



Interchangeability The What, When and How



Alan Hume, BSc MSc
Pickering Interfaces
April 2013



Switch On to Pickering
www.pickeringtest.com

References

References to NI in this document refer to National Instrument Corporation.
MAX is an abbreviation for Measurement and Automation Explorer, an NI product.
www.ni.com

PI refers to Pickering Interfaces Ltd
www.pickeringtest.com

The IVI Foundation promotes and maintains the IVI driver standard (and other standards)
www.ivifoundation.org



What is Interchangeability?

- **Interchangeability permits similar devices from different manufacturers to be interchanged without the need to modify application code**
- **This permits the user to freely change hardware for similar products with no implications to the software developed on a test system**
- **This concept is the central tenet of the IVI suite of drivers, if properly used**

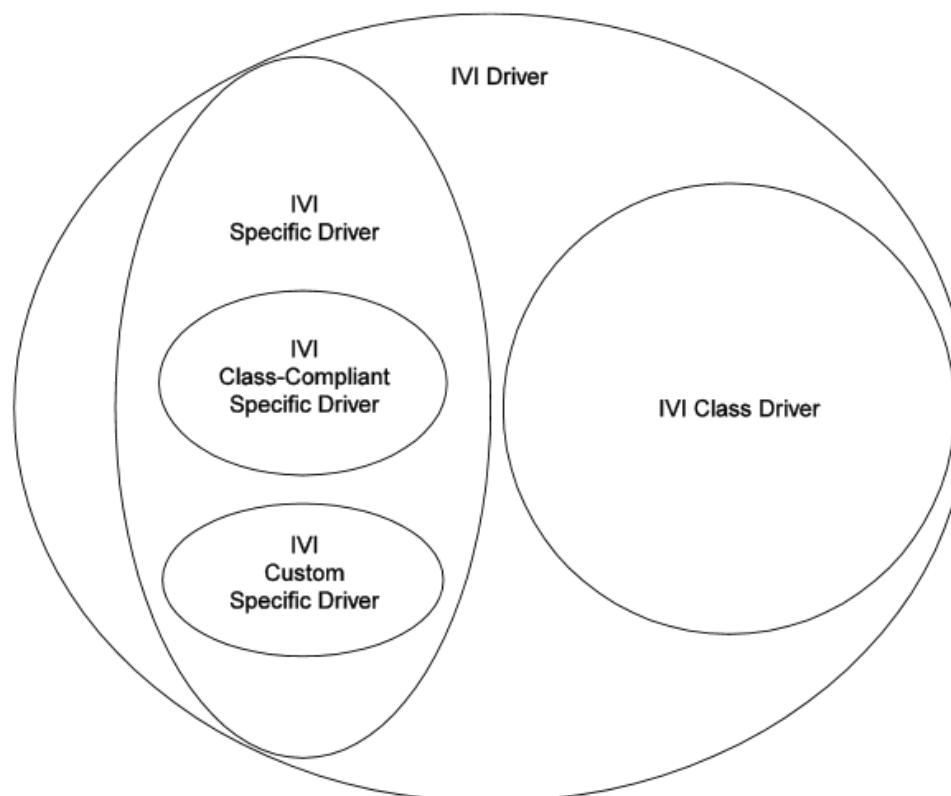
When to use interchangeability

- **Interchangeability can be used to reduce system down-time by allowing multiple options for hardware**
- **It can protect against hardware obsolescence**
- **It can allow cheaper hardware options to be considered**

When NOT to use interchangeability

- **If the hardware does not conform to an IVI class then interchangeability is not supported**
- **Interchangeability requires the use of the class driver, so any special features provided by any particular vendor cannot be used. If you need that special feature, you cannot use it via the interchangeability features of IVI**
- **Speed can be an issue. In some cases the IVI class driver can result in slower execution of applications**

Types of IVI driver



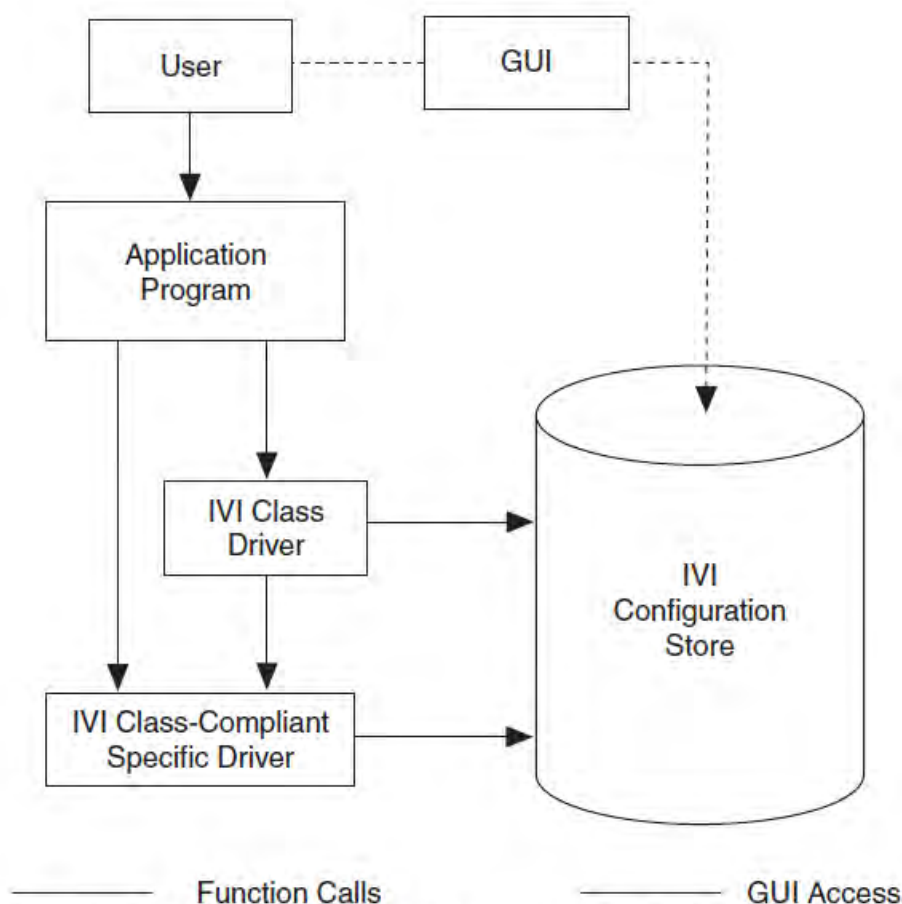
Note: When necessary to distinguish between API types, IVI specific drivers are further categorized by replacing "IVI" with "IVI-C", "IVI-COM", or "IVI.NET".

