration Format***



- **Direction Fields**—
  - 0—configures the nibble for output
  - 1—configures the nibble for input

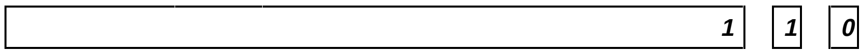
**Advanced Configuration Example:**

This example shows how to configure channel 2 in nibbles. It configures the lower nibble as output and the upper nibble as input.

1. Select the channel 2 in the drop-down channel list.
2. Check the *Initialize* box.
3. Choose the Nibble Configuration format and fill in the Direction fields, right-to-left, as follows:

   **0**—to configure the lower nibble as output

   **1**—to configure the upper nibble as input

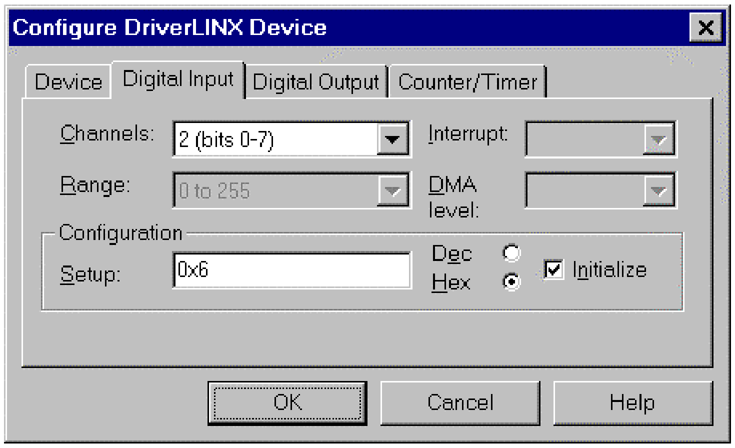


2. Calculate the hexadecimal or decimal value:

```
0x4 + 0x2 + 0x0 = 0x6

4 + 2 + 0 = 6
```

3. Then, enter *0x6* (hexadecimal) or *6* (decimal) for the *Setup* property.


Configure DriverLINX Device

Device | Digital Input | Digital Output | Counter/Timer

Channels: 2 (bits 0-7)

Interrupt:

Range: 0 to 255

DMA level:

Configuration

Setup: 0x6

Dec

[✓ Hex]

[✓ Initialize]

### *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.

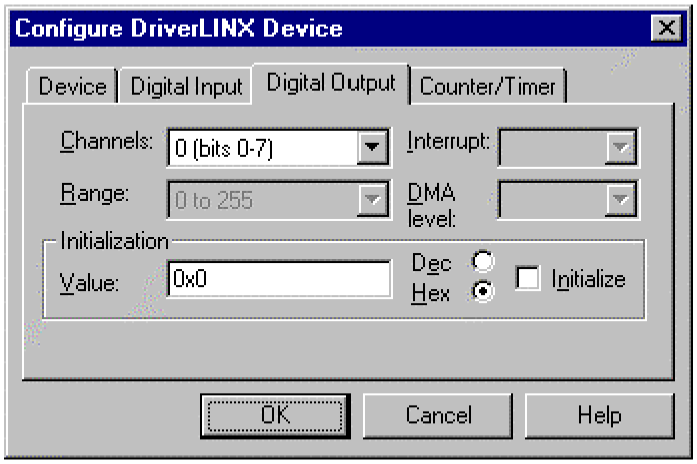
### *Dec*

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



Channels: 0 (bits 0-7)
Interrupt:
Range: 0 to 255
DMA level:
Initialization
Value: 0x0
Dec ○ Hex ●
☐ Initialize

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-PIO models have three or twelve digital input/output ports. The Digital Output subsystem shares these ports with the Digital Input subsystem. The KPCI-PIO requires configuration of each port in the digital input/output ports as input or output. See "Digital Input Subsystem Page" on page 15.

### 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-PIO Series board. DriverLINX disables this property.

### DMA level

The KPCI-PIO 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-PIO, 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*

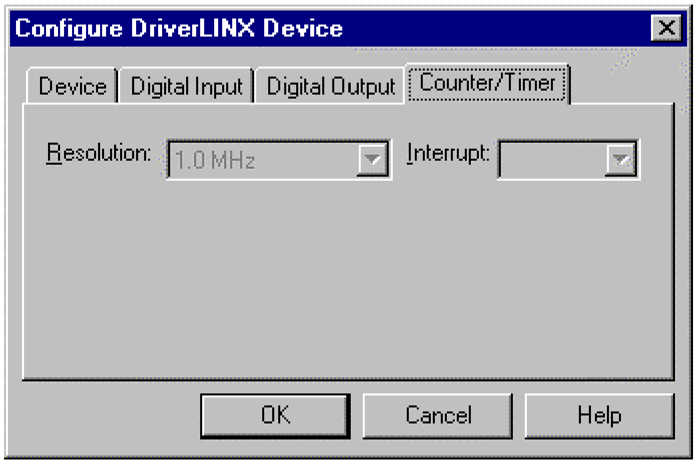
This check box converts the *Initialization Value* property to decimal.

### *Hex*

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

## Counter/Timer Subsystem Page

*For the KPCI-PIO, there are no configurable options on the Counter/Timer subsystem page.*


Configure DriverLINX Device

Resolution: 1.0 MHz

Interrupt:

### Resolution

The *Resolution* property specifies the clock frequency of the master oscillator. DriverLINX provides a System Pacer Clock that simulates a 1 MHz hardware clock for pacing input/output tasks.

### Interrupt

The KPCI-PIO does not support interrupts from counter/timers. DriverLINX disables this property and displays it as blank.

# Programming the KPCI-PIO Series

## Selecting an API

The KPCI-PIO 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.
- **I/O Emulation**—An interface for Win 95/98 only that supports single-tasking, single-threaded access to an I/O driver that emulates the Intel 8255 Programmable Peripheral Interface chip.
- **Direct I/O**—an interface that is operating system independent and supports single-tasking, single-threaded access either to an Intel 8255-like function call interface or to a KPCI-PIO hardware-specific 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 95/98 and Windows NT 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 16- and 32-bit applications on Windows 95/98 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.
- **Nibble configuration—**On models that support it, configurations of Port C with one input nibble and one output nibble require the use of the Configure DriverLINX Device dialog.
- **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 I/O Emulation

The I/O Emulation Interface simulates the hardware behavior of Intel 8255-based products, such as the PIO24 and PIO96, on the KPCI-PIO Series. 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 **inp** and **outp** instructions.
- **Intel 8255 emulation—**This interface simulates the I/O address map and programming protocols of Intel's 8255 chip.
- **MS-DOS window support**—This interface will host 16- and 32-bit console applications running in an MS-DOS window.

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

- **Win 9x only**—The I/O Emulation interface only runs on Windows 95/98. Windows NT does not support direct I/O.
- **Non-exclusive hardware access**—This interface does not synchronize or coordinate hardware access among threads or processes.

- **8-bit I/O only**—The virtual device driver for I/O emulation only supports 8-bit I/O access to the hardware.
- **Speed of single-value I/O**—I/O performance is slower than ISA-based products. The virtual driver uses a hardware exception trap for invalid I/O instructions to redirect the I/O command to the PCI bus.
- **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 or newer KPCI-PIO hardware features not available with Intel 8255-based products.

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

## Choosing Direct I/O

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

- **Operating system independence**—Both Windows 95/98 and Windows NT 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 the fastest access to the hardware registers of the three interfaces.
- **Versatility**—Most Windows compilers and scripting languages support ActiveX Automation objects.
- **Native hardware register access**—For special-purpose applications, this interface also supports product-specific access to the hardware registers.
- **Speed of single-value I/O**—The I/O performance of this interface is the fastest of the three interfaces when using early binding to the interface.

The Direct I/O interface also has disadvantages:

- **32-bit only interface**—16-bit applications cannot use this interface.
- **Non-exclusive hardware access**—This interface does not synchronize or coordinate hardware access among threads or processes.
- **8-bit I/O only**—The Intel 8255-based methods only support 8-bit hardware access.
- **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-PIO programming interface, select one of the following topics for more information.

## Installing DriverLINX

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

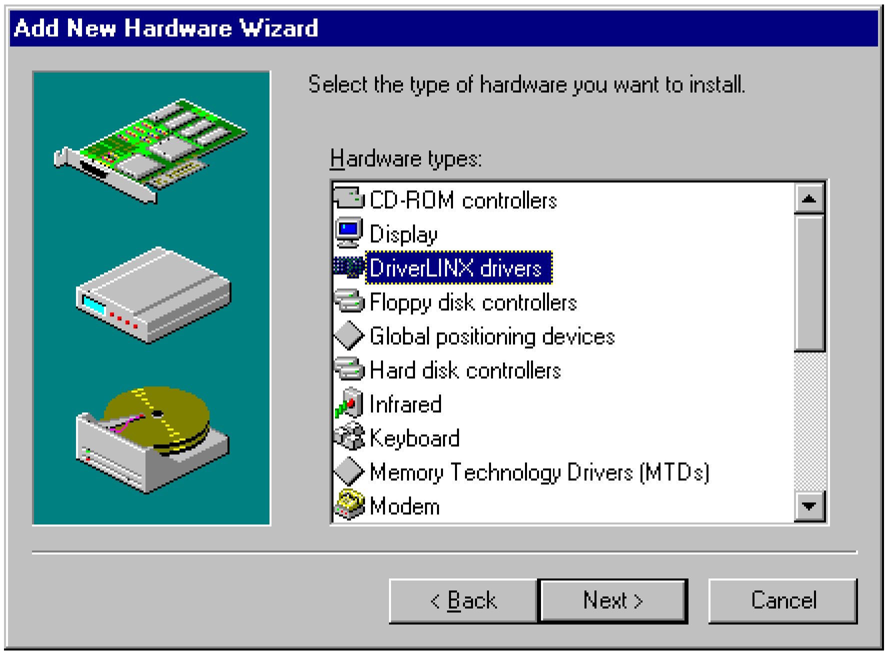
## Installing the I/O Emulation Interface

To use the I/O Emulation interface, you must manually install a Win 95/98 virtual device driver and configure it.

### *Installing the I/O Emulation Driver*

You must be running Win 95/98 and have already installed the DriverLINX drivers from the DriverLINX CD Browser.

1. From the Windows Control Panel, click the Add New Hardware icon.
2. Click Next to begin installing the driver.
3. Select No for the question, “Do you want Windows to search for your new hardware?” and click Next.
4. Select the icon for “DriverLINX drivers” and click Next.

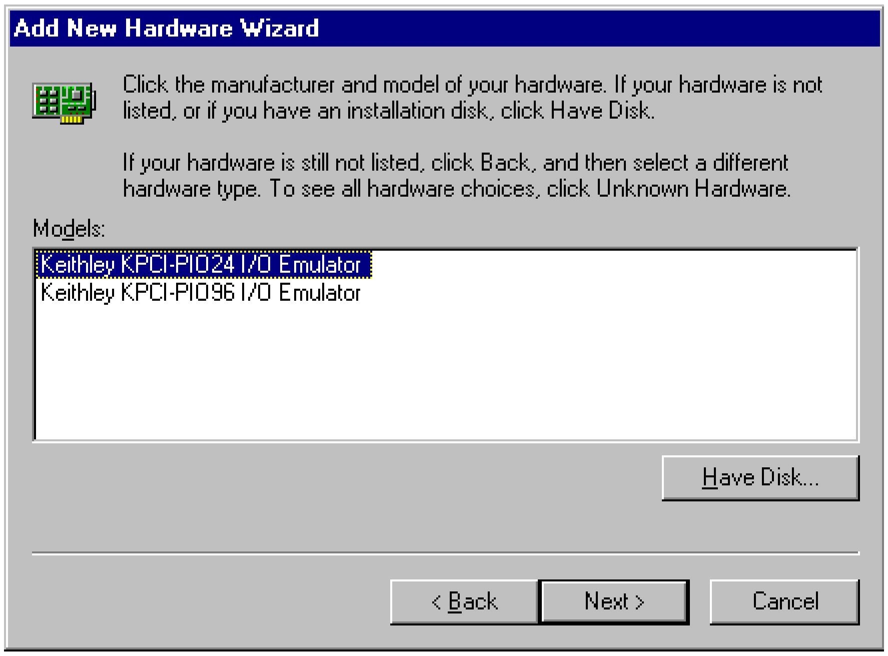

Add New Hardware Wizard

Select the type of hardware you want to install.

Hardware types:

- CD-ROM controllers
- Display
- DriverLINX drivers
- Floppy disk controllers
- Global positioning devices
- Hard disk controllers
- Infrared
- Keyboard
- Memory Technology Drivers (MTDs)
- Modem

5. Select the name of the I/O Emulator you want to install. Your selection must be compatible with your hardware model. Click Next to continue.


Add New Hardware Wizard

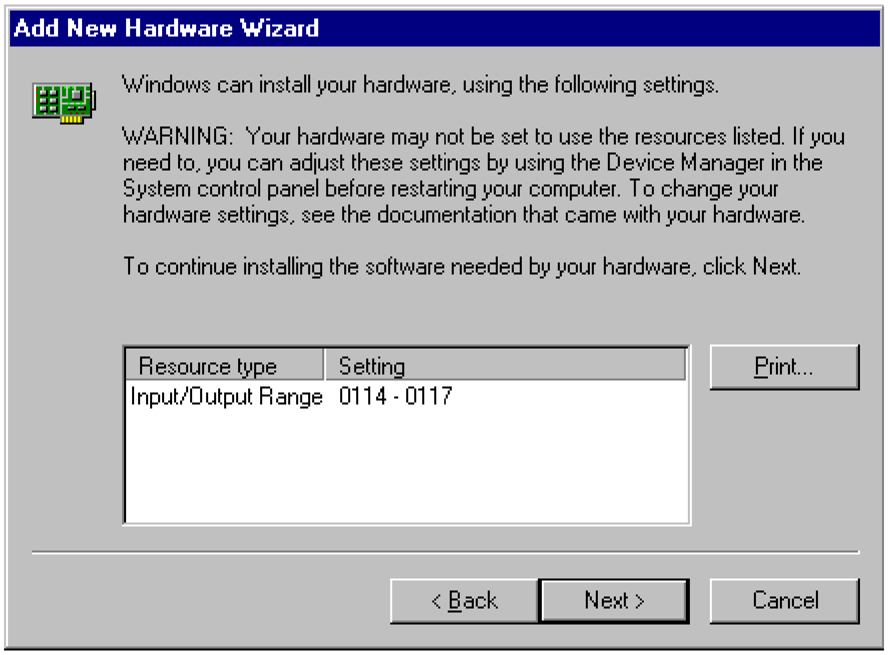
Click the manufacturer and model of your hardware. If your hardware is not listed, or if you have an installation disk, click Have Disk.

If your hardware is still not listed, click Back, and then select a different hardware type. To see all hardware choices, click Unknown Hardware.

Models:

Keithley KPCI-PIO24 I/O Emulator
Keithley KPCI-PIO96 I/O Emulator

6. Windows will select an available I/O address for the emulation driver. Click Next to continue.


Add New Hardware Wizard

Windows can install your hardware, using the following settings.

WARNING: Your hardware may not be set to use the resources listed. If you need to, you can adjust these settings by using the Device Manager in the System control panel before restarting your computer. To change your hardware settings, see the documentation that came with your hardware.

To continue installing the software needed by your hardware, click Next.

