Software Options for ISA PIO cards

Three options exist for programming an ISA PIO card: DriverLINX driver, register level control, or the legacy ASO driver (not recommended for new applications).

DriverLINX:

DriverLINX is a Windows (Win9x to WinXP) device driver for a plug-in board. The board will show up in Windows' Device Manager and the resources assigned to the board are registered as in use. DriverLINX provides three different API for programming an ISA digital I/O board (see the section titled "Programming the PIO Series" in the kmbpio.pdf software guide (page 21) in the drive:\drvlinx4\docs\notes folder for full information).

First recommendation is to use the Direct I/O COM object (ActiveX Direct I/O). This object allows use of simple read and write commands to interact with the digital I/O boards. Unless the external interrupt feature of the PIO-12 or PIO-24 are required, the Direct I/O COM object should always be considered. The Direct I/O object is compatible with any ActiveX hosting language (VB, MSVC++, Delphi, Builder,LabWindows/CVI, Excel, etc.). The methods of this object emulate the 8-bit port I/O behavior of an Intel 8255 chip (one control register and one register for each 8-bit digital port). The kmbpio.pdf document gives a good step by step process of how to use this object from the Visual Basic environment. Chapter 7 of the DriverLINX Tutorial Manual provides additional information. Example programs are available from the download center of this web site. **If you wish to dynamically configure the board so that Port C is split, this API is the only method for accomplishing this.**

Second recommendation is to use the Service Request API of DriverLINX. The advantage of the Service Request API is that it provides a hardware independent means of programming common functions of plug-in data acquisition boards. For example, the same DriverLINX code that controls a port of a PIO board can be reassigned to the corresponding digital I/O port of a multifunction board such as the KPCI-3108 with its 4 digital ports (32 lines of digital) or to the PCI version of a PIO card (KPCI-PIO24, etc.). The calls are the same because the underlying driver files handle the hardware differences. Thus the required code does not change, and time invested in application development is easily ported to new hardware when using the DriverLINX API. DriverLINX for your ISA PIO board (not available for PIO-INT) can be downloaded from this web site.

Applications which need to split PortC can do so when using this API, but only by first statically configuring the port in the DriverLINX Configuration Panel. Any application which relies upon this static configuration will fail if this configuration is subsequently changed. Using this Service Request API, an 8-bit port can be dynamically configured for input or output mode using his API, just the subset of trying to split port C into 4-bit nibbles cannot be accomplished from the application code.

Example programs in various languages using the full DriverLINX API for digital I/O are available for download.

Register Level Control: Because of their simple architecture, the ISA PIO cards easily lend themselves to many methods of software control. Unless the external interrupt feature of the PIO-12 or PIO-24 are required, direct register level control should always be considered. For these two boards, only 4 registers are required to completely control the 24 digital lines. DOS C/C++ provides inp and outp functions to directly access the I/O space the board occupies. From Visual Basic, a 32-

bit port I/O utility is required (port95nt.exe). This native 32-bit utility restores "DOS like" port I/O capability to Windows programming languages in Win9X, WinNT and even Win2000 and Windows XP. (C/C++ programmers working in WinNT or higher should use this utility in order to have kernel mode access to port I/O space.)

The PIO-96 and PIO-96J, which do not have an Interrupt feature, can also be controlled via register level commands. The register map of the PIO-96X series is very similar to the PIO-12 or PIO-24. The PIO-96X series can be thought of as 4 of the 24 line PIO cards assembled onto a single card. Each of the 4 groups of 24 lines on the PIO-96X series are controlled in exactly the same manner as the corresponding PIO-12 or PIO-24.

The rest of Keithley's ISA PIO line such as the REL-16, PDISO-8, PIO-32 series, etc., can also easily be programmed by direct register level control. This method of software control is often the most straightforward and easiest to maintain.