Figure 2-1.Types of IVI Drivers

Types of IVI Driver

- **As can be seen in the diagram, there are different types of IVI driver**
- **To achieve interchangeability the driver must be class compliant, that means it must support the minimum set of functions defined by the IVI Foundation specification for that hardware type**

Types of IVI Driver



A user program may simultaneously access the Class-Compliant and the Specific drivers.

However, use of the Specific driver may compromise interchangeability.

Figure 2-5. Using and IVI-C Class Compliant Specific Driver

Reproduced from IVI-3.1 Specification

IVI Switch

- **The IviSwtchBase capability group defines the following functions:**
 - Can Connect
 - **Connect**
 - **Disconnect**
 - **Disconnect All**
 - Get Channel Name (IVI-C only)
 - Get Path
 - Is Debounced (IVI-C only)
 - **Set Path**
 - Wait For Debounce

Only those functions in **red can control the state of a switch, a very limited set of functions.**

Pickering IVI-Swtch driver

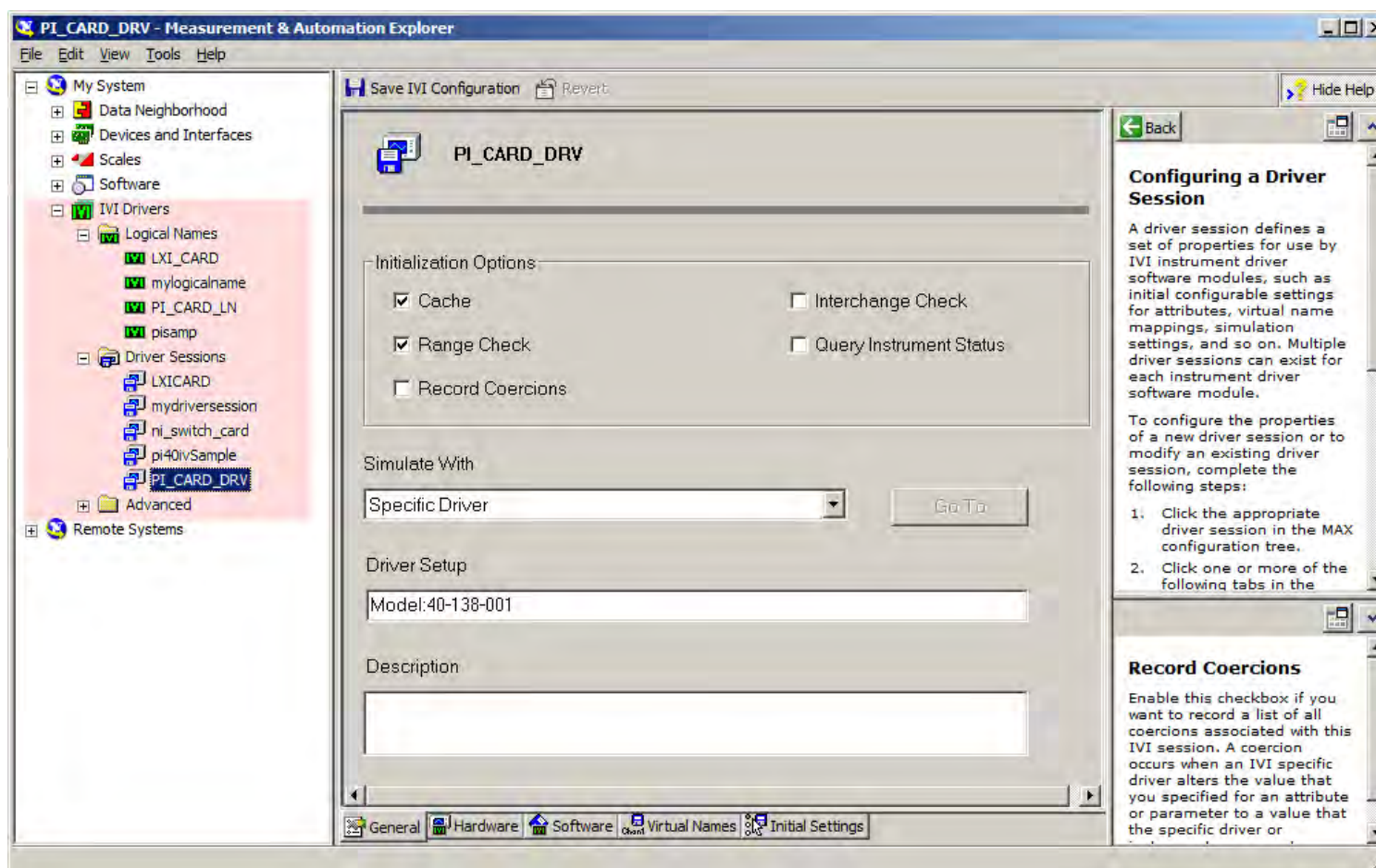
- **The Pickering IVI-Swtch driver is a class compliant specific driver. As such it supports all the base functions, but also provides further functions for more powerful control of Pickering switches**
- **These further functions cannot be used if interchangeability is required.**

Note: it is possible to access the specific driver via the class driver ,so providing access to the specific features. However, if any of the alternate products does not provide an equivalent feature, then complete interchangeability is probably impossible.