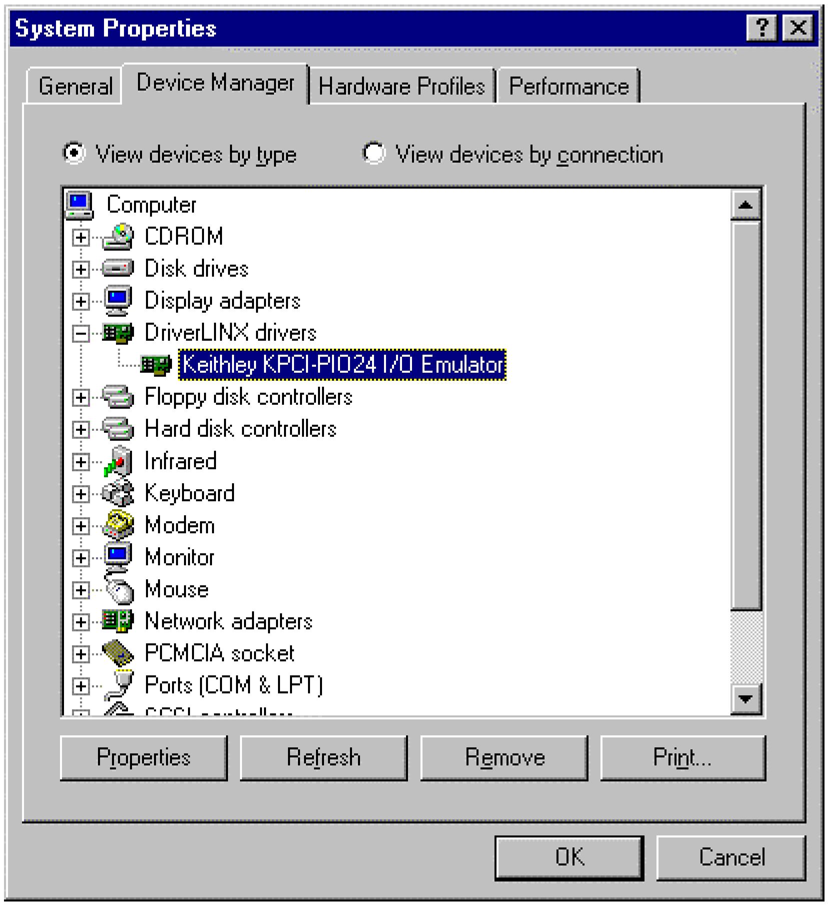
| Resource type | Setting |
|---|---|
| Input/Output Range | 0114 - 0117 |

7. Click Finish to close the wizard. Windows will load the I/O Emulation driver the next time you reboot Windows.

## Configuring the I/O Emulation Driver

Windows should automatically select an available I/O address for the I/O Emulation Driver. If you wish to check or modify 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.

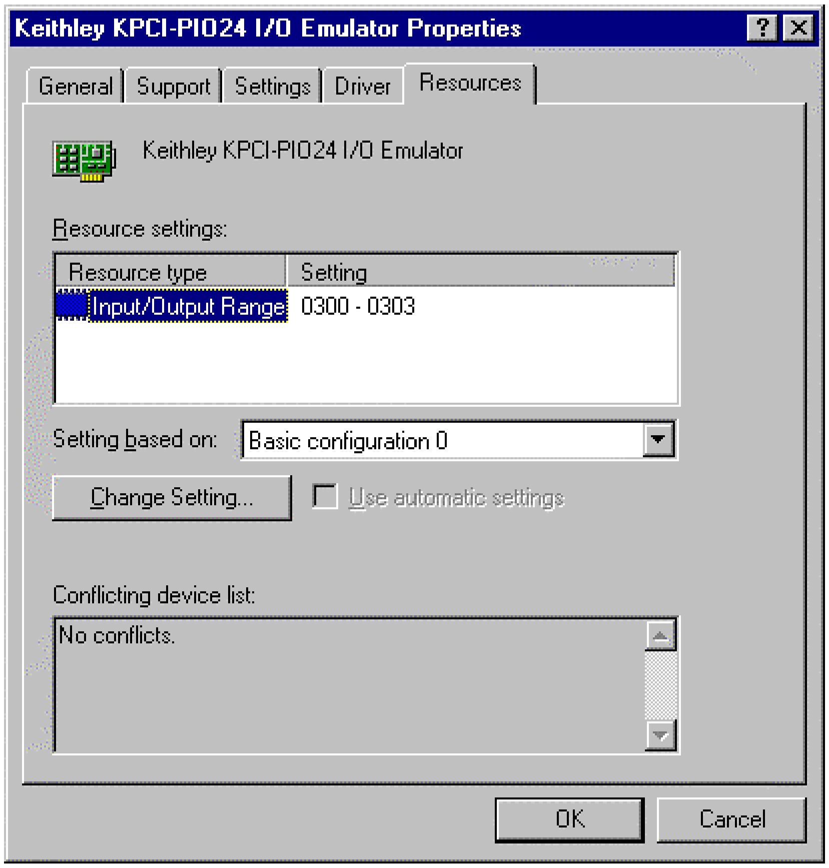

System Properties

General | Device Manager | Hardware Profiles | Performance

View devices by type [✓] · View devices by connection

Computer
- CDROM
- Disk drives
- Display adapters
- DriverLINX drivers
  - Keithley KPCI-PIO24 I/O Emulator
- Floppy disk controllers
- Hard disk controllers
- Infrared
- Keyboard
- Modem
- Monitor
- Mouse
- Network adapters
- PCMCIA socket
- Ports (COM & LPT)

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



Keithley KPCI-PIO24 I/O Emulator Properties

Keithley KPCI-PIO24 I/O Emulator

Resource settings:

| Resource type | Setting |
|---|---|
| Input/Output Range | 0300 - 0303 |

Setting based on: Basic configuration 0

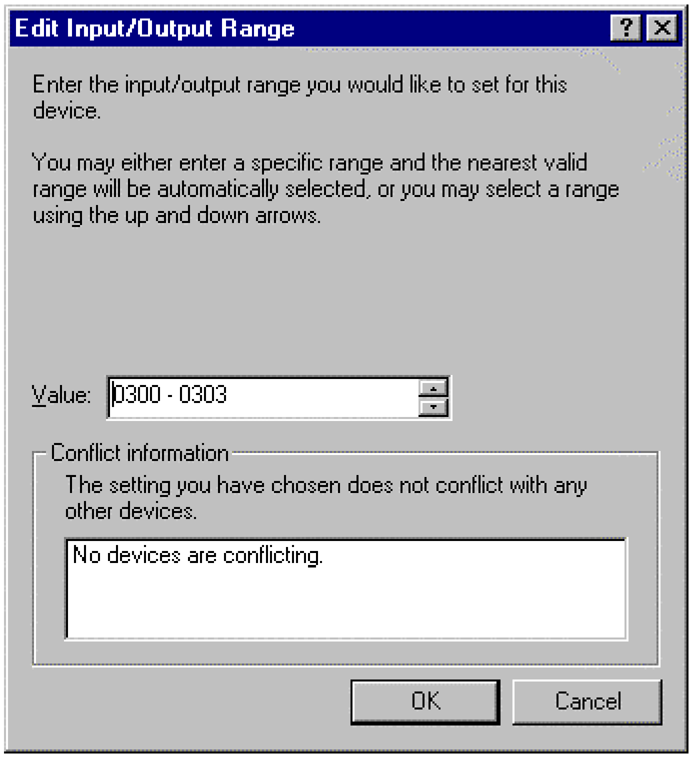
Use automatic settings [ ]

Conflicting device list:

No conflicts.

5. Select Input/Output Range and click the "Change Setting…" button.

6. Enter a new input/output range for the device. Click OK to accept the changes.


Edit Input/Output Range

Enter the input/output range you would like to set for this device.

You may either enter a specific range and the nearest valid range will be automatically selected, or you may select a range using the up and down arrows.

Value: 0300 - 0303

Conflict information

The setting you have chosen does not conflict with any other devices.

No devices are conflicting.

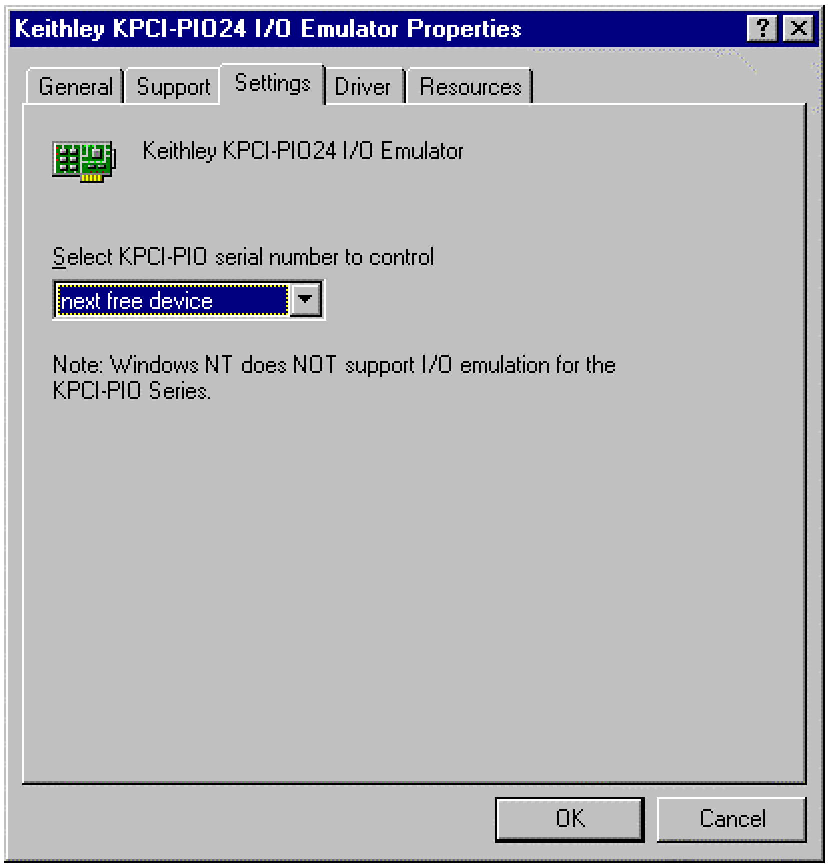
7. Click OK to close the Device Manager. Windows will make the changes the next time you reboot.

## *Assigning the I/O Emulator to a KPCI-PIO Board*

Normally Windows assigns the I/O emulation address to the next available KPCI-PIO device it finds during scanning the PCI bus. If you have more than one device installed, changing a card's bus slot may change the I/O address that Windows previously assigned to this device. To prevent this, you can configure the I/O Emulation driver to always attach to a device with a given serial number.

1. From Windows Device Manager, select the driver you wish to configure and click the "Properties" button

2. Click the Settings tab to select the KPCI-PIO device for the driver.



Keithley KPCI-PIO24 I/O Emulator

Select KPCI-PIO serial number to control

next free device

Note: Windows NT does NOT support I/O emulation for the KPCI-PIO Series.

3. To assign the driver's I/O address to a specific KPCI-PIO board, select the board's serial number in the drop-down list. Note that this list only displays currently installed boards.

4. To assign the driver to the next available board, select "next free device" in the drop-down list.

5. Click OK to close the Emulator Properties.

6. Windows will make the requested I/O address assignment the next time you reboot Windows.

## Installing the 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 Direct I/O Driver*

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

This driver requires that you install the DriverLINX kernel drivers as it uses their services. On Win NT, the kernel driver is KPCIPIO.SYS. On Win 95/98, it is KPCIPIO.VXD. Follow the normal DriverLINX installation and configuration procedures to install these kernel drivers.

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

The Direct I/O Driver requires that you configure a DriverLINX Logical Device using the DriverLINX Configuration Panel utility. The Direct I/O driver does not require any separate configuration.

# Programming an API

To use a KPCI-PIO 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 I/O Emulation Interface

This interface uses only the x86 CPU instructions, **inp** and **outp**, for 8-bit I/O. See “Configuring the I/O Emulation Driver” on page 27 to determine the I/O address range for the virtual Intel 8255 chip(s).

Each Intel 8255 chip consists of three 8-bit I/O ports and an 8-bit control port. The three data ports are called Port A, B, and C and they reside at base address + 0, 1, 2, respectively. The PIO24 has one 8255-type port while the PIO96 has four 8255-type ports.

The Intel 8255 control port assigns functions to each bit as follows:

| **Function** | **Bit Offset** |
|---|---|
| Port C, lower: 1 = input, 0 = output. | 0 |
| Port B: 1 = input, 0 = output. | 1 |
| Group B mode select: 0 = mode 0, 1 = mode 1 | 2 |
| Port C, upper: 1 = input, 0 = output | 3 |
| Port A: 1 = input, 0 = output | 4 |
| Group A mode select: 00 = mode 0, 01 = mode 1, 1x = mode 2 | 5,6 |
| Mode set flag, 1 = active | 7 |

The KPCI-PIO hardware does not emulate exactly the Intel 8255 design. The KPCI-PIO96 only supports mode 0 (basic input/output with transparent input buffers); Bits 2, 5, and 6 must be 0 for the KPCI-PIO96.

The KPCI-PIO24 supports a modified version of mode 1 (strobed input/output)—latched input buffers using the Interrupt Input line as the read buffer latch instead using Port C lines as handshaking signals. The I/O Emulation driver enables KPCI-PIO24 read buffer latching when either the Group A or Group B mode select bits are one.

For more information on programming an Intel 8255-based digital I/O device, consult either an Intel hardware manual or the programming section of an 8255-based Keithley hardware product.

Keithley does not recommend this interface for new applications.

## Programming the Direct I/O Interface

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

The methods of the control's default interface, IKPCIPIO, emulate the 8-bit port I/O behavior of an Intel 8255 chip. Consult an Intel hardware manual or the programming section of an 8255-based Keithley hardware product for how to program an 8255 chip.

The methods of the control's alternate interface, IHardware, support 8-, 16-, and 32-bit access to the hardware register map of the KPCI-PIO Series. Consult the KPCI-PIO hardware manual for the layout of the register map. Note that this interface is highly hardware-dependent and will most likely make your application incompatible with other Keithley digital I/O hardware products.

### *IKPCIPIO Intel 8255 Interface*

The following syntax descriptions are shown in C/C++. For Visual Basic, use the Object Browser to see the VB syntax. See "Using the Direct I/O Driver in Visual Basic" on page 34.

- HRESULT OpenDevice(long Device);
  Opens a KPCI-PIO device.
  Device is the DriverLINX Logical Device number of the KPCI-PIO board to open.
- HRESULT CloseDevice();
  Closes a previously opened KPCI-PIO 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 KPCI-PIO 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 KPCI-PIO 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.

## *KPCI-PIO IHardware Interface*

The following syntax descriptions are shown in C. For Visual Basic, use the Object Browser to see the VB syntax. See “Using the Direct I/O Driver in Visual Basic” on page 34.

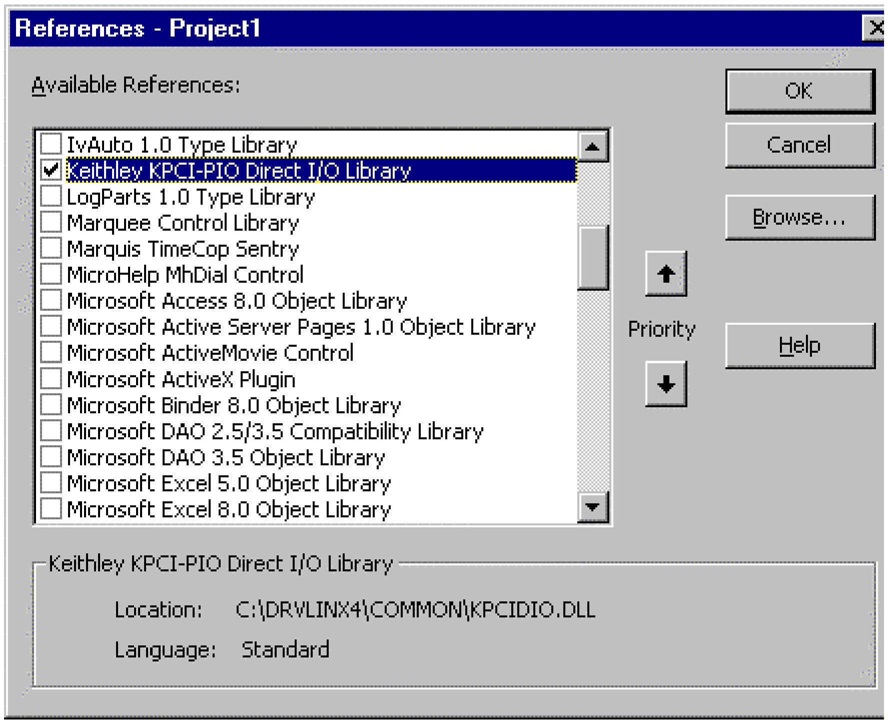
Before using this interface, the client must open a KPCI-PIO device using OpenDevice in the IKPCIPIO interface.

