## Using the Direct I/O COM API of DriverLINX for the KPCI-PDISO8A or KPCI-PIO32IOA with LabWindows/CVI

DriverLINX drivers for digital I/O boards provides two programming interfaces:

- 1. Service Request API for hardware independence with other KPCI or PIO boards
- 2. Direct I/O COM API for simple, fast interaction with the registers of the digital I/O hardware

COM objects are easily used from Visual Basic or from C++. Use of a COM object from a purely C language such as CVI requires a bit more work to build a proper interface layer into the object. Luckily, LabWindows/CVI provides an automated tool to greatly simplify creation of the interface layer. The resulting function panel file, \*.fp, can then be used in conventional ways by CVI to make use of the function calls of the exposed object.

Below is a step by step guide. The completed project is also available as an example program in the download center.

🔊 f:	\\Di	irectI(	)\Dire	ctIO_	API.	prj			8				_ 🗆 🗙
<u>F</u> ile	Edit	<u>V</u> iew	<u>B</u> uild	<u>R</u> un	Ins	trument	Lit	orary	<u>T</u> ools	<u>W</u> indow	Options	Help	
Nam	ne				_	S	С	0	Crea	ate Active	<u>X</u> Automati	on Con	troller
									Crea	ate IVI <u>I</u> ns	trument Di	iver	
									Sour	rce Code (	Control		)
									UI to	o <u>C</u> ode Co	nverter		
									User	<sup>,</sup> Interface	e <u>L</u> ocalizer.		
													-
4					F					4			> //

Step 1: In the Tools menu, select 'Create ActiveX Automation Controller....'

Step 2: Scroll down the list until you see the Keithley Digital I/O Library. Select it and then click Next.



Step 3: Fill in a name for the target .fp file. Any name is fine; I used DirectIO (be sure to include the path to the location). Then click the Next button.

- 101 4 L 01 4 L 2 n	Automation Server	Keithley Digital I/O Library	
Active	Transmitter Corror.	<u>,</u>	
Automation	Instrument <u>P</u> refix:	KDIGIOLib	
2 Controller	,		
<b>Wizard</b>	Target .fp File:		Browse
Selou	Property Access Fund	ctions	
Instantion	Per Property (fas	ter)-	
Winds	Per Obj	ect-	
Get Pioper	Per Ser	ver-	
Event Collector			
Global Application			
Document			

Step 4: the object will be analyzed by CVI and several output files will be built and placed into the directory location you specified. If any other dialogs present, accept the default values and click Next. When complete, the screen below will appear.



Step 5: Now to use the object. From this same session of CVI, the function panel will already be loaded in the Instrument Driver menu drop down. If starting a new session of CVI, first load the fp file.

🔊 f:	\\D	irectIC	)\Dire	ctIO_/	API.prj							<
Eile	Edit	⊻iew	Build	<u>R</u> un	Instrument	Library	Tools	<u>W</u> indow	Options	Help		
Nan	ne				<u>1</u> Keithley	Digital I/(	) Library	/			 	
					Load Unload Edit						-	
-												-
					¥			4			- F .	11.

Step 6: Once the class has been selected, the member functions are visible. The important ones are New, OpenDevice, CloseDevice, Read and Write. This was just a preview of where we will be going, so hit the Cancel button for now.

Select Function Panel		
Instrument: Keithley Digital I/O Library Class: IKDigitallo		
Window or Class		
New IKDigitallo Open IKDigitallo Active IKDigitallo OpenDevice CoseDevice		Selec
Read Write GetDriver GetDevice		Up Cance
GetUpenLount		<u>H</u> elp.
☐ <u>Function Names</u> ☐ <u>F</u> latten	III Alphabalica 区 New Window	

Step 7: Create a project called testDirectIO. Add the DirectIO.fp file to the project. Add also a uir file to the project. Place several buttons onto the user interface as shown below and create all the callback code stubs (a \*.c file is created).

📑 f:\CV	15.5\	Activ	eX\te	stDirectI	0.uir				×
<u>File</u>	it ⊆r	eate	<u>V</u> iew	Arrange	Co <u>d</u> e	<u>R</u> un	Library	Tools	₩in
<u>8</u> [	] <u>A</u>	K							
		Γ	Jsing D	irect I/O C	ОМ ОЬј	ect			*
		C		<u>O</u> pen Driv	/er		)		
		Ć.		<u>C</u> lose Rel	ays		)		
		C		<u>O</u> pen Rel	ays		)		
		C		<u>E</u> xit			)		
<u> </u>									

In the call back for the Open Driver button, the Direct IO API will be initialized and a session opened for the installed board (the device number used in the DriverLINX Configuration Panel will be used).