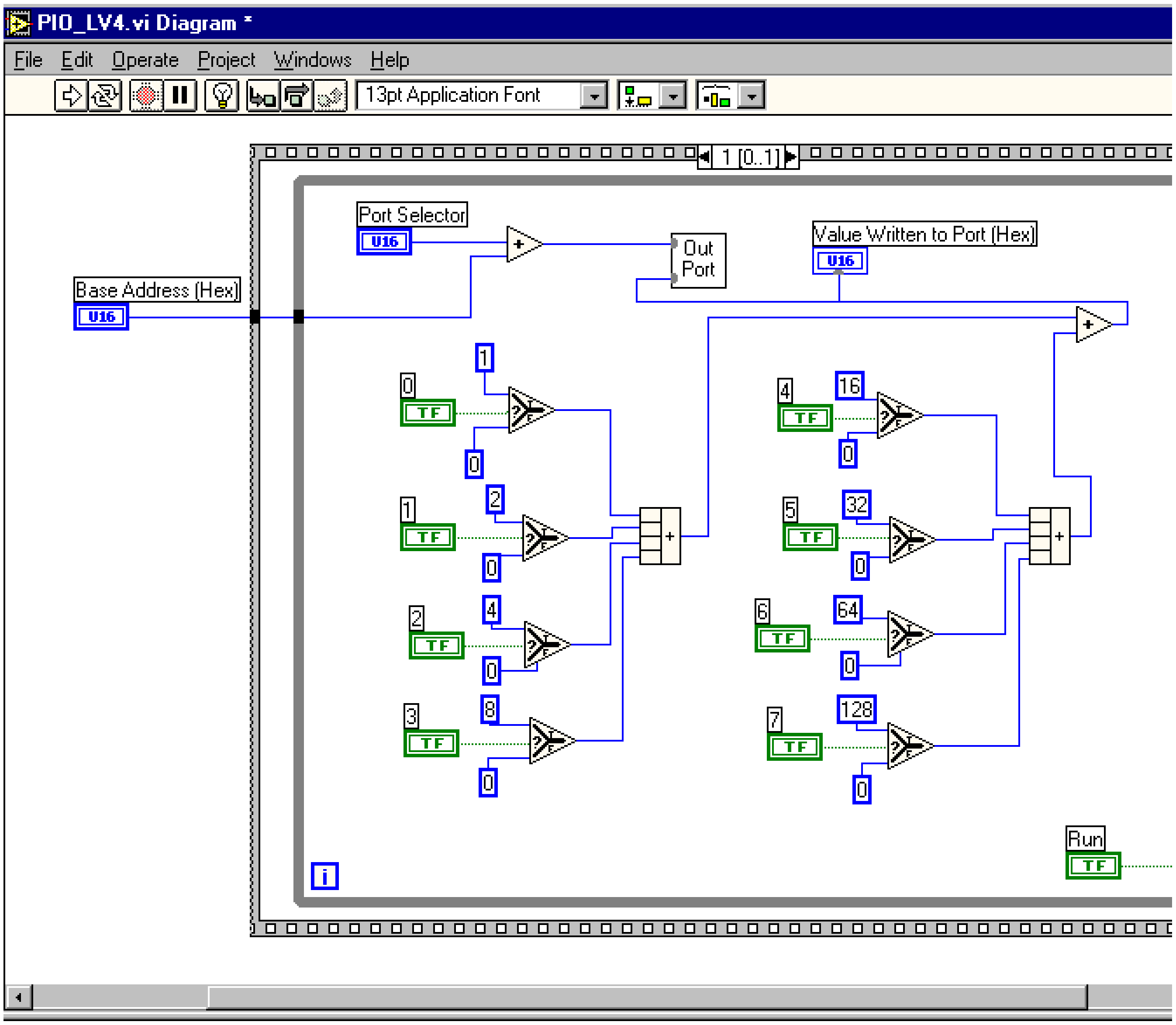
ASO (K_DOWrite type commands): The Advanced Software Option (ASO) was developed for Win3.1 for many Keithley/Metrabyte boards. These function call libraries were later upgraded for use in Win9X with 32bit developement environments like Visual Basic 5.0 or MSVC++ 4.0. **The ASO function call libraries will not work in WinNT or higher.** Since these ASO libraries are loaded only at run-time, the PIO card will not show up in Windows' Device Manager. To use the ASO driver with 32-bit compilers such as VB6 or VC++ 6, both the original 16 bit ASO software (pioaso11.exe) and the 32 bit upgrade (asowin95.exe) are required.

ISA PIO boards and LabVIEW

Two Options for LabVIEW:

Option 1. A full suite of VIs based on DriverLINX are available by download. There are easy, intermediate and advanced VIs in this package. For those familiar with National Instruments' VIs, these will seem very familiar. If direct register level control is not appealing, the VIs based upon the DriverLINX driver are recommended. **Requirements:** LabVIEW 5.0 or higher and DriverLINX drivers.

Option 2. LabVIEW provides VIs for direct register level control, In Port and Out Port. The versions of these VIs that ship with version 5.0 of LabVIEW are for Win9X and will not function in WinNT or higher, however, National Instruments has upgrades for these available for download from their web site (AccessHW.zip). These replacement VIs have the same look, feel, and function as the ones that ship with the LabVIEW. Below is a screen capture of what the use of these Port VIs looks like:


PIO_LV4.vi Diagram *
File Edit Operate Project Windows Help
13pt Application Font
1 [0..1]
Port Selector
U16
Base Address (Hex)
U16
Out
Port
Value Written to Port (Hex)
U16
0
1
2
3
4
5
6
7
TF
1
2
4
8
16
32
64
128
0
Run
TF
i