- HRESULT ReadRegister8(short Offset, unsigned char * Result);
  Reads an 8-bit value from the KPCI-PIO board.
  Offset is the value of the register to read relative the base address.
  Result is the value read from the register.
- HRESULT ReadRegister16(short Offset, short * Result);
  Reads a 16-bit value from the KPCI-PIO board.
  Offset is the value of the register to read relative the base address.
  Result is the value read from the register.
- HRESULT ReadRegister32(short Offset, long * Result);
  Reads a 32-bit value from the KPCI-PIO board.
  Offset is the value of the register to read relative the base address.
  Result is the value read from the register.
- HRESULT WriteRegister8(short Offset, unsigned char Value);
  Writes an 8-bit value to the KPCI-PIO board.
  Offset is the value of the register to write relative the base address.
  Value is the output to write to the register.
- HRESULT WriteRegister16(short Offset, short Value);
  Writes a 16-bit value to the KPCI-PIO board.
  Offset is the value of the register to write relative the base address.
  Value is the output to write to the register.
- HRESULT WriteRegister32(short Offset, long Value);
  Writes a 32-bit value to the KPCI-PIO board.
  Offset is the value of the register to write relative the base address.
  Value is the output to write to the register.

## *Using the 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 KPCI-PIO Direct I/O Library”.


References - Project1

Available References:

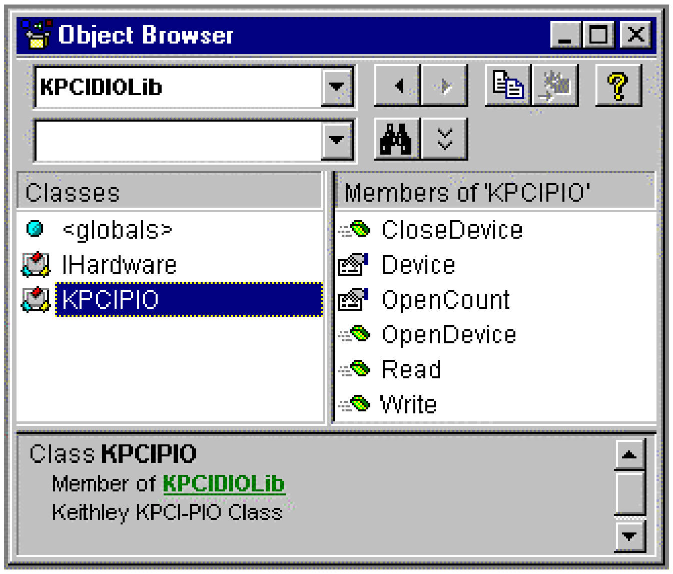
- [ ] IvAuto 1.0 Type Library
- [✓] Keithley KPCI-PIO Direct I/O Library
- [ ] LogParts 1.0 Type Library
- [ ] Marquee Control Library
- [ ] Marquis TimeCop Sentry
- [ ] MicroHelp MhDial Control
- [ ] Microsoft Access 8.0 Object Library
- [ ] Microsoft Active Server Pages 1.0 Object Library
- [ ] Microsoft ActiveMovie Control
- [ ] Microsoft ActiveX Plugin
- [ ] Microsoft Binder 8.0 Object Library
- [ ] Microsoft DAO 2.5/3.5 Compatibility Library
- [ ] Microsoft DAO 3.5 Object Library
- [ ] Microsoft Excel 5.0 Object Library
- [ ] Microsoft Excel 8.0 Object Library

Keithley KPCI-PIO Direct I/O Library

Location: C:\DRVLINX4\COMMON\KPCIDIO.DLL

Language: Standard

3. Check “Keithley KPCI-PIO Direct 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 “KPCIDIOLib”.


Object Browser

KPCIDIOLib

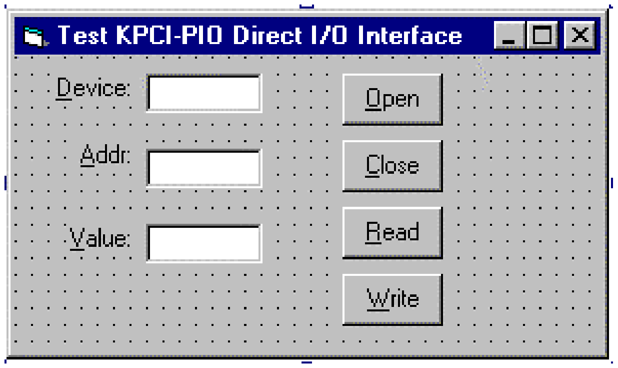
| Classes | Members of 'KPCIPIO' |
|---|---|
| <globals> | CloseDevice |
| IHardware | Device |
| KPCIPIO | OpenCount |
| | OpenDevice |
| | Read |
| | Write |

Class **KPCIPIO**
Member of KPCIDIOLib
Keithley KPCI-PIO Class

6. The Object Browser shows the VB syntax for the KPCIPIO (see “IKPCIPIO Intel 8255 Interface” on page 33) and IHardware (see “KPCI-PIO IHardware Interface” on page 34) classes.

## *Creating a Simple Visual Basic Application*

1. Create a Visual Basic form:



2. Add the following source code to declare, open, close, read, and write a KPCI-PIO device.

```
Option Explicit
Dim KPIO As New KPCIPIO

Private Sub btClose_Click()
  KPIO.CloseDevice
End Sub

Private Sub btOpen_Click()
  KPIO.OpenDevice (Val(tbDevice))
End Sub

Private Sub btRead_Click()
  tbValue = Hex(KPIO.Read(Val(tbAddr)))
End Sub

Private Sub btWrite_Click()
  Dim v As Byte

  v = Val(tbValue)
  KPIO.Write Val(tbAddr), v
End Sub
```

# Accessing KPCI-PIO 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-PIO Subsystem chapters, which follow, explain the implementation of DriverLINX's hardware model for the KPCI-PIO Series. This chapter is for engineers who are more familiar with the board's features than DriverLINX's model. It surveys the Keithley KPCI-PIO24 and KPCI-PIO96 hardware features to show how they fit into DriverLINX's hardware model.

This chapter also discusses the differences between the KPCI-PIO24/96 boards and the ISA PIO24/96 boards they may replace. Although, the two series have very similar capabilities, they differ in the level of emulation of the Intel 8255 Programmable Peripheral Interface chip. In addition, direct access techniques—important to non-DriverLINX applications—differ between the PCI bus series and the ISA bus series. Consideration of these issues may be important when attempting to port applications between boards.

# KPCI-PIO Hardware Features

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

- Data Lines
- Control Lines on the KPCI-PIO24
- Input Latching on the KPCI-PIO24

### Data Lines

| Feature | Description | Reference/Notes |
|---|---|---|
| PA0-7<br>PB0-7<br>PC0-7 | The KPCI-PIO24 has one group of data lines. The KPCI-PIO96 has four. Each group consists of three 8-bit ports: A, B and C. DriverLINX numbers the ports consecutively, starting with 0. | "Connecting Signals to the KPCI-PIO" on page 47. |
| | DriverLINX also allows access to groups of 1, 2, 4, 8, 16 or 32 lines in a single or consecutive ports. | "Extended Logical Channel Addressing" on page 67. |

### Control Lines on the KPCI-PIO24

| 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.:<br>• Timing signal —"Digital Input Timing Events" on page 56.<br>• Start trigger —"Digital Input Start Events" on page 60.<br>• Stop trigger—"Digital Input Stop Events" on page 61. |
| INT_EN | Interrupt Enable—the INT_EN signal is an active-low input that enables the INT_REQ line. | Because software cannot sense or control the INT_EN signal, it has no representation in DriverLINX's hardware model. |

### Input Latching on the KPCI-PIO24

| Feature | Description | Reference/Notes |
|---|---|---|
| INT LATCH | The KPCI-PIO24 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:<br>• ActiveX Control—"Service Request Control - Request Group" in the *DriverLINX/VB Technical Reference Manual.*<br>• C/C++—"Service Requests - Request Group" in the *DriverLINX Technical Reference Manual.* |
| | • Polled Mode<br>Port configuration:<br>• Basic mode (Mode 0)—**unlatched input**<br>• Strobed or Strobed_NE mode (Mode 1)—**latched input** | Configure the board using one of the following methods:<br>• Static configuration using the DriverLINX Configuration Panel— "Configuring the KPCI-PIO" on page 11.<br>• Dynamic configuration using a *Configure* Service Request—"Configuring a Digital Port at Run Time" in the *DriverLINX Digital I/O Programming Guide.* |
| | • Interrupt Mode<br>Clock source:<br>• INTERNAL (system timer)—**unlatched mode**<br>• EXTERNAL, EXTERNAL+, EXTERNAL- —**latched mode** | Set the Rate Event's clock source property in the Service Request:<br>• ActiveX Control—"Service Request Control - Events Group - Rate Events" in the *DriverLINX/VB Technical Reference Manual.*<br>• C/C++—"Service Requests - Events Group - Rate Events" in the *DriverLINX Technical Reference Manual.* |

## Differences Between the PCI and ISA Models

This section lists the differences between the KPCI-PIO Series and the ISA PIO Series models:

- INT_REQ on the KPCI-PIO24 and ISA PIO24
- Bus Interface
- Intel 8255 Mode Emulation

### INT_REQ on the KPCI-PIO24 and ISA PIO24

| Feature | Description | Reference/Notes |
|---|---|---|
| INT POLARITY | The KPCI-PIO24 support rising or falling edge interrupts on its INT_REQ line. The ISA PIO24 | On the KPCI-PIO24, the rising edge is the default active edge. Applications can change the interrupt |

|  |  |  |
|---|---|---|
|  | allows only rising edge interrupts. | polarity using a Dio Setup event or by choosing between External+ and External- clock source in a timing event. |
| INT LATCH | The KPCI-PIO24 can optionally latch the input buffers when it receives an INT_REQ signal.<br>On the ISA PIO24 cannot latch the input buffers but only triggers an interrupt. | This added feature on the KPCI-PIO24 allows it to perform strobed input. See Intel 8255 Mode Emulation—Mode 1, below. |

### Bus Interface

| Feature | Description | Reference/Notes |
|---|---|---|
| PCI<br>ISA | Keithley’s KPCI-PIO models have a PCI bus interface, while the PIO models have an ISA bus interface.<br>The PCI bus maps the board’s registers to memory addresses. The ISA bus maps the board’s registers to I/O ports.<br>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.<br>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. |
| Mode 1 | Strobed I/O—The KPCI-PIO24 emulates strobed mode through input buffer latching with its INT_REQ line. The ISA PIO96 also supports this mode. | Static configuration using the DriverLINX Configuration Panel— “Configuring the KPCI-PIO” on page 11. |
| Mode 2 | Bidirectional I/O—Only the ISA PIO96 allows this mode. | Dynamic configuration using a *Configure* Service Request—“Configuring a Digital Port at Run Time” in the *DriverLINX Digital I/O Programming Guide*. |

# Using DriverLINX with the KPCI-PIO Series

## Introduction

This chapter describes how to set up and use KPCI-PIO 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 specifies 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-PIO

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-PIO hardware capabilities.

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

### DriverLINX Subsystems

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

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

2. **Digital Input**—refers to the 8-bit digital input/output ports as well as the 1-bit digital input (TTL) control signal, INT_REQ.
3. **Digital Output**—refers to the 8-bit digital input/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-PIO 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-PIO.

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

*KPCI-PIO 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-PIO. 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** | | |
|---|---|---|---|---|
| Mode | | Timing | Start | Stop |
| **Digital Input** | | | | |
| Polled | Start | null | null, cmd, dig* | null, cmd, TC |
| Interrupt | Start, Stop, Status | dig**, rate | cmd, dig | cmd, TC, dig |
| Other | Initialize, Config | diosetup | | |
| **Digital Output** | | | | |
| Polled | Start | null | null, cmd, dig* | null, cmd, TC |
| Interrupt | Start, Stop, Status | dig**, rate | cmd, dig | cmd, TC, dig |
| Other | Initialize, Config | diosetup | | |
| **Counter/Timer** | | | | |
| Other | Initialize | | | |
| **Device** | | | | |
| Other | Initialize, Configure, Capabilities | | | |

* In polled mode a digital start event sets up bit-masked input/output.
** Only the KPCI-PIO24 support digital timing.

*Allowed Operations and Events for KPCI-PIO 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-PIO

The Keithley hardware manual describes the data and control signals for the KPCI-PIO and the connector pinouts for these signals. 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 Digital Input subsystem has three or twelve 8-bit digital input/output ports and one control input which DriverLINX models as a 1-bit logical digital input port (KPCI-PIO24 only). DriverLINX maps these signals to Logical Channels as shown in the following tables.

**KPCI-PIO24**

| Logical Channel | DriverLINX Function | KPCI-PIO External Connector |
|---|---|---|
| 0 | Digital Input/Output Port | Digital input/output lines PA0 - PA7 |
| 1 | Digital Input/Output Port | Digital input/output lines PB0 - PB7 |
| 2 | Digital Input/Output Port | Digital input/output lines PC0 - PC7 |
| 3 or ExtClk | External Clock | INT_REQ |
| 4 or ExtTrg | External Trigger | INT_REQ |

**KPCI-PIO96**

| Logical Channel | DriverLINX Function | KPCI-PIO External Connector |
|---|---|---|
| 0 | Digital Input/Output Port | Digital input/output lines Group 0 PA0 - PA7 |
| 1 | Digital Input/Output Port | Digital input/output lines Group 0 PB0 - PB7 |
| 2 | Digital Input/Output Port | Digital input/output lines Group 0 PC0 - PC7 |
| 3 | Digital Input/Output Port | Digital input/output lines Group 1 PA0 - PA7 |
| 4 | Digital Input/Output Port | Digital input/output lines Group 1 PB0 - PB7 |
| 5 | Digital Input/Output Port | Digital input/output lines Group 1 PC0 - PC7 |
| 6 | Digital Input/Output Port | Digital input/output lines Group 2 PA0 - PA7 |
| 7 | Digital Input/Output Port | Digital input/output lines Group 2 PB0 - PB7 |
| 8 | Digital Input/Output Port | Digital input/output lines Group 2 PC0 - PC7 |
| 9 | Digital Input/Output Port | Digital input/output lines |

| | | |
|---|---|---|
| | | Group 3 PA0 - PA7 |
| 10 | Digital Input/Output Port | Digital input/output lines Group 3 PB0 - PB7 |
| 11 | Digital Input/Output Port | Digital input/output lines Group 3 PC0 - PC7 |

**Note:** To use a line in an 8-bit digital input/output port for input, first configure it for input on the Digital Input Subsystem Page of the *DriverLINX Configuration Panel*. See “Configuring the KPCI-PIO” on page 11.

The Digital Input subsystem shares the System Pacer Clock with the Digital Output subsystem. DriverLINX designates the System Pacer Clock as Logical Channel 0 of the Counter/Timer subsystem.