In the Close Relays and Open Relays buttons, values will be written to the appropriate register of the KPCI-PDISO8A or KPCI-PIO32IOA to control the state of the relays.

In the Exit button, the driver will be closed.

Step 8: Here is what the project will look like with it's three files.

🔊 f:\C¥I5.5\ActiveX\D	irectIO.prj		- O ×
<u>File E</u> dit <u>V</u> iew <u>B</u> uild	<u>R</u> un <u>I</u> nstrument Library	Tools Window Options He	lp
Name	🔲 S C O I 🗹	Date	
testDirectIO.c	-	5/28/2002, 3:50 PM	
DirectIO.fp	🗖 A	5/28/2002, 3:21 PM	
testDirectIO.uir		5/28/2002, 3:54 PM	
4		4	* /

Step 9: Let's use the fp file to add code to our .c file. Select the NewIKDigitallo and then click the Select button on the right.

Select Function Panel		
Instrument: Keithley Digital I/O Library Class: IKDigitallo		*
Window or Class		
New IKDigitallo Open IKDigitallo Active IKDigitallo OpenDevice		Select
CloseDevice Read Write GetDriver		<u>U</u> p
GetDevice GetOpenCount		Cancel
		<u></u> <u>H</u> elp
Function Names	Г Арларовісо	
 <u> </u>	🔽 New Mindow	

Step 10: the screen below is presented to aid in the use of the function. Notice the variable Object Handle is not filled in with a value. This indicates it needs to be declared by you. Select this object with the mouse, and then ask CVI to declare the variable (Ctrl+D, or menu structure of Code or the short cut buttons). (The Status variable could also be declared for a more robust implementation that includes error checking, but for brevity, this is skipped in this application note.)

🧱 Keithley Digital I/O Libra	ry - New IKDigitalIo		
<u>File C</u> ode <u>V</u> iew <u>I</u> nstrument	: <u>L</u> ibrary <u>T</u> ools <u>W</u> indow	Options <u>H</u> elp	
****		3 0	
	KDIGIOLib_Nev	wIKDigitallo	
Server	Support Multithreading	Locale	Reserved
NULL	1	LOCALE_NEUT	0
	Object Ha	indle	
	<u> </u>		
			Status
KDIGIOLib_NewIKDigi	talIo (NULL, 1, L	OCALE_NEUTRAL,	0, ); 🕘
			-

Step 11: The following dialog is presented for declaring our object handle variable. Fill in a variable name; I used myPIO. Also check the box indicating the declaration should be added to the top of the target .c file for our project. Then click the OK button.

🏘 Declare Variabl	le		×
Variable Type:	CAObjHandl	e	
⊻ariable Name:	myPIO		
Execute declara	tion in Interactive Win	dow	
Add declaration	to top of target file "te	stDirectIO.c"	
C Add declaration	to current <u>b</u> lock in targ	jet file "testDirectI0.c"	
		ı	
<u>S</u> et	Target File	<u>O</u> K	Cancel

Step 12: Notice the field is now filled in with the proper calling syntax (address of operator). In the .c file, place the curser within the callback function for the Open button, and then back on the screen below ask CVI to "insert function call" using the Code menu structure.

🧱 Keithley Digital I/O Lit	orary - New IKDigitalIo		<u>_   ×</u>					
<u>File C</u> ode <u>V</u> iew <u>I</u> nstrum	ent <u>L</u> ibrary <u>T</u> ools <u>W</u> indow	) Options <u>H</u> elp						
🕺 🌠 🖑 📎 🚔 🎿	s <b>922</b> est							
KDIGIOLib_NewIKDigitallo								
Server	Support Multithreading	Locale	Reserved					
NULL	1	LOCALE_NEUT	0					
Object Handle								
			<del>ب</del> <u>حر</u>					
KDIGIOLib_NewIKDig   _	gitalIo (NULL, 1, 1	LOCALE_NEUTRAL,	0, &myPIO); 🗳					

Step 13: Below is a screen shot of the callback for the open button in the .c file after two functions of the Direct IO COM object have been used. All subsequent methods of the Direct I/O object require the Object Handle obtained from the NewIKDigitalIo method. Notice the syntax used with the OpenDevice method (no & operator). Also notice that this method requires the driver name as a string, "kpciiso", and the device number of the installed board from the DriverLINX Configuration Panel. If more than one board of the kpciiso driver was installed, the device number would be used to differentiate between them.



Once the object has been created and the driver opened for the installed device, the Read and Write functions can be used to get data into and out of the KPCI card. You will need to know something of the card's register map to know at what offsets (address) and what values to write. For example, the relay outputs of the KPCI-PDISO8A are at an offset of 4. A value of 255 will close all of them; a value of 0 will open all of them.

A complete CVI project is available in the download center of the www.keithley.com web site.