How to code for interchangeability

- **Key to interchangeability is the IVI Configuration Store**
- **This is an XML file which contains definitions for the IVI drivers and a layer to achieve interchangeability**
- **The most common tool for interaction with the IVI Configuration store is probably NI MAX, however other means are possible**

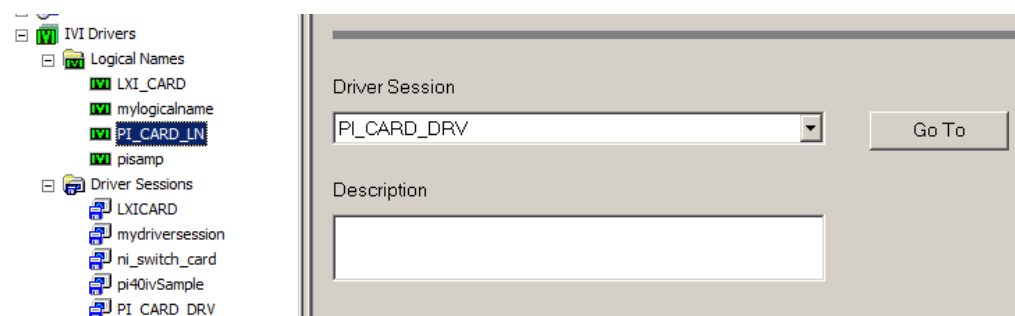
NI MAX and the IVI Configuration Store



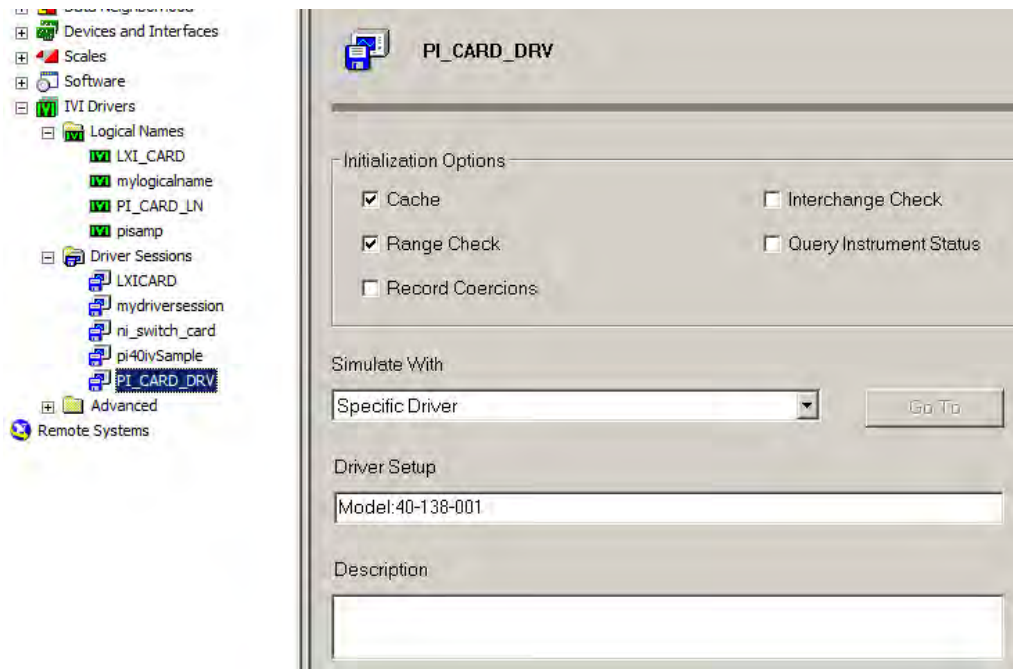
The IVI section is highlighted in pink

Explaining the IVI Configuration

- There are a couple of levels of indirection in the store, let's examine these to see what they allow us to do:
- **Logical Names**
 - A logical name is nothing more than a pointer to a Driver Session.