## 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-PIO24**

| Logical Channel | DriverLINX Function | KPCI-PIO External Connector |
|---|---|---|
| 0 | Digital Input/Output Port | Digital input/output lines PA0 - PA7 |
| 1 | Digital Input/Output Port | Digital input/output lines PB0 - PB7 |
| 2 | Digital Input/Output Port | Digital input/output lines PC0 - PC7 |

**KPCI-PIO96**

| Logical Channel | DriverLINX Function | KPCI-PIO External Connector |
|---|---|---|
| 0 | Digital Input/Output Port | Digital input/output lines Group 0 PA0 - PA7 |
| 1 | Digital Input/Output Port | Digital input/output lines Group 0 PB0 - PB7 |
| 2 | Digital Input/Output Port | Digital input/output lines Group 0 PC0 - PC7 |
| 3 | Digital Input/Output Port | Digital input/output lines Group 1 PA0 - PA7 |
| 4 | Digital Input/Output Port | Digital input/output lines Group 1 PB0 - PB7 |
| 5 | Digital Input/Output Port | Digital input/output lines Group 1 PC0 - PC7 |
| 6 | Digital Input/Output Port | Digital input/output lines Group 2 PA0 - PA7 |
| 7 | Digital Input/Output Port | Digital input/output lines Group 2 PB0 - PB7 |
| 8 | Digital Input/Output Port | Digital input/output lines Group 2 PC0 - PC7 |

| | | |
|---|---|---|
| 9 | Digital Input/Output Port | Digital input/output lines Group 3 PA0 - PA7 |
| 10 | Digital Input/Output Port | Digital input/output lines Group 3 PB0 - PB7 |
| 11 | Digital Input/Output Port | Digital input/output lines Group 3 PC0 - PC7 |

**Note:** To use a line in an 8-bit digital input/output port for output, first configure it for output on the Digital Input Subsystem Page of the *DriverLINX Configuration Panel*. See “Configuring the KPCI-PIO” on page 11.

The Digital Output subsystem shares the System Pacer Clock with the Digital Input subsystem. DriverLINX designates the System Pacer Clock as Logical Channel 0 of the Counter/Timer subsystem.

## Counter/Timer Subsystem Signals

DriverLINX provides a system timer using software to pace digital input/output. DriverLINX maps the timer to Logical  Channel 0.

| Timer | Connector Name | Logical Channels |
|---|---|---|
| System timer | none | 0 |

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

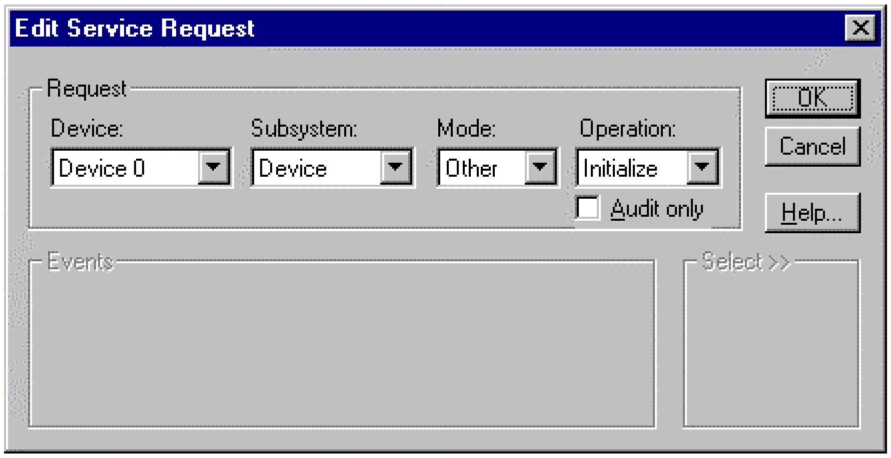
Applications cannot use this timer independently from tasks on an input/output subsystem.

# Using the Device Subsystem

## Introduction

This chapter describes how DriverLINX implements Device Subsystem features for the KPCI-PIO 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-PIO 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*



Device: Device 0
Subsystem: Device
Mode: Other
Operation: Initialize
Audit only

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-PIO 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-PIO 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.*

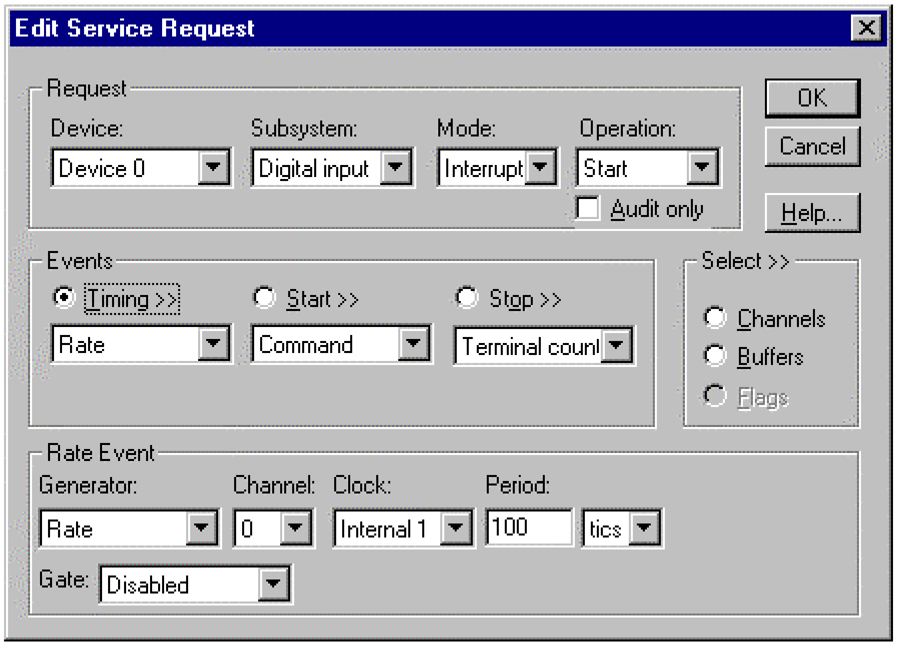
- **Initialize**—DriverLINX aborts all data-acquisition tasks for every subsystem controlled by the current process. DriverLINX then performs an initialization for each supported subsystem.
- **Configure**—DriverLINX displays the *Configure DriverLINX Device* dialog for the current Logical Device. Use 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-PIO 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-PIO 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*


Edit Service Request

Request
Device: Device 0
Subsystem: Digital input
Mode: Interrupt
Operation: Start
☐ Audit only

Events
[✓ Timing >>] Rate
Start >> Command
Stop >> Terminal count

Select >>
Channels
Buffers
Flags

Rate Event
Generator: Rate
Channel: 0
Clock: Internal 1
Period: 100 tics
Gate: Disabled

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-PIO 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-PIO 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-PIO Series.

## Digital Port Configuration

A KPCI-PIO board has three or twelve digital ports that DriverLINX shares between Digital Input and Digital Output Subsystems. DriverLINX provides two methods to configure them as input or output:

- Users can statically configure ports, including the upper and lower nibbles of the C ports, using the *DriverLINX Configuration Panel*. (See “Configuring the KPCI-PIO” on page 11.)
- The KPCI-PIO Series also supports dynamic configuration, enabling applications to configure ports, with DIO Setup Events. (To configure the C ports as nibbles use the *DriverLINX Configuration Panel*.)

### *Dynamically Configuring KPCI-PIO Digital Ports*

When the KPCI-PIO 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 the driver reloads during the next boot.

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-PIO24.
- All inputs on the KPCI-PIO24 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<br>DIO_STROBED or<br>DIO_STROBED_NE |

- Specify the channel you want to reconfigure as the Logical Channel in the Digital Setup Event. (To configure the C ports as nibbles use the *DriverLINX Configuration Panel*.)
- 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-PIO reads the next digital input sample from the port.

The KPCI-PIO 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-PIO supports only fixed rate digital input 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 acquisition of each sample. Digital timing is available only on the KPCI-PIO24.

### None or Null Event

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

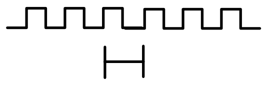
### Rate Event

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

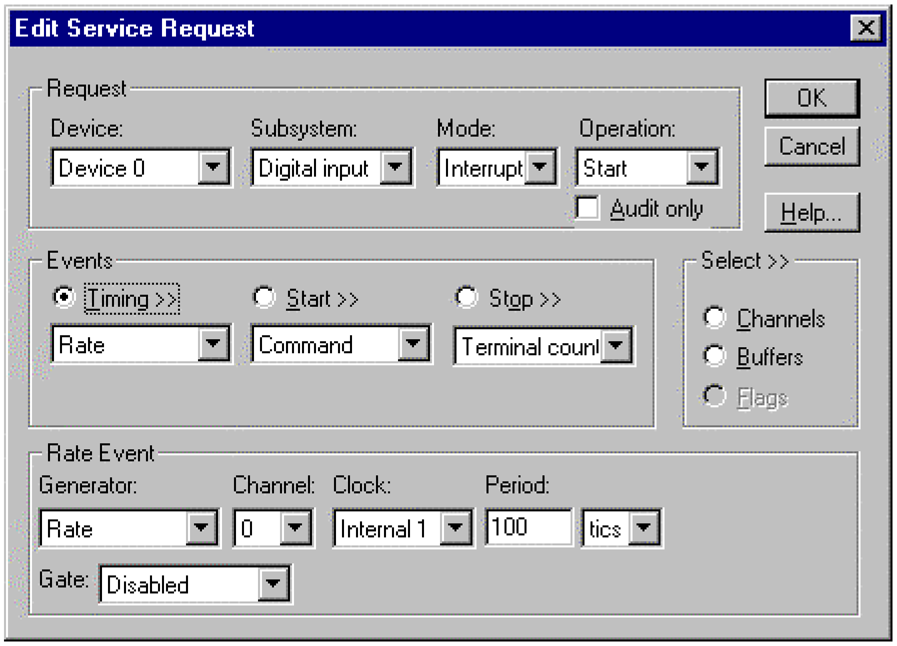
- **Rate Generator**—Generates a fixed rate clock with equal time intervals between tics.

## Rate Generator: Internal Clocking

An internally clocked Rate Generator produces a fixed rate clock with equal time intervals between tics.



Since the KPCI-PIO does not support a dedicated clock for digital input timing, DriverLINX uses the operating system timing clock for pacing digital input. The system timing clock does not have the resolution or time granularity of a dedicated clock, but it is an excellent substitute for low frequency digital input polling operations.



Request
Device: Device 0
Subsystem: Digital input
Mode: Interrupt
Operation: Start
Audit only [ ]

Events
[✓ Timing >>] Start >> Stop >>
Rate | Command | Terminal count

Select >>
Channels
Buffers
Flags

Rate Event
Generator: Rate
Channel: 0
Clock: Internal 1
Period: 100 tics
Gate: Disabled

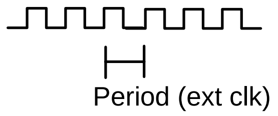
*How to set up the KPCI-PIO for fixed rate sampling using an internal clock.*

*For hardware independence, specify the clock channel using the symbolic constant, DEFAULTTIMER, which always maps to the default Logical Channel for digital input timing.*

- Specify internal clocking using a **Rate** *Generator* on *Logical Channel* **0** with the **Internal 1** *Clock* source.
- The *Period* property specifies the time interval between samples in tics, where a system timer tic is 1 µs, or 1 MHz. The minimum period is 1000 tics, or 1 kHz. The maximum period is 4,294,967,295 tics ($2^{32}-1$), or 0.0002 Hz. The resolution and granularity of the system timer are operating system dependent. Under current versions of Windows, throughput is less than 100 Hz. See “Using the Counter/Timer Subsystem” on page 88 for details on the system timer.

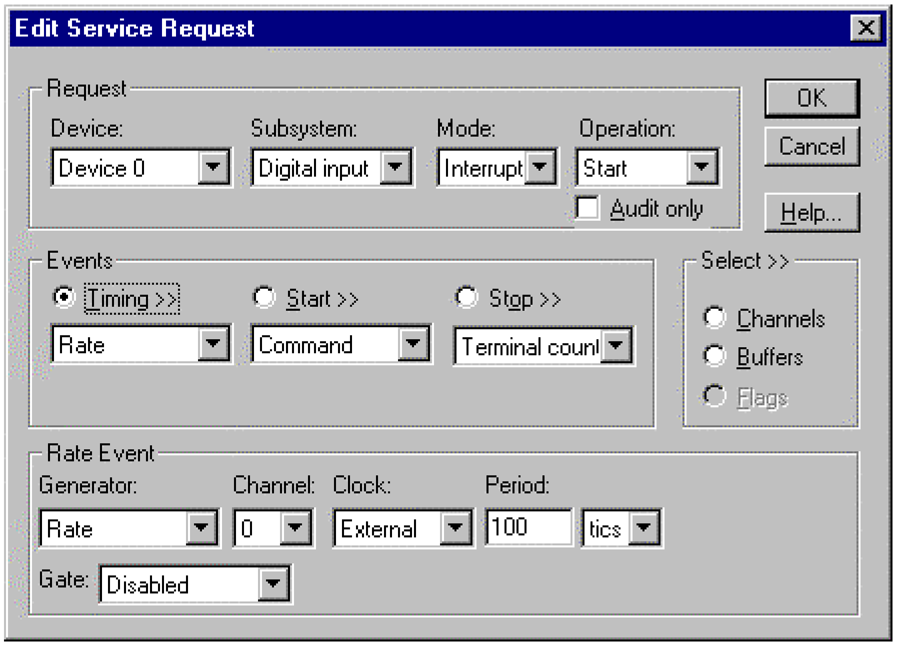
## Rate Generator: External Clocking

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



*Only the KPCI-PIO24 supports external clocking.*

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



Request
Device: Device 0
Subsystem: Digital input
Mode: Interrupt
Operation: Start
[ ] Audit only

Events
(●) Timing >> Rate
( ) Start >> Command
( ) Stop >> Terminal count

Select >>
( ) Channels
( ) Buffers
( ) Flags

Rate Event
Generator: Rate
Channel: 0
Clock: External
Period: 100 tics
Gate: Disabled

*How to set up the KPCI-PIO24 for fixed rate sampling using an external clock.*

*A task cannot use the INT_REQ line (Digital Input Logical Channel 4) for a digital start or stop event while using it as an external clock source.*

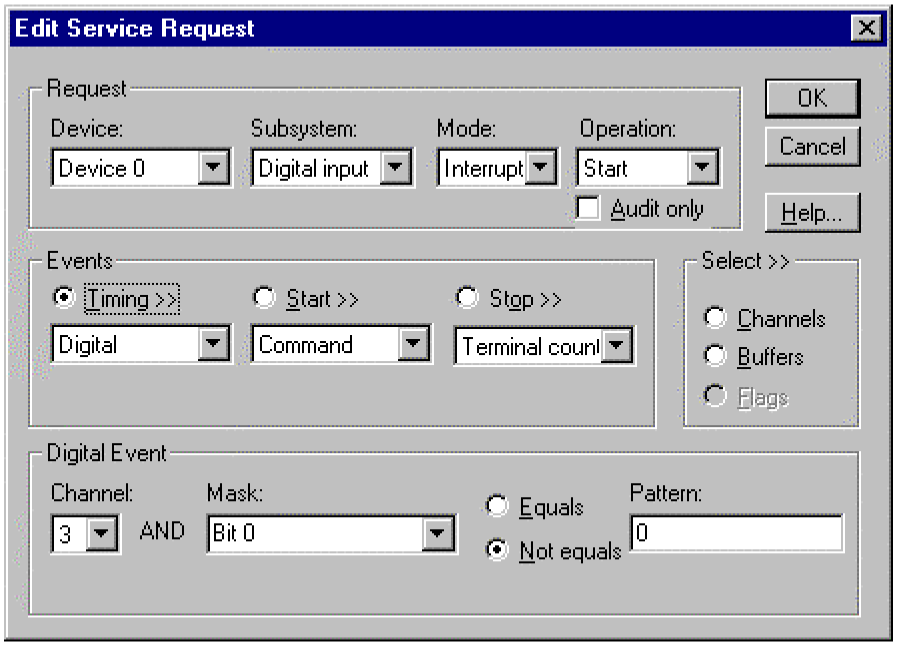
***BE SURE that the external clock source is TTL compatible, 0 V minimum to +5 V maximum!***

- 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

*Only the KPCI-PIO24 supports external clocking.*

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.