Driver Session

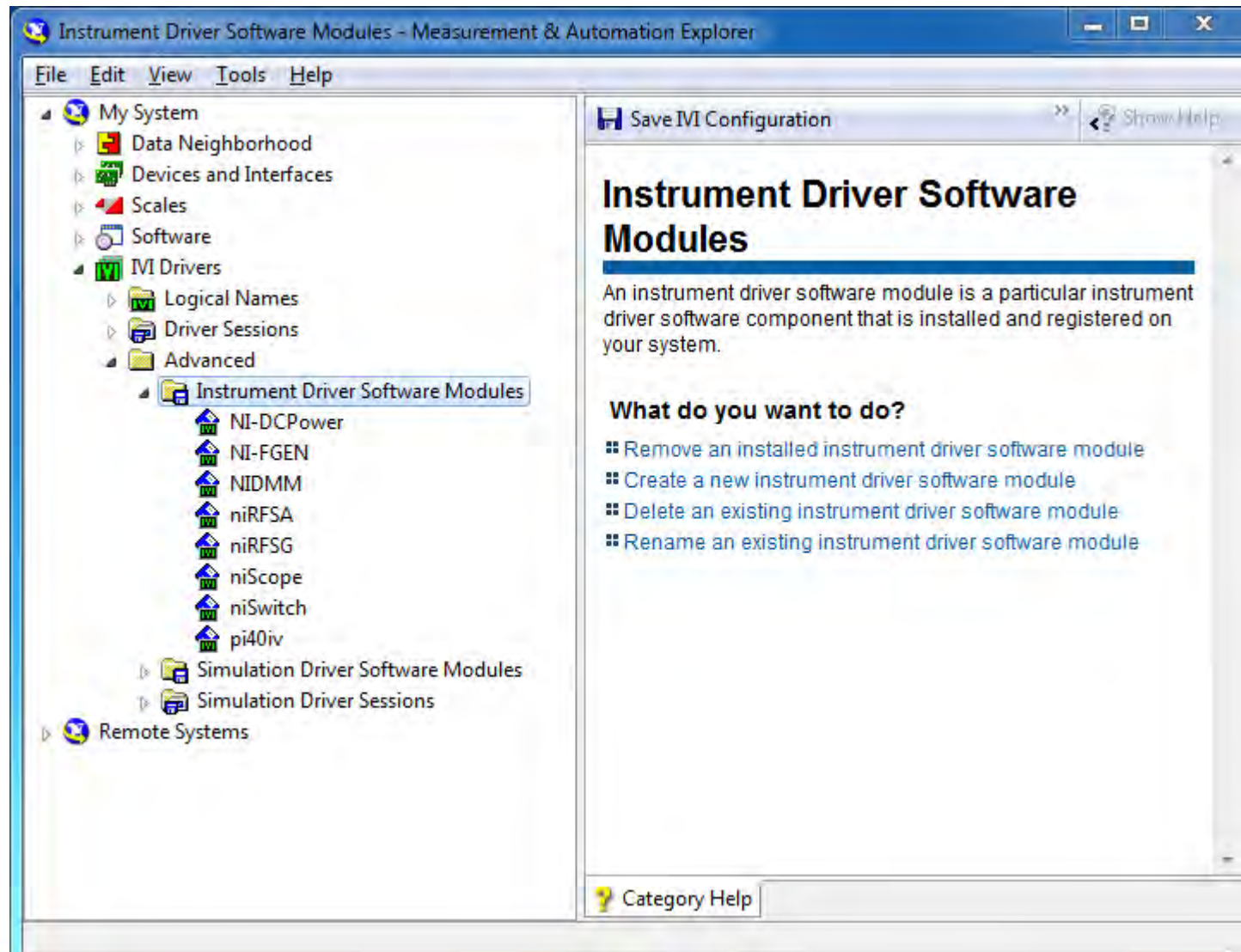


The driver session contains the specific parameters to be passed to the IVI driver 'InitWithOptions' function.

The driver session could be modified to refer to different hardware with no need to modify the users application

Drivers

The drivers available are also contained in the IVI Configuration Store. Driver Sessions are constructed around the available drivers.



pickering

Logical Name

- **So, the key steps for interchangeability are:**
 - **use the Logical Name instead of a hardware address**
 - **the Logical Name can be modified to refer to alternate Driver Sessions, this modification is done in the IVI Configuration Store, the user code is unaffected**

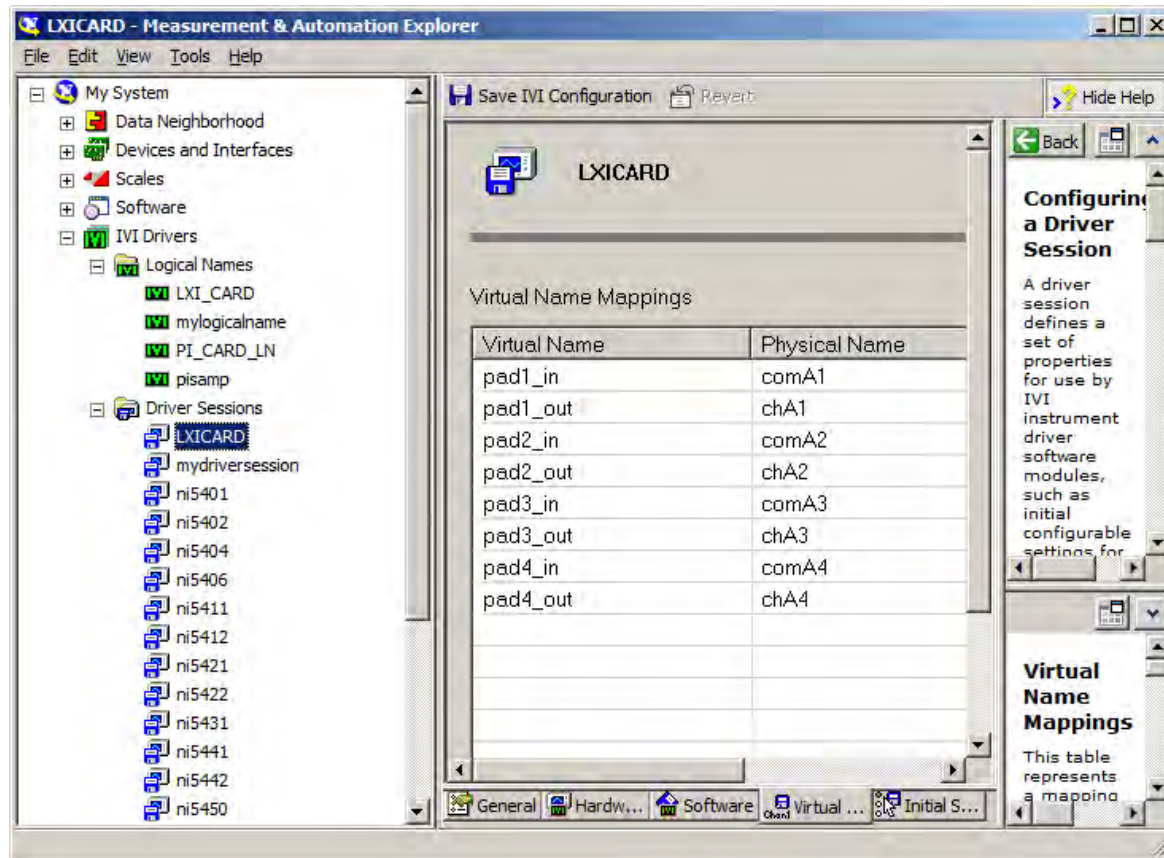
Logical Name

- **There are other areas that need some attention in order to achieve interchangeability**
 - **There is no specification for the coding of some of the driver capabilities, different drivers may use different nomenclature**
 - **For example: NI switch cards enumerate from a base of 0, PI cards from a base of 1**
NI: x0, x1, x2, y0, y1, y2
PI: x1, x2, x3, y1, y2, y3
- **A means of dealing with these differences is needed to achieve interchangeability**

Channel Names

- The IVI driver model treats a switch object like a 'black box'. The interface to the outside world is defined, but details of the internal operation is hidden.
- So, each switch is represented by its terminals and paths created by requesting connections between those terminals, for example:
 - `IviSwch_Connect(drv_session, "x1", "y1");`
- So, an IVI driver will present the user with a list of channels that it recognises,

Logical Name



The IVI Configuration Store provides a system of Virtual Names that allow users to alias channel names. This overcomes nomenclature differences between cards.

This is also a useful means to provide more meaningful channel names, even if interchangeability is not required.