Edit Service Request

Request

Device: Device 0

Subsystem: Digital input

Mode: Interrupt

Operation: Start

[ ] Audit only

Events

[✓ Timing >>] Digital

Start >> Command

Stop >> Terminal count

Select >>: Channels, Buffers, Flags

Digital Event

Channel: 3 AND Mask: Bit 0

Equals [✓ Not equals]

Pattern: 0

*How to set up the KPCI-PIO24 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.*

***BE SURE that the external clock source is TTL compatible, 0 V minimum to +5 V maximum!***

- Specify external clocking using *Logical Channel* **3**. 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 (≠0), or **1** for a falling, or negative, edge clock (≠1).

## Digital Input Start Events

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

The KPCI-PIO 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-PIO hardware for the task.
- **Digital**—The KPCI-PIO 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.

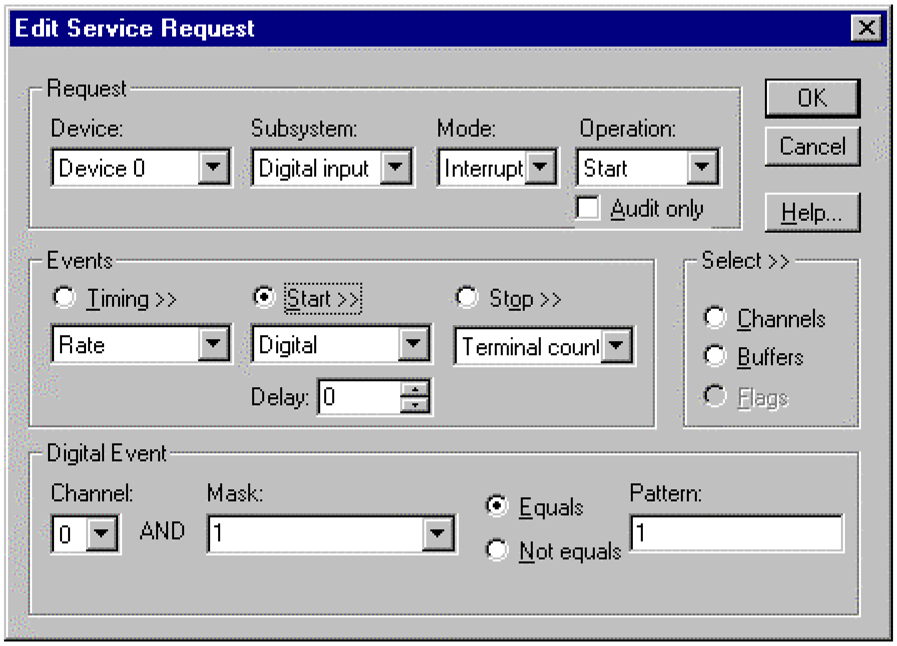
### Command Event

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

### Post-Trigger Sampling with a Digital Event

The KPCI-PIO 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, only the KPCI-PIO24 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.


Edit Service Request

Request
Device: Device 0
Subsystem: Digital input
Mode: Interrupt
Operation: Start
☐ Audit only

Events
○ Timing >> Rate
● Start >> Digital
Delay: 0
○ Stop >> Terminal count

Select >>
○ Channels
○ Buffers
○ Flags

Digital Event
Channel: 0 AND Mask: 1
● Equals
○ Not equals
Pattern: 1

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

*A Digital Start Event requires an external timing source because the KPCI-PIO 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 Input Stop Events

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

The KPCI-PIO 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-PIO hardware has filled all the data buffers once.
- **Digital**—The KPCI-PIO 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.

## 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.

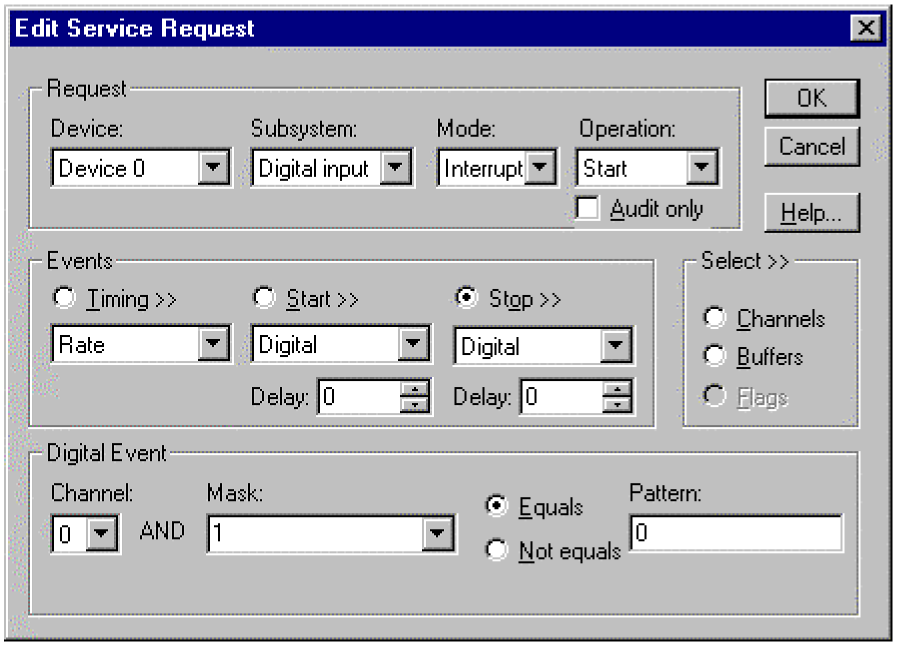
## Terminal Count Event

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-PIO 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, only the KPCI-PIO24 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.


Edit Service Request

Request
Device: Device 0
Subsystem: Digital input
Mode: Interrupt
Operation: Start
Audit only [ ]

Events
Timing >> [ ]  Start >> [ ]  [✓ Stop >>]
Rate
Digital  Delay: 0
Digital  Delay: 0

Select >>
Channels [ ]  Buffers [ ]  Flags [ ]

Digital Event
Channel: 0  AND  Mask: 1
[✓ Equals]  Not equals [ ]
Pattern: 0

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

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 multiple of the number of channels in the scan, from 0 to $(2^{32} - 1)$.

## Digital Input Channels

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

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

In addition, DriverLINX provides two methods to address parts or groups of digital channels:

- **Extended Logical Channels Addressing**—Acquire data from 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 Input Logical Channels

The KPCI-PIO24 has three digital input/output ports that DriverLINX designates as Logical Channels 0 to 2. DriverLINX defines two additional Logical Channels for the external interrupt signal but applications cannot directly read their values.

The KPCI-PIO96 has twelve digital input/output ports that DriverLINX designates as Logical Channels 0 to 11.

**KPCI-PIO24**

| Logical Channel | DriverLINX Function | KPCI-PIO External Connector |
|---|---|---|
| 0 | Digital Input/Output Port | Digital input/output lines PA0 - PA7 |
| 1 | Digital Input/Output Port | Digital input/output lines PB0 - PB7 |
| 2 | Digital Input/Output Port | Digital input/output lines PC0 - PC7 |
| 3 or ExtClk | External Clock | INT_REQ |
| 4 or ExtTrg | External Trigger | INT_REQ |

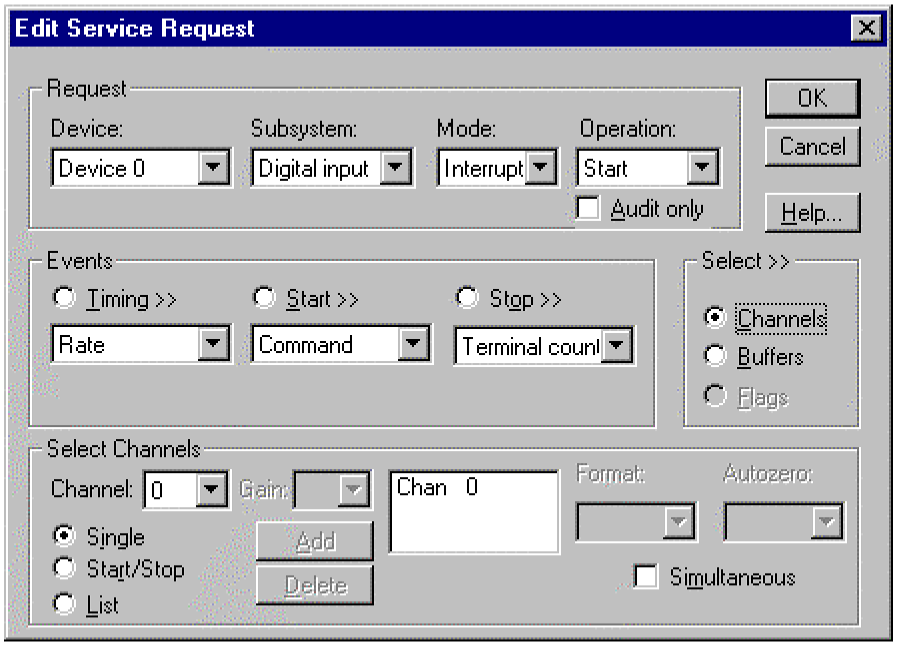
**KPCI-PIO96**

| Logical Channel | DriverLINX Function | KPCI-PIO External Connector |
|---|---|---|
| 0 | Digital Input/Output Port | Digital input/output lines Group 0 PA0 - PA7 |
| 1 | Digital Input/Output Port | Digital input/output lines Group 0 PB0 - PB7 |
| 2 | Digital Input/Output Port | Digital input/output lines Group 0 PC0 - PC7 |
| 3 | Digital Input/Output Port | Digital input/output lines Group 1 PA0 - PA7 |
| 4 | Digital Input/Output Port | Digital input/output lines Group 1 PB0 - PB7 |
| 5 | Digital Input/Output Port | Digital input/output lines Group 1 PC0 - PC7 |
| 6 | Digital Input/Output Port | Digital input/output lines Group 2 PA0 - PA7 |
| 7 | Digital Input/Output Port | Digital input/output lines Group 2 PB0 - PB7 |
| 8 | Digital Input/Output Port | Digital input/output lines Group 2 PC0 - PC7 |
| 9 | Digital Input/Output Port | Digital input/output lines |

| | | |
|---|---|---|
| | | Group 3 PA0 - PA7 |
| 10 | Digital Input/Output Port | Digital input/output lines Group 3 PB0 - PB7 |
| 11 | Digital Input/Output Port | Digital input/output lines Group 3 PC0 - PC7 |

## Single Channel Digital Input

In Single Channel mode, the KPCI-PIO acquires all data from one channel.



Request
Device: Device 0
Subsystem: Digital input
Mode: Interrupt
Operation: Start
☐ Audit only

Events
○ Timing >> Rate
○ Start >> Command
○ Stop >> Terminal count

Select >>
● Channels
○ Buffers
○ Flags

Select Channels
Channel: 0
Gain:
Chan 0
Format:
Autozero:
● Single
○ Start/Stop
○ List
Add
Delete
☐ Simultaneous