Writing Code

- A programmer may be tempted to code like this:

```
err = pi40iv_InitWithOptions( "PXI5::15::INSTR",  
    VI_FALSE, VI_FALSE,  
    "Simulate=0,RangeCheck=1,QueryInstrStatus=1,Cache=1,DriverSetup=Model:  
41-182-003;", &vi);
```

```
Pi40iv_Connect(vi, "x0", "y0");
```

This offers NO interchangeability since:

- a) The hardware address is hard coded into the program
- b) The hardware model is hard coded into the program
- c) The custom driver is hard coded into the program
- d) The channel names may change on a different card

Writing Code

- **One step toward interchangeability:**

```
err = pi40iv_init("atten_In", 0, 0, &vi);
```

```
pi40iv_Connect(vi, "x0", "y0");
```

This offers some interchangeability since:

- a) The hardware address is no longer hard coded into the program
- b) The hardware model is no longer hard coded into the program
- c) BUT - the custom driver is hard coded into the program
- d) The channel names are still specific to the manufacturer

This would permit interchangeability of Pickering switch cards, but not interchangeability between vendors



Writing Code

- **Full interchangeability:**

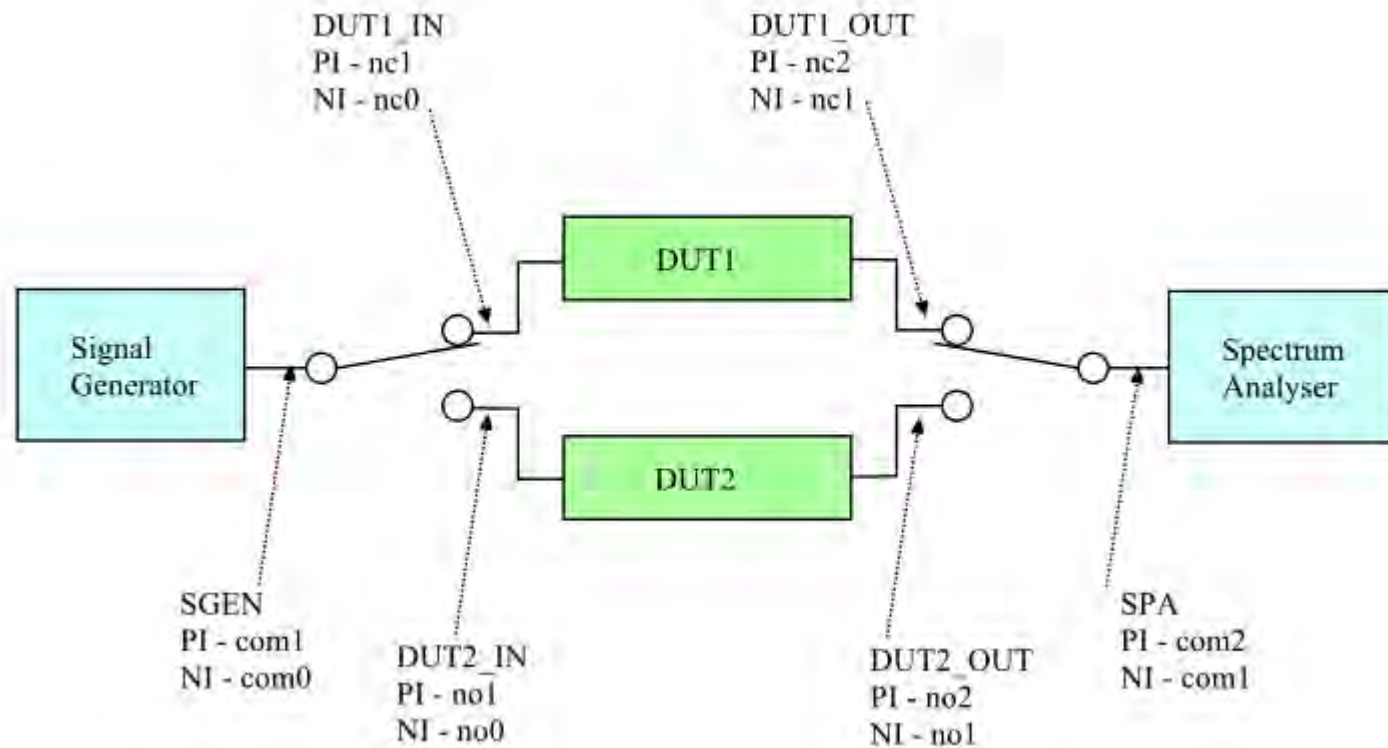
```
err = IviSwch_init("atten_In", 0, 0, &vi);  
  
IviSwch_Connect(vi, "DUT1", "DMM");
```

This offers full interchangeability since:

- a) The hardware address is no longer hard coded into the program
- b) The hardware model is no longer hard coded into the program
- c) The custom driver is no longer hard coded into the program
- d) Virtual channel names have been used and can be differently defined for different cards