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

## Multi-channel Digital Input Range

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

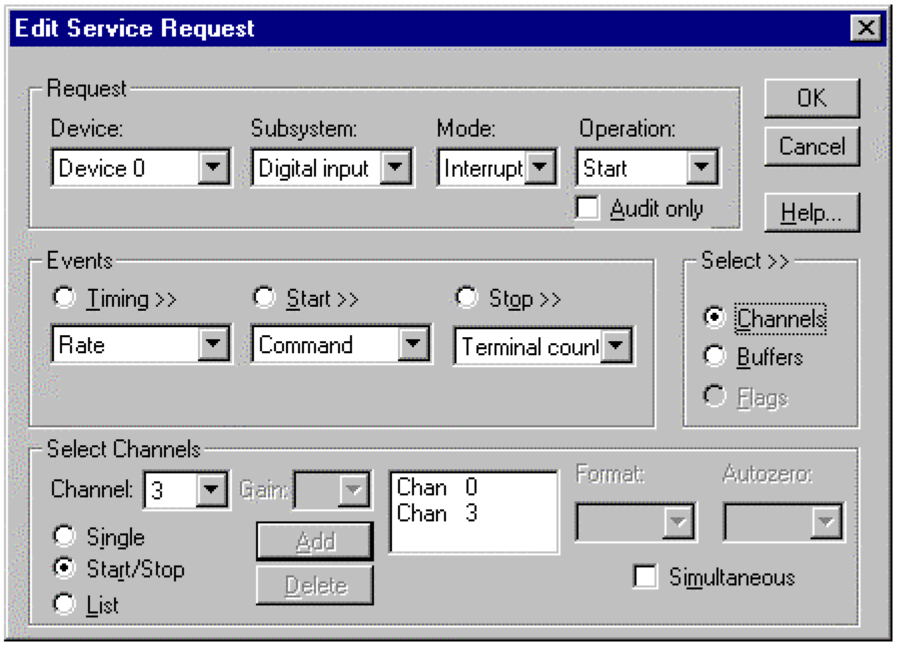
- Specifying a Stop Channel less than the Start Channel instructs DriverLINX to use the following sequence:

  ```
  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.


Edit Service Request

Request
Device: Device 0
Subsystem: Digital input
Mode: Interrupt
Operation: Start
☐ Audit only

OK · Cancel · Help...

Events
Timing >> Rate
Start >> Command
Stop >> Terminal count

Select >>
[✓ Channels] Buffers Flags

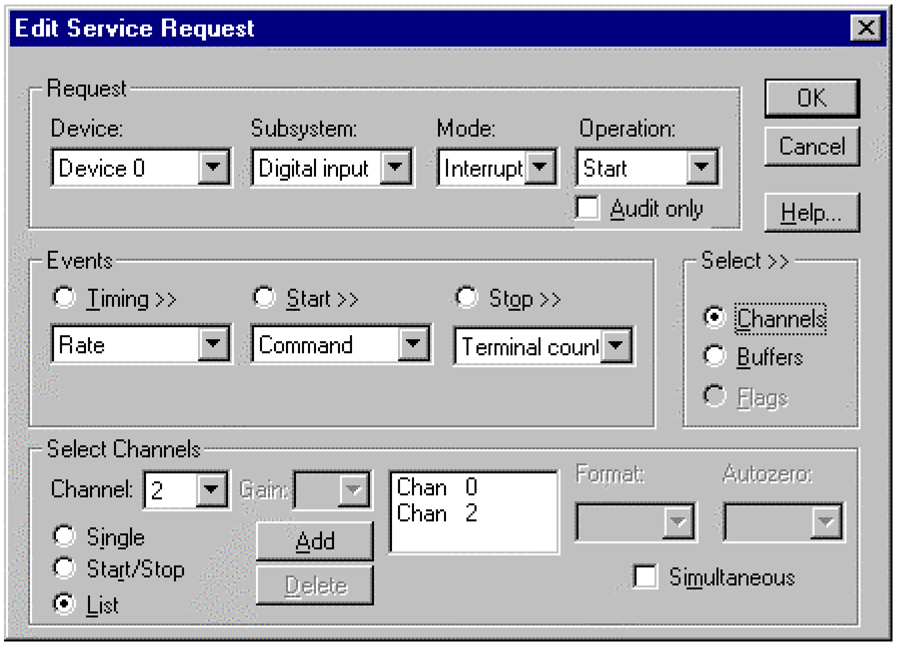
Select Channels
Channel: 3
Gain:
Single [✓ Start/Stop] List
Add · Delete
Chan 0
Chan 3
Format:
Autozero:
☐ Simultaneous

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

## Multi-channel Digital Input List

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

- Specifying *Simultaneous*, instructs DriverLINX to sample all the selected channels at each timing event.


Edit Service Request

Request
Device: Device 0
Subsystem: Digital input
Mode: Interrupt
Operation: Start
☐ Audit only

OK · Cancel · Help...

Events
Timing >> Rate
Start >> Command
Stop >> Terminal count

Select >>
[✓ Channels] Buffers Flags

Select Channels
Channel: 2
Gain:
Single Start/Stop [✓ List]
Add · Delete
Chan 0
Chan 2
Format:
Autozero:
☐ Simultaneous

How to set up the KPCI-PIO to read from 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:

| ***0*** | ***Size Code*** | | | ***Channel Number*** | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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-PIO 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 |
| 6 | dword | 32 |
| 7 | qword | 64 |

*"Native" units refer to the hardware-defined digital channel size. For the KPCI-PIO 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-PIO models, the 8-bit digital input/output ports have the following channel addresses for each size code:

**KPCI-PIO24**

| Unit | Channels | Address (dec) | Address (hex) |
|---|---|---|---|
| native | 0..2 | 0..2 | 0000..0002 |
| bit | 0..23 | 4096..4119 | 1000..1017 |
| half nibble | 0..11 | 8192..8203 | 2000..200B |
| nibble | 0..5 | 12288..12293 | 3000..3005 |
| byte | 0..2 | 16384..16386 | 4000..4002 |
| word | 0..1 | 20480..20481 | 5000..5001 |

**KPCI-PIO96**

| Unit | Channels | Address (dec) | Address (hex) |
|---|---|---|---|
| native | 0 .. 11 | 0 .. 11 | 0000 .. 000B |
| bit | 0 .. 95 | 4096 .. 4191 | 1000 .. 105F |
| half nibble | 0 .. 47 | 8192 .. 8239 | 2000 .. 202F |
| nibble | 0 .. 23 | 12288 .. 12311 | 3000 .. 3017 |
| byte | 0 .. 11 | 16384 .. 16395 | 4000 .. 400B |
| word | 0 .. 5 | 20480 .. 20485 | 5000 .. 5005 |
| dword | 0 .. 2 | 24576 .. 24578 | 6000 .. 6002 |
| qword | 0 .. 0 | 28672 .. 28672 | 7000 .. 7000 |

**Notes:**

- DriverLINX only supports single-value transfers on Logical Channels smaller than or equal to the 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.

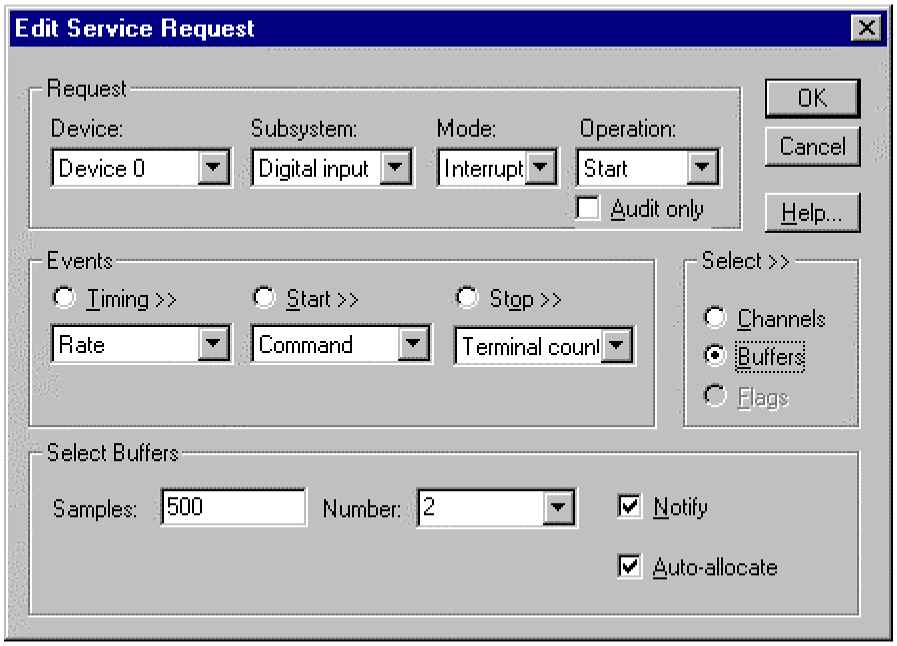
Bit masking allows an application or thread to access specific bits in a port without affecting bits controlled by other applications.

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 **256** and the buffer size as desired.


Edit Service Request

Request
Device: Device 0
Subsystem: Digital input
Mode: Interrupt
Operation: Start
☐ Audit only

Events
Timing >> Rate
Start >> Command
Stop >> Terminal count

Select >>
○ Channels
● Buffers
○ Flags

Select Buffers
Samples: 500
Number: 2
☑ Notify
☑ Auto-allocate

*How to set up the KPCI-PIO 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. *Applicable only to the ExtTrg channel on the KPCI-PIO24.* |
| Stop Event | DriverLINX has processed the interrupt for a hardware stop event. *Applicable only to the ExtTrg channel on the KPCI-PIO24.* |
| 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:

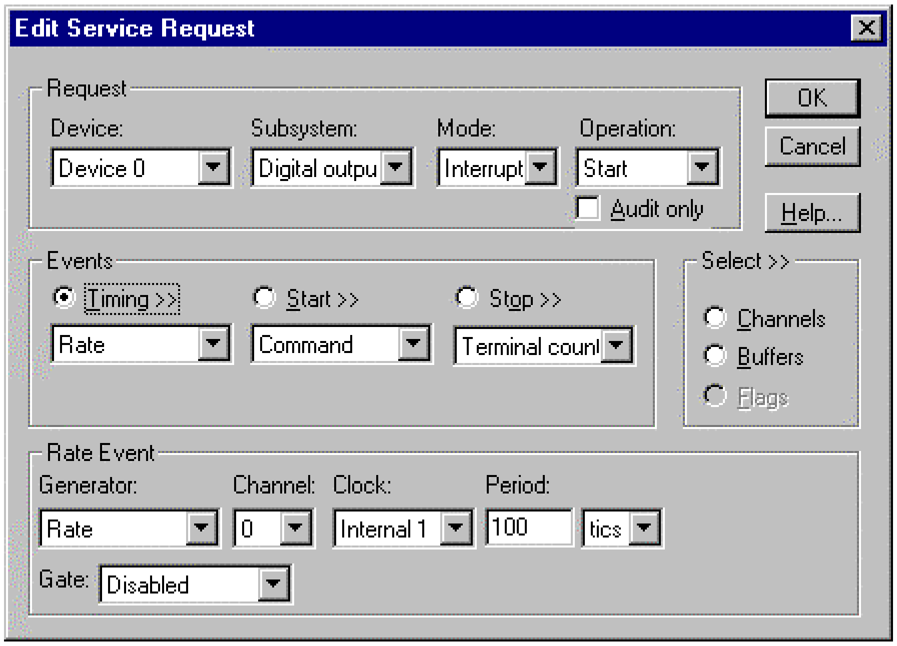
- ActiveX Control—“Overview of DriverLINX/VB Control Events” in the *DriverLINX/VB Technical Reference Manual.*
- C/C++—“DriverLINX Messages - Overview” in the *DriverLINX Technical Reference Manual.*

# Using the Digital Output Subsystem

## Introduction

This chapter describes how DriverLINX implements Digital Output Subsystem features for the KPCI-PIO 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-PIO 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*


Edit Service Request

Request
Device: Device 0
Subsystem: Digital outpu
Mode: Interrupt
Operation: Start
Audit only: [ ]

Events
[● Timing >>] Start >> Stop >>
Timing: Rate
Start: Command
Stop: Terminal count

Select >>
Channels
Buffers
Flags

Rate Event
Generator: Rate
Channel: 0
Clock: Internal 1
Period: 100 tics
Gate: Disabled

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-PIO 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-PIO 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.
- **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 output 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-PIO Series.

## Digital Port Configuration

A KPCI-PIO board has three or twelve digital ports that DriverLINX shares between Digital Input and Digital Output Subsystems. DriverLINX provides two methods to configure them as input or output:

- Users can statically configure ports, including the upper and lower nibbles of the C ports, using the *DriverLINX Configuration Panel*. (See "Configuring the KPCI-PIO" on page 11.)
- The KPCI-PIO Series also supports dynamic configuration, enabling applications to configure ports, with DIO Setup Events. (To configure the C ports as nibbles use the *DriverLINX Configuration Panel*.)

### *Dynamically Configuring KPCI-PIO Digital Ports*

When the KPCI-PIO 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 the driver reloads during the next boot.

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-PIO24.
- All inputs on the KPCI-PIO24 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<br>DIO_STROBED or<br>DIO_STROBED_NE |

- Specify the channel you want to reconfigure as the Logical Channel in the Digital Setup Event. (To configure the C ports as nibbles use the *DriverLINX Configuration Panel*.)
- 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 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-PIO writes the next digital output sample to the port.

The KPCI-PIO 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-PIO 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. Digital timing is available only on the KPCI-PIO24.

### None or Null Event

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

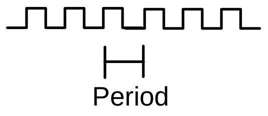
### Rate Event

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

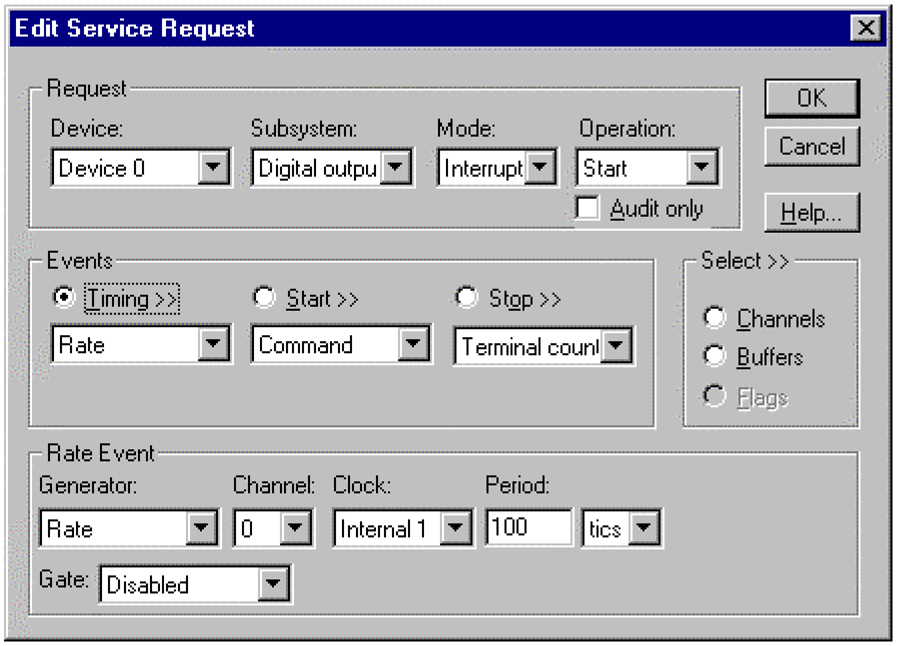
- **Rate Generator**—Generates a fixed rate clock with equal time intervals between tics.

## Rate Generator: Internal Clocking

An internally clocked Rate Generator produces a fixed rate clock with equal time intervals between tics.



Since the KPCI-PIO does not support a dedicated clock for digital output timing, DriverLINX uses the operating system timing clock for pacing digital output. The system timing clock does not have the resolution or time granularity of a dedicated clock, but it is an excellent substitute for low frequency digital output polling operations.



Edit Service Request

Request
Device: Device 0
Subsystem: Digital outpu
Mode: Interrupt
Operation: Start
☐ Audit only

Events
(● Timing >>) Rate
(○ Start >>) Command
(○ Stop >>) Terminal count

Select >>
○ Channels
○ Buffers
○ Flags

Rate Event
Generator: Rate
Channel: 0
Clock: Internal 1
Period: 100 tics
Gate: Disabled

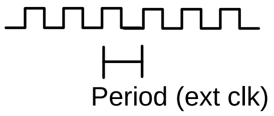
*How to set up the KPCI-PIO for fixed rate sampling using an internal clock.*

*For hardware independence, specify the clock channel using the symbolic constant, DEFAULTTIMER, which always maps to the default Logical Channel for digital output timing.*

- Specify internal clocking using a **Rate** *Generator* on *Logical Channel* **0** with the **Internal 1** *Clock* source.
- The *Period* property specifies the time interval between samples in tics, where a system timer tic is 1 μs, or 1 MHz. The minimum period is 1000 tics, or 1 kHz. The maximum period is 4,294,967,295 tics ($2^{32} - 1$), or 0.0002 Hz. The resolution and granularity of the system timer are operating system dependent. Under current versions of Windows, throughput is less than 100 Hz. See “Using the Counter/Timer Subsystem” on page 88 for details on the system timer.

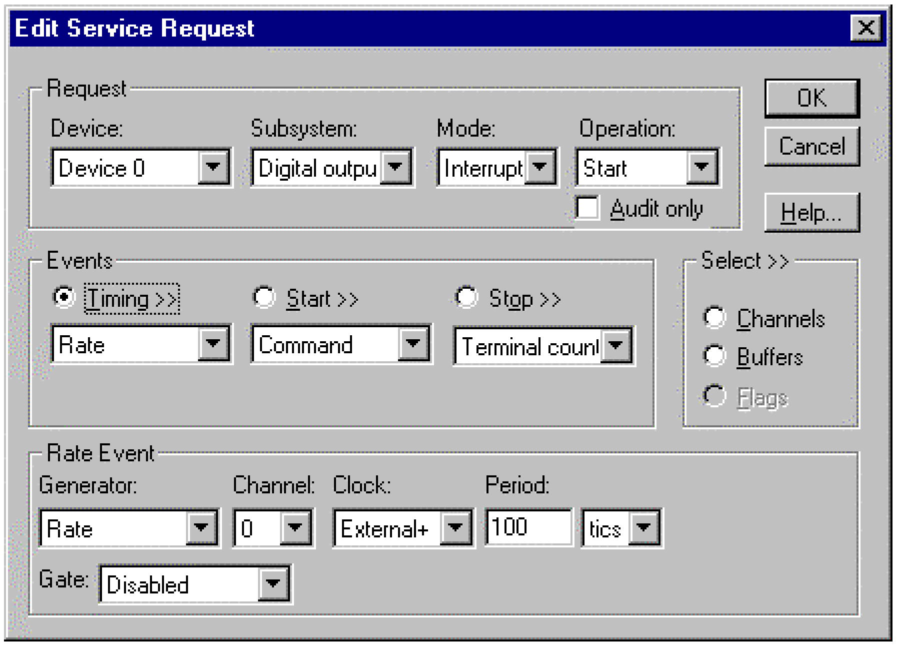
## Rate Generator: External Clocking

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



*Only the KPCI-PIO24 supports external clocking.*

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



Request
Device: Device 0
Subsystem: Digital outpu
Mode: Interrupt
Operation: Start
☐ Audit only

Events
● Timing >> Rate
○ Start >> Command
○ Stop >> Terminal count

Rate Event
Generator: Rate
Channel: 0
Clock: External+
Period: 100 tics
Gate: Disabled

*How to set up the KPCI-PIO24 for fixed rate sampling using an external clock.*

*A task cannot use the INT_REQ line (Digital Input Logical Channel 4) for a digital start or stop event while using it as an external clock source.*

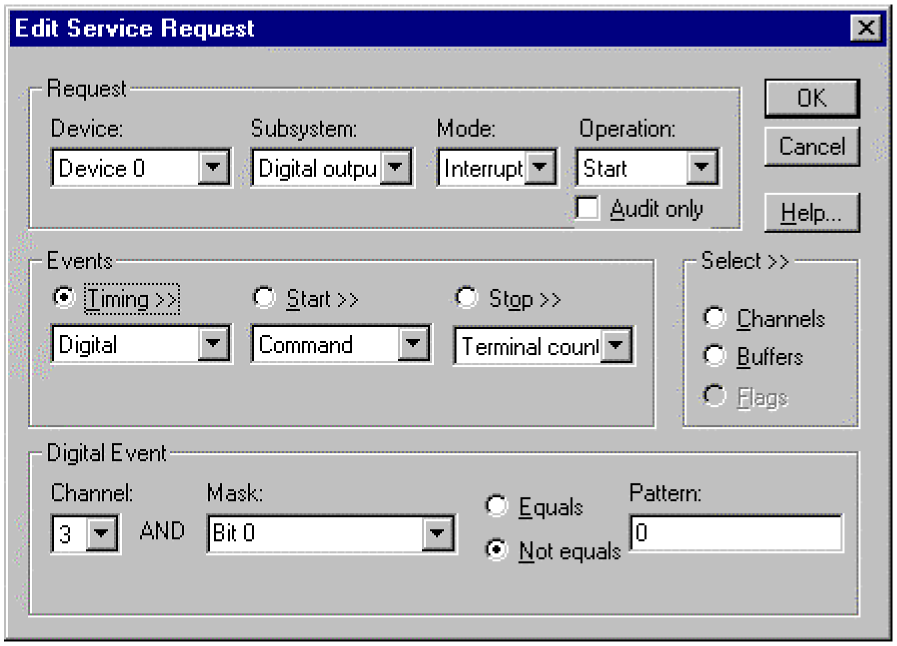
***BE SURE that the external clock source is TTL compatible, 0 V minimum to +5 V maximum!***

- 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

*Only the KPCI-PIO24 supports external clocking.*

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-PIO24 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.*

***BE SURE that the external clock source is TTL compatible, 0 V minimum to +5 V maximum!***

- Specify external clocking using *Logical Channel* **3**. 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 (≠0), or **1** for a falling, or negative, edge clock (≠1).

## Digital Output Start Events

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

The KPCI-PIO 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-PIO hardware for the task.
- **Digital**—The KPCI-PIO 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.

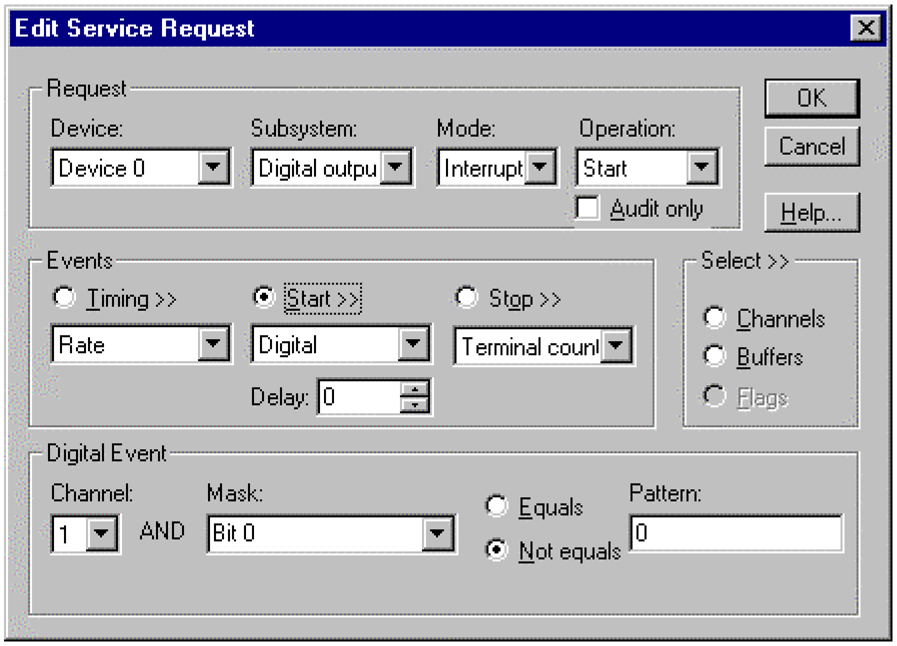
### Command Event

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

### Post-Trigger Sampling with a Digital Event

The KPCI-PIO 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, only the KPCI-PIO24 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.


Edit Service Request

Request — Device: Device 0; Subsystem: Digital outpu; Mode: Interrupt; Operation: Start; [ ] Audit only

Events — ( ) Timing >> Rate; (•) Start >> Digital; ( ) Stop >> Terminal count; Delay: 0

Select >> — ( ) Channels ( ) Buffers ( ) Flags

Digital Event — Channel: 1 AND Mask: Bit 0; ( ) Equals (•) Not equals; Pattern: 0

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

*A Digital Start Event requires an external timing source because the KPCI-PIO 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-PIO hardware stops writing digital output data.

The KPCI-PIO 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-PIO hardware has filled all the data buffers once.
- **Digital**—The KPCI-PIO 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.

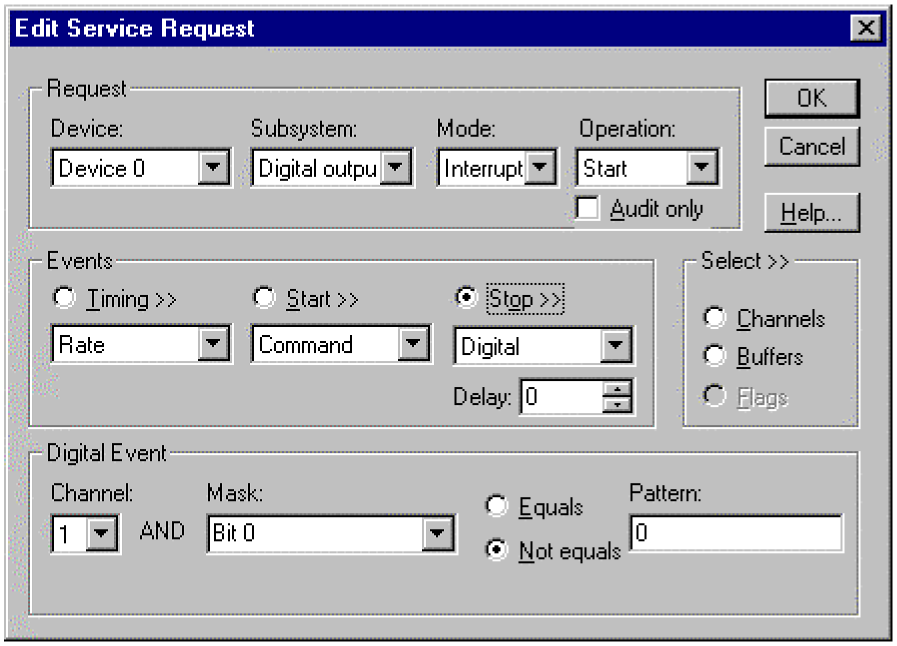
## Terminal Count Event

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-PIO 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-PIO24 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.


Edit Service Request

Request
Device: Device 0
Subsystem: Digital outpu
Mode: Interrupt
Operation: Start
☐ Audit only

Events
○ Timing >> Rate
○ Start >> Command
[✓ Stop >>] Digital
Delay: 0

Select >>
○ Channels
○ Buffers
○ Flags

Digital Event
Channel: 1 AND Mask: Bit 0
○ Equals
[✓ Not equals]
Pattern: 0

*How to set up the KPCI-PIO 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 any multiple of the number of channels in the scan, from 0 to $(2^{32} - 1)$.

## Digital Output Channels

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

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

In addition, DriverLINX provides two methods to address parts or groups of digital channels:

- **Extended Logical Channels Addressing**—Acquire data from 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-PIO24 has three digital input/output ports that DriverLINX designates as Logical Channels 0 to 2.

The KPCI-PIO96 has twelve digital input/output ports that DriverLINX designates as Logical Channels 0 to 11.

**KPCI-PIO24**

| Logical Channel | DriverLINX Function | KPCI-PIO External Connector |
|---|---|---|
| 0 | Digital Input/Output Port | Digital input/output lines PA0 - PA7 |
| 1 | Digital Input/Output Port | Digital input/output lines PB0 - PB7 |
| 2 | Digital Input/Output Port | Digital input/output lines PC0 - PC7 |

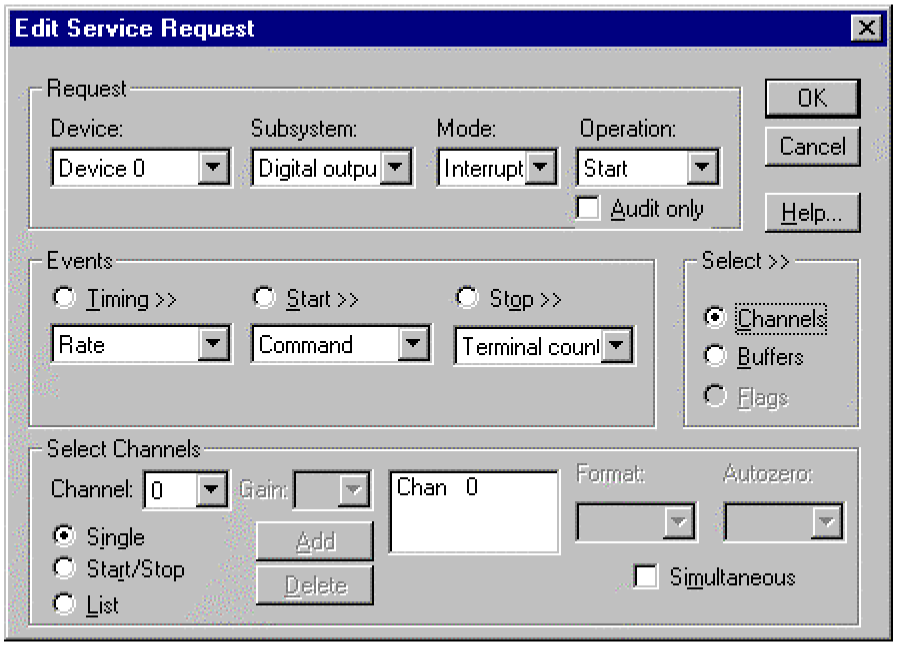
**KPCI-PIO96**

| Logical Channel | DriverLINX Function | KPCI-PIO External Connector |
|---|---|---|
| 0 | Digital Input/Output Port | Digital input/output lines Group 0 PA0 - PA7 |
| 1 | Digital Input/Output Port | Digital input/output lines Group 0 PB0 - PB7 |
| 2 | Digital Input/Output Port | Digital input/output lines Group 0 PC0 - PC7 |
| 3 | Digital Input/Output Port | Digital input/output lines Group 1 PA0 - PA7 |
| 4 | Digital Input/Output Port | Digital input/output lines Group 1 PB0 - PB7 |
| 5 | Digital Input/Output Port | Digital input/output lines Group 1 PC0 - PC7 |
| 6 | Digital Input/Output Port | Digital input/output lines Group 2 PA0 - PA7 |
| 7 | Digital Input/Output Port | Digital input/output lines Group 2 PB0 - PB7 |
| 8 | Digital Input/Output Port | Digital input/output lines Group 2 PC0 - PC7 |
| 9 | Digital Input/Output Port | Digital input/output lines Group 3 PA0 - PA7 |
| 10 | Digital Input/Output Port | Digital input/output lines Group 3 PB0 - PB7 |

| | | |
|---|---|---|
| 11 | Digital Input/Output Port | Digital input/output lines Group 3 PC0 - PC7 |

## Single Channel Digital Output

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


Edit Service Request

Request — Device: Device 0; Subsystem: Digital outpu; Mode: Interrupt; Operation: Start; Audit only [ ]

Events — Timing >>: Rate; Start >>: Command; Stop >>: Terminal count

Select >> — [✓ Channels] Buffers Flags

Select Channels — Channel: 0; Gain: ; [✓ Single] Start/Stop List; Chan 0; Format: ; Autozero: ; Simultaneous [ ]

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

## Multi-channel Digital Output Range

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

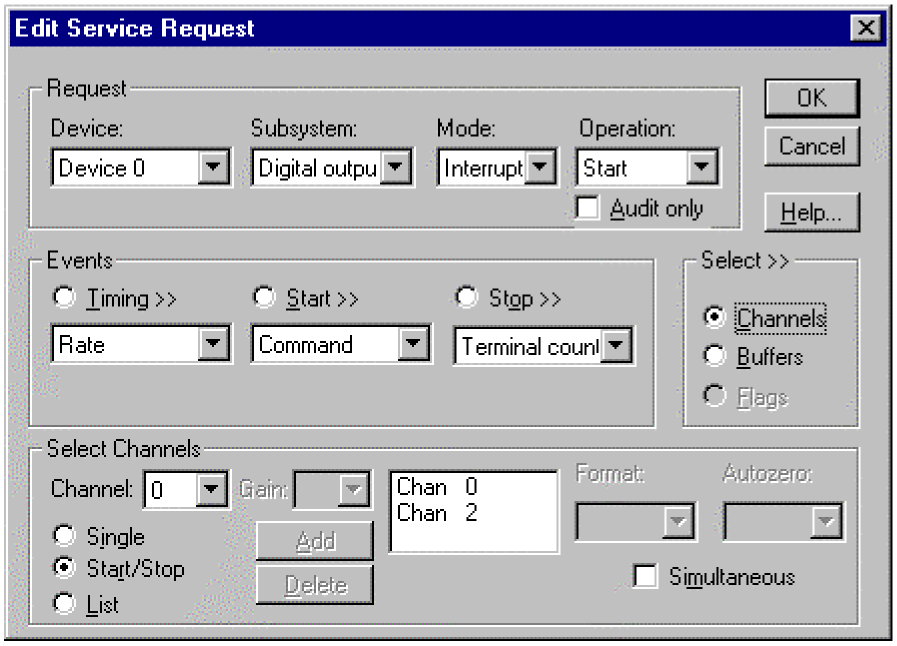
- Specifying a Stop Channel is less than the Start Channel instructs DriverLINX to use the following sequence:

  ```
  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 write to all the selected channels at each timing event.


Edit Service Request

Request — Device: Device 0; Subsystem: Digital outpu; Mode: Interrupt; Operation: Start; Audit only ☐

Events — Timing >>: Rate; Start >>: Command; Stop >>: Terminal count

Select >> — [✓ Channels] Buffers Flags

Select Channels — Channel: 0; Gain:; Single [✓ Start/Stop] List

Chan 0
Chan 2

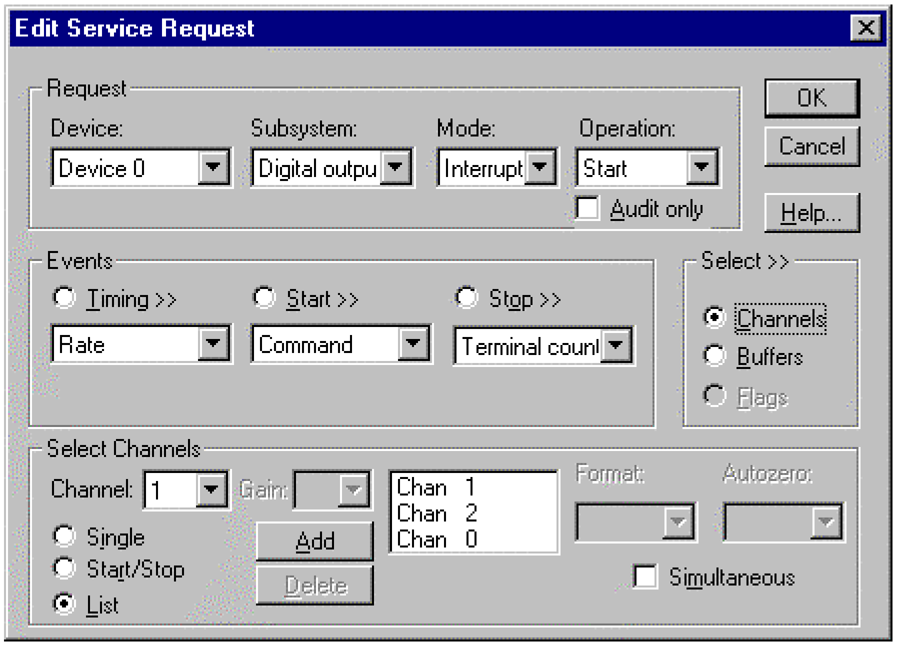
Format: Autozero: Simultaneous ☐

*How to set up the KPCI-PIO to write from a range of  channels*

## Multi-channel Digital Output List

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

- Specifying *Simultaneous*, instructs DriverLINX to sample all the selected channels at each timing event.


Edit Service Request

Request — Device: Device 0; Subsystem: Digital outpu; Mode: Interrupt; Operation: Start; Audit only ☐

Events — Timing >>: Rate; Start >>: Command; Stop >>: Terminal count

Select >> — [✓ Channels] Buffers Flags

Select Channels — Channel: 1; Gain:; Single Start/Stop [✓ List]

Chan 1
Chan 2
Chan 0

Format: Autozero: Simultaneous ☐

*How to set up the KPCI-PIO to write from 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:

| ***0*** | ***Size Code*** | | | ***Channel Number*** | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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-PIO 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 |
| 6 | dword | 32 |
| 7 | qword | 64 |

*"Native" units refer to the hardware-defined digital channel size. For the KPCI-PIO 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-PIO models, the 8-bit digital input/output ports have the following channel addresses for each size code:

**KPCI-PIO24**

| Unit | Channels | Address (dec) | Address (hex) |
|---|---|---|---|
| native | 0..2 | 0..2 | 0000..0002 |
| bit | 0..23 | 4096..4119 | 1000..1017 |
| half nibble | 0..11 | 8192..8203 | 2000..200B |
| nibble | 0..5 | 12288..12293 | 3000..3005 |
| byte | 0..2 | 16384..16386 | 4000..4002 |
| word | 0..1 | 20480..20481 | 5000..5001 |

**KPCI-PIO96**

| Unit | Channels | Address (dec) | Address (hex) |
|---|---|---|---|
| native | 0 .. 11 | 0 .. 11 | 0000 .. 000B |
| bit | 0 .. 95 | 4096 .. 4191 | 1000 .. 105F |
| half nibble | 0 .. 47 | 8192 .. 8239 | 2000 .. 202F |
| nibble | 0 .. 23 | 12288 .. 12311 | 3000 .. 3017 |
| byte | 0 .. 11 | 16384 .. 16395 | 4000 .. 400B |
| word | 0 .. 5 | 20480 .. 20485 | 5000 .. 5005 |
| dword | 0 .. 2 | 24576 .. 24578 | 6000 .. 6002 |
| qword | 0 .. 0 | 28672 .. 28672 | 7000 .. 7000 |

**Notes:**

- DriverLINX only supports single-value transfers on Logical Channels smaller than or equal to the 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.

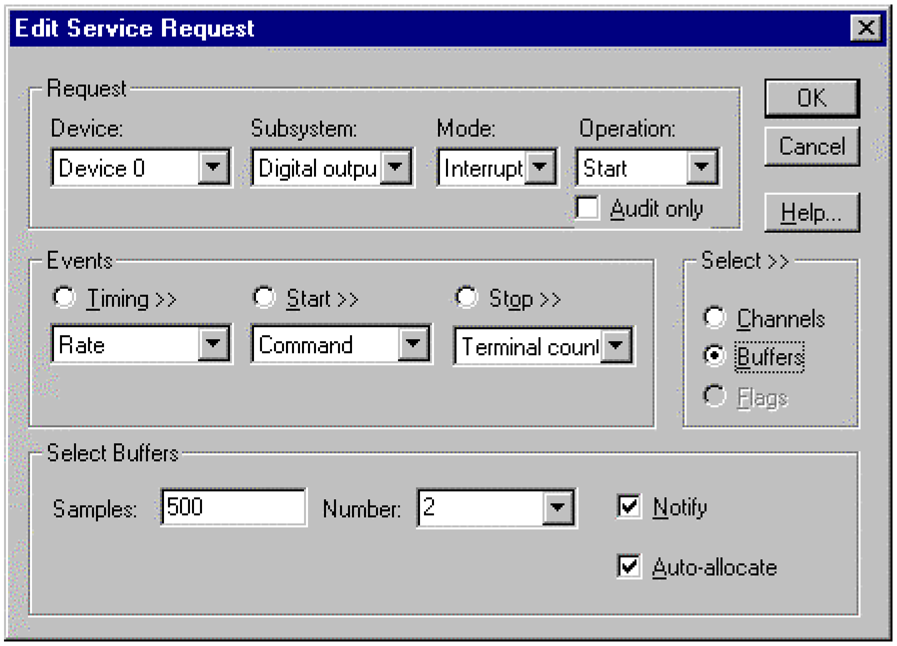
Bit masking allows an application or thread to access specific bits in a port without affecting bits controlled by other applications.

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 **256** and the buffer size as desired.


Edit Service Request

Request

Device: Device 0

Subsystem: Digital outpu

Mode: Interrupt

Operation: Start

[ ] Audit only

Events

( ) Timing >> Rate

( ) Start >> Command

( ) Stop >> Terminal count

Select >>

( ) Channels

(●) Buffers

( ) Flags

Select Buffers

Samples: 500

Number: 2

[✓] Notify

[✓] Auto-allocate

*How to set up the KPCI-PIO 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. *Applicable only to the ExtTrg channel on the KPCI-PIO24.* |
| Stop Event | DriverLINX has processed the interrupt for a hardware stop event. *Applicable only to the ExtTrg channel on the KPCI-PIO24.* |
| 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-PIO has a single counter/timer. DriverLINX provides this counter/timer through software as the KPCI-PIO does not have a hardware clock. You can use the counter/timer to 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** | **Clocks** | | **Modes** | **Gates** |
|---|---|---|---|---|
| **Channel** | **Source** | **Tic Period** | | |
| 0 — System | Internal 1 | 1 µs (1 MHz) | Rate Gen | Disabled<br>No Connect |
| | External | | | |
| | External+ | | | |
| | External- | | | |

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

## System Pacer Clock

The System Pacer Clock (Logical Channel 0) is an internal, software clock source based on the Windows system timer. DriverLINX uses this internal system clock to pace digital I/O as the KPCI-PIO does not have an internal hardware timer. The System Pacer Clock does not support independent counter/timer tasks.

**Note:** DriverLINX provides the system clock for *low frequency* polling of digital I/O. The resolution and granularity of the system timer are machine and operating system dependent.

### Internal Clocking

The System Pacer Clock has a single frequency source to pace digital I/O tasks. DriverLINX defines the following internal clock source for this Logical Channel:

- **Internal 1—**specifies a 1 MHz clock source. With this clock source each tic is 1 µs.

Typically, the system timer has a 10 msec resolution and granularity. Under current versions of Windows, throughput is less than 100 Hz.

### External Clocking

The KPCI-PIO 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 regular intervals based on an internal or 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-PIO 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-PIO 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-PIO Series, you must

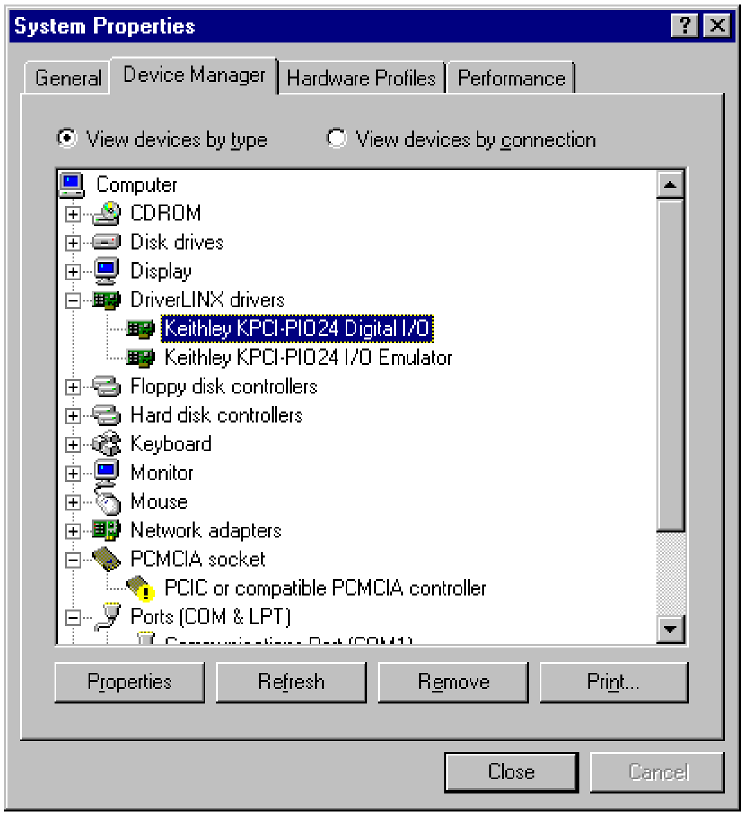
- Disable the DriverLINX driver.
- Shut down your computer to remove the hardware.
- Reboot your computer to unload the driver.
- Run the DriverLINX uninstall program.

### *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-PIO 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-PIO card that you are uninstalling.
5. Close the DriverLINX Configuration Panel.
6. When finished, shut down your computer and physically remove any installed KPCI-PIO hardware.
7. Reboot Windows.
8. To finish uninstalling, see “How to Remove DriverLINX for Keithley KPCI-PIO Series” on page 92.

## How to Disable a DriverLINX Driver in Windows 95/98

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-PIO devices.



System Properties

General | Device Manager | Hardware Profiles | Performance

(●) View devices by type ( ) View devices by connection

- Computer
  - CDROM
  - Disk drives
  - Display
  - DriverLINX drivers
    - Keithley KPCI-PIO24 Digital I/O
    - Keithley KPCI-PIO24 I/O Emulator
  - Floppy disk controllers
  - Hard disk controllers
  - Keyboard
  - Monitor
  - Mouse
  - Network adapters
  - PCMCIA socket
    - PCIC or compatible PCMCIA controller
  - Ports (COM & LPT)

3. Select the KPCI-PIO 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-PIO card or driver that you uninstalling.
7. When finished, click "Close", shut down your computer, and physically remove any installed KPCI-PIO hardware.
8. Reboot Windows.
9. To finish uninstalling, see "How to Remove DriverLINX for Keithley KPCI-PIO Series" on page 92.

## How to Remove DriverLINX for Keithley KPCI-PIO 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-PIO 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-PIO 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-PIO hardware requires successful completion of three steps.

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

If you are having a problem installing or configuring your KPCI-PIO 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-PIO hardware and then install KPCI-PIO drivers for the hardware.

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

If you installed the KPCI-PIO hardware before you installed the DriverLINX software, Windows 95/98 may install the KPCI-PIO 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 95/98*

If you installed the KPCI-PIO hardware before installing the DriverLINX software, Windows 95/98 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”. If you do not see it, go to “Editing the Windows 95/98 Registry” on page 96.
5. Click on the “+” icon next to “Other devices” and search for “PCI Card”. If you do not find it, go to “Editing the Windows 95/98 Registry” on page 96.
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”. If this fails, see “Editing the Windows 95/98 Registry” on page 96.
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-PIO Series” on page 11.

## Editing the Windows 95/98 Registry

Before trying to edit the Windows Registry, follow the instructions for "Fixing Problems Installing Hardware Before Software on Windows 95/98" on page 95.

***Caution: Incorrect or unintended changes to Registry can prevent Windows from booting. A complete reinstall of Windows may be the only recovery option. For safety, first make a backup copy of the Registry using the “Registry\Export Registry File…” menu option in the Registry Editor. Then follow these instructions carefully.***

1. From the Windows “Start” menu, select “Run”, type “regedit” into the Run dialog and click OK
2. Under “My Computer”, first click the “+” next to “HKEY_LOCAL_MACHINE”.
3. Next click the “+” next to “Enum” and then “PCI”.
4. If you have a KPCI-PIO24, search for “VEN_11F3&DEV_0011” on Win 95 or “VEN_11F3&DEV_0011&SUBSYS_00000000&REV_00” on Win 98. If you have a KPCI-PIO96, search for “VEN_11F3&DEV_0012” on Win 95 or “VEN_11F3&DEV_0012&SUBSYS_00000000&REV_00” on Win 98.
5. If you find neither entry, no Registry edits are needed. Exit the Registry Editor and quit these instructions.
6. If you find either entry, select it and then delete the key by selecting “Edit\Delete” from the Registry Editor menu. Answer “Yes” to “Are you sure you want to delete this key?”
7. Install the DriverLINX software, if you have not previously done so, and then reboot your machine.

## Solving Problems Configuring the Drivers

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

## Solving Problems Loading Drivers

Before the KPCI-PIO drivers can load, you must

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

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

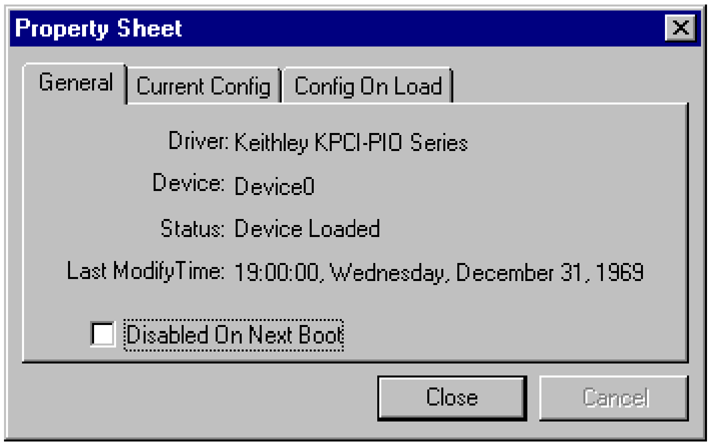
On both Windows NT and Windows 95/98, the operating system will automatically assign hardware resources to the KPCI-PIO cards. Automatic resource assignment can fail sometimes on

- 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-PIO 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 95/98, you see only a hardware description, you did not configure the driver. See “Configuring the KPCI-PIO 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.


Property Sheet

General | Current Config | Config On Load

Driver: Keithley KPCI-PIO Series

Device: Device0

Status: Device Loaded

Last ModifyTime: 19:00:00, Wednesday, December 31, 1969

☐ Disabled On Next Boot

8. If you rebooted the computer after configuring and Windows did not load your device, see “Checking for Device Errors” on page 98.

## 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 95/98 or Windows NT using the DriverLINX Event Viewer.

Windows 95/98 maintains additional driver information in the Device Manager. Also see “Getting More Driver Information on Windows 95/98” on page 98.

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 95/98

Windows 95/98 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 95.
4. Click the “+” next to “DriverLINX drivers”.
5. Does your KPCI-PIO product appear in the list? If not, see "Solving Problems Recognizing and Installing Drivers" on page 95.
6. Does the icon next to your KPCI-PIO product display an exclamation point (!)? If no, Windows has loaded your KPCI-PIO 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*

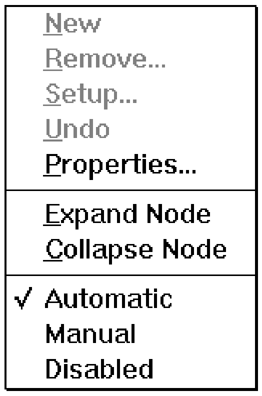
On Windows NT, the only reasons that a driver does not load are

- 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 98 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-PIO 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 95, 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.

## TTL

Abbreviation for transistor-transistor logic, a family of digital logic elements.