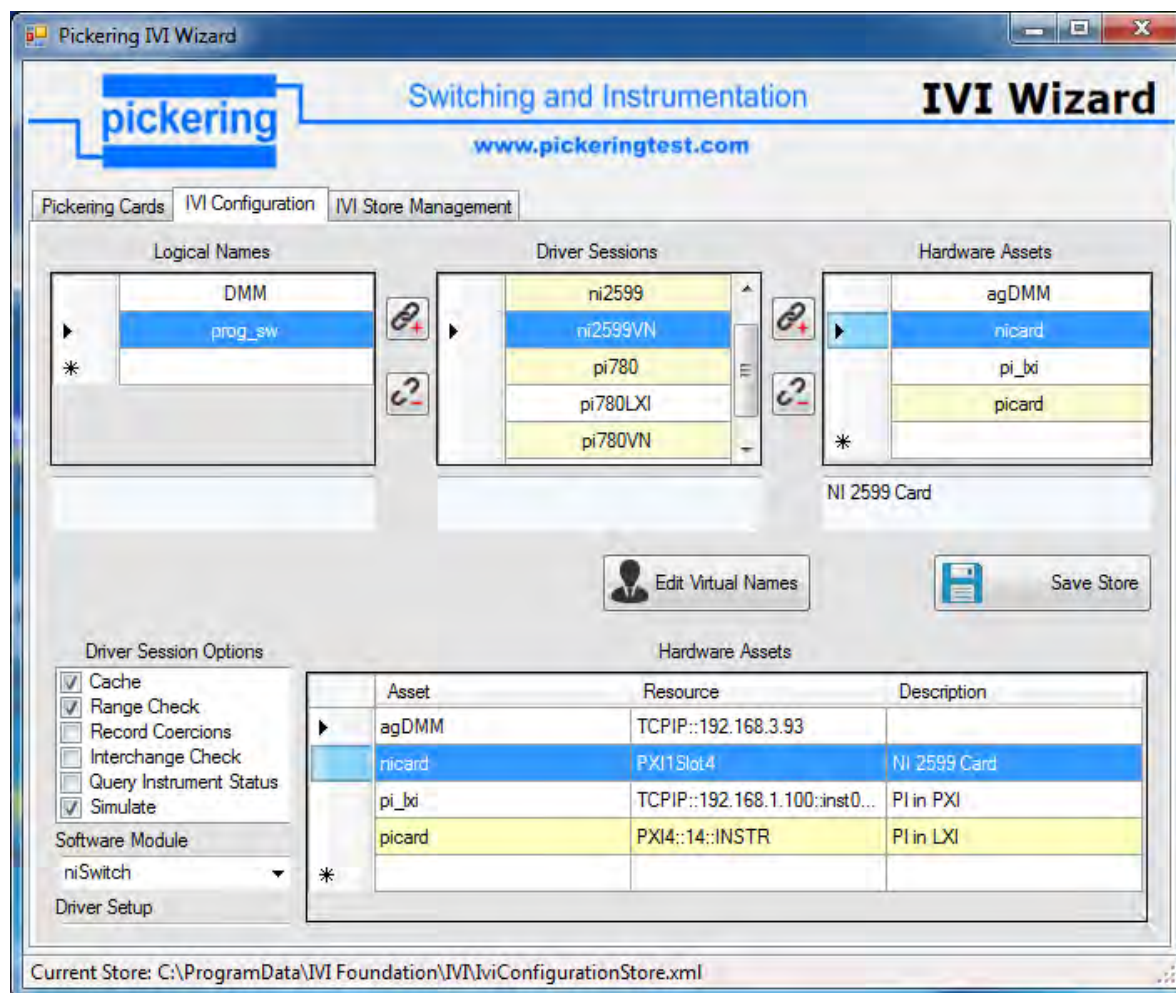
This would permit interchangeability of between vendors

Example – the hardware

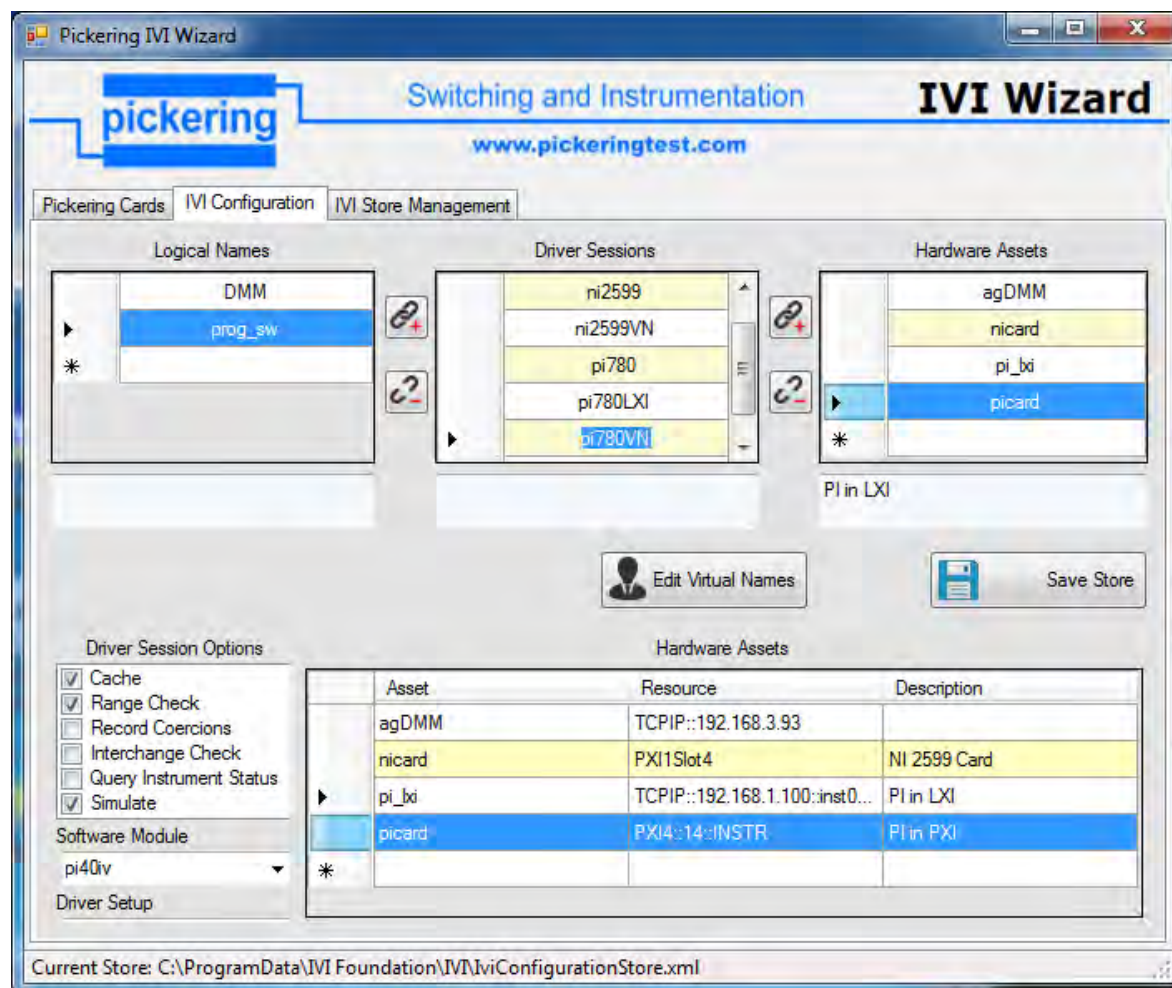


The designer has decided that either an NI PXI-2599 or a PI 40-780-522 would be suitable for this system; furthermore the PI card may be mounted in a PXI chassis or an LXI chassis. How do we make these devices interchangeable?

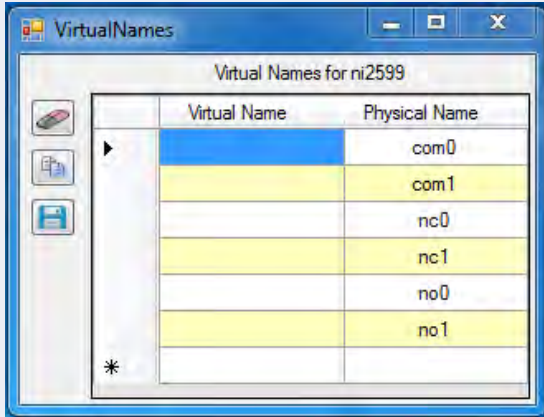
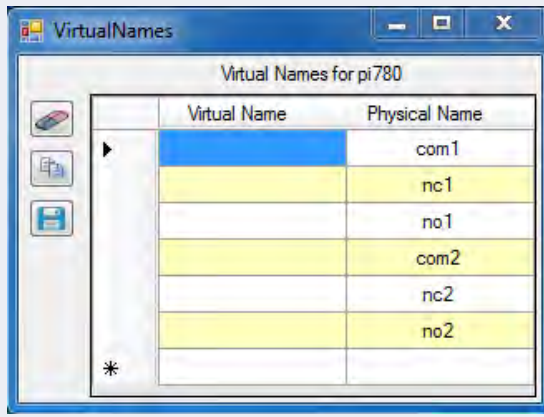
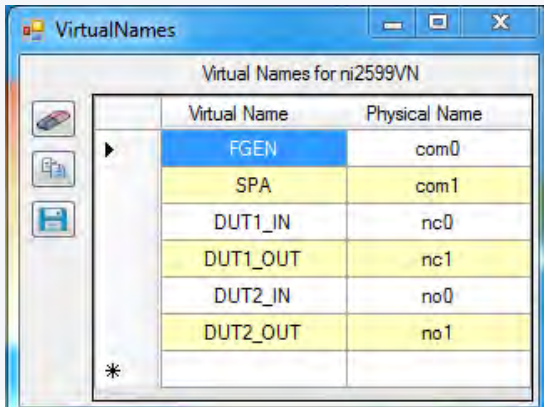
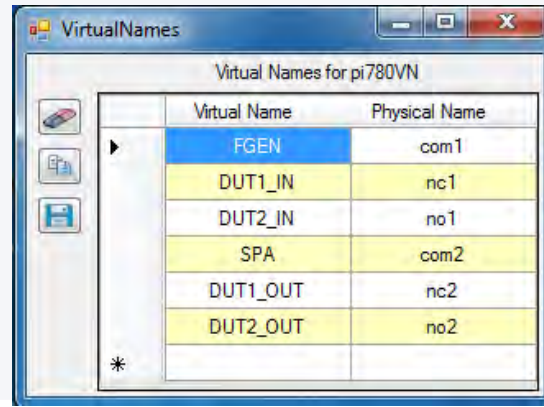
Example – the IVI Store – NI setup



Example – the IVI Store – PI setup



Example – looking from the IVI Driver

	NI	PI																												
Raw	 <p>Virtual Names for ni2599</p> <table><tr><th>Virtual Name</th><th>Physical Name</th></tr><tr><td></td><td>com0</td></tr><tr><td></td><td>com1</td></tr><tr><td></td><td>nc0</td></tr><tr><td></td><td>nc1</td></tr><tr><td></td><td>no0</td></tr><tr><td></td><td>no1</td></tr></table>	Virtual Name	Physical Name		com0		com1		nc0		nc1		no0		no1	 <p>Virtual Names for pi780</p> <table><tr><th>Virtual Name</th><th>Physical Name</th></tr><tr><td></td><td>com1</td></tr><tr><td></td><td>nc1</td></tr><tr><td></td><td>no1</td></tr><tr><td></td><td>com2</td></tr><tr><td></td><td>nc2</td></tr><tr><td></td><td>no2</td></tr></table>	Virtual Name	Physical Name		com1		nc1		no1		com2		nc2		no2
Virtual Name	Physical Name																													
	com0																													
	com1																													
	nc0																													
	nc1																													
	no0																													
	no1																													
Virtual Name	Physical Name																													
	com1																													
	nc1																													
	no1																													
	com2																													
	nc2																													
	no2																													
With Virtual Names	 <p>Virtual Names for ni2599VN</p> <table><tr><th>Virtual Name</th><th>Physical Name</th></tr><tr><td>FGEN</td><td>com0</td></tr><tr><td>SPA</td><td>com1</td></tr><tr><td>DUT1_IN</td><td>nc0</td></tr><tr><td>DUT1_OUT</td><td>nc1</td></tr><tr><td>DUT2_IN</td><td>no0</td></tr><tr><td>DUT2_OUT</td><td>no1</td></tr></table>	Virtual Name	Physical Name	FGEN	com0	SPA	com1	DUT1_IN	nc0	DUT1_OUT	nc1	DUT2_IN	no0	DUT2_OUT	no1	 <p>Virtual Names for pi780VN</p> <table><tr><th>Virtual Name</th><th>Physical Name</th></tr><tr><td>FGEN</td><td>com1</td></tr><tr><td>DUT1_IN</td><td>nc1</td></tr><tr><td>DUT2_IN</td><td>no1</td></tr><tr><td>SPA</td><td>com2</td></tr><tr><td>DUT1_OUT</td><td>nc2</td></tr><tr><td>DUT2_OUT</td><td>no2</td></tr></table>	Virtual Name	Physical Name	FGEN	com1	DUT1_IN	nc1	DUT2_IN	no1	SPA	com2	DUT1_OUT	nc2	DUT2_OUT	no2
Virtual Name	Physical Name																													
FGEN	com0																													
SPA	com1																													
DUT1_IN	nc0																													
DUT1_OUT	nc1																													
DUT2_IN	no0																													
DUT2_OUT	no1																													
Virtual Name	Physical Name																													
FGEN	com1																													
DUT1_IN	nc1																													
DUT2_IN	no1																													
SPA	com2																													
DUT1_OUT	nc2																													
DUT2_OUT	no2																													

Example – the test program

// ivi_example.cpp : Example of IVI interchangeability

```
#include "stdafx.h"
#include "iviswtdch.h"
```

```
int main(int argc, char *argv[])
{
```

```
    ViSession vi = 0;
    ViStatus error = 0;
```

```
    checkErr( IviSwtdch_init("prog_sw",
                           VI_FALSE, VI_FALSE, &vi) );
```

```
    IviSwtdch_DisconnectAll(vi);
```

```
    checkErr( IviSwtdch_Connect(vi, "SGEN", "DUT1_IN") );
    checkErr( IviSwtdch_Connect(vi, "DUT1_OUT", "SPA") );
```

```
    Go_Do_A_Test("DUT1");
```

```
    IviSwtdch_DisconnectAll(vi);
```

```
    checkErr( IviSwtdch_Connect(vi, "SGEN", "DUT2_IN") );
    checkErr( IviSwtdch_Connect(vi, "DUT2_OUT", "SPA") );
```

```
    Go_Do_A_Test("DUT2");
```

```
    IviSwtdch_DisconnectAll(vi);
```

```
    if (error != VI_SUCCESS) printf("Error: %08x\n", error);
```

```
    IviSwtdch_close(vi);
```

```
    return EXIT_SUCCESS;
```

```
}
```

(IviSwtdch_init("prog_sw",

// initialize a session on the switch

// make sure all switches are at default

// connect SGEN

// connect SPA

// Perform test on DUT1

// reset all switches

// connect SGEN

// connect SPA

// Perform test on DUT2

// reset all switches

// close the session on the switch

Note use of
class driver,
logical names,
and virtual
names making
this code
interchangeable

IviSwtdch_Connect(vi, "SGEN", "DUT1_IN")



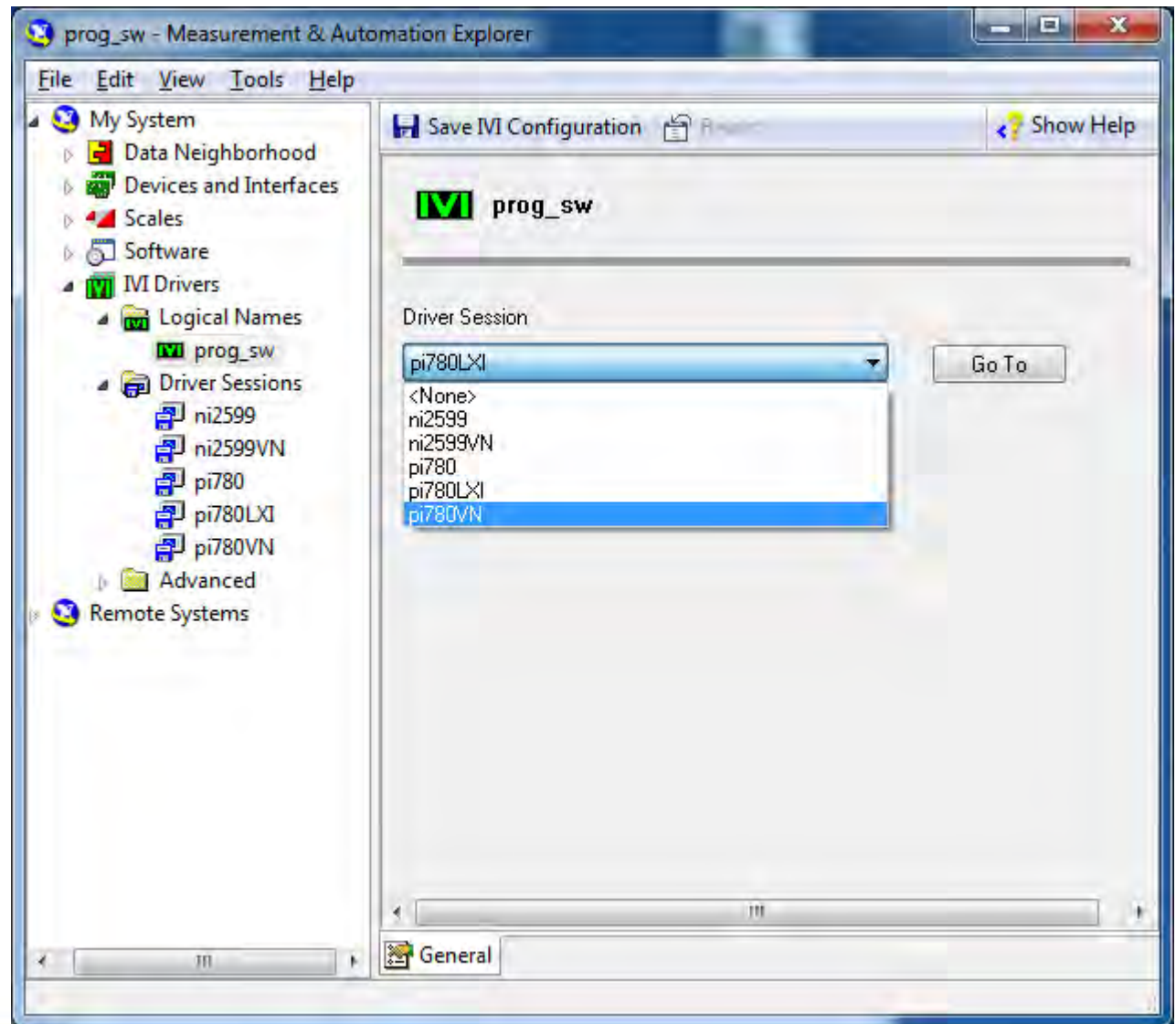
Example – the test program

At any time the IVI Configuration Store may be edited to modify the Device Driver referenced by the Logical Name.

This permits full interchangeability between the 3 different switches.

If a further switch type is required, all that has to be done is to create a Device Driver for that hardware, matching the Virtual Names, then modify the Logical Name to refer to the new Driver Session.

**NO CODE MODIFICATION
REQUIRED**



Summary

- **‘Under the bonnet’ the IVI system handles the interchangeability.**
- **When program calls are made, the IVI system calls the appropriate driver with the appropriate parameters**
- **If multiple Driver Sessions are created for different hardware solutions, then all the user has to do is to modify the Logical Name to point to the alternate Driver Session**

Remember

- **Be sure that interchangeability will not compromise performance or capability**
- **Choose products that offer an IVI Class-Compliant driver**
- **Program using the IVI Class Driver, not vendor provided drivers**
- **Manage the differences between cards in the IVI Configuration Store, not in the user code**