IVI-COM Instrument Driver Programming Guide (LabVIEW Edition)

Sep 2005 Revision 2.0

1- Overview

1-1 Supporting IVI-C Drivers

LabVIEW has a capability to import VXI Plug&Play and/or IVI-C instrument drivers. Although it is possible to use IVI-COM instrument drivers directly, it is much easier to program utilising IVI-C or VXI Plug&Play instrument drivers if they are supported. IVI-COM instrument drivers from Kikusui provide not only IVI-COM interfaces, but also support IVI-C programming interfaces. The IVI-C specifications are the advanced versions of VXI Plug & Play instrument driver specifications, therefore they are the most suitable driver types for use with LabVIEW.

Notes:

IVI-COM instrument drivers from Kikusui do support both IVI-COM and IVI-C interfaces, as long as the driver version is 2.x.x.x or later. Mind that any versions 1.x.x.x or prior do not support IVI-C.

To use IVI-C instrument drivers, you must separately install NI IVI Compliance Package 2.x. This software package is not automatically installed with Kikusui's IVI driver setup program. Also, not all the LabVIEW versions do install it.

This guidebook assumes that you use IVI-COM Kikusui4800 instrument driver (for KIKUSUI PIA4800 series DC Power Supply Controller). You can also use IVI-COM instrument drivers for other models in the same manner.

This guidebook assumes that you use LabVIEW 7.1.

When using an IVI instrument driver, there are two approaches – using specific interfaces and using class interfaces. The former is to use interfaces that are specific to an instrument driver and you can utilise the most of features of the instrument you use. The later is to utilise instrument class interfaces that are defined in the IVI specifications allowing to utilise interchangability features, but instrument specific features are restricted.

Notes:

The instrument class to which the instrument driver belongs is documented in Readme.txt for each of drivers. The Readme document can be viewed from Start button➔Program➔IVI folder.

If the instrument driver does not belong to any instrument classes, you can't utilise class interfaces. This means that you cannot develop applications that utilise interchangability features.

2- Example Using Specific Interfaces

Here we introduce an example using specific interfaces. By using specific interfaces, you can utilise the maximum power of driver features but you have to spoil interchangability.

2-1 Importing Instrument Driver

Because IVI-C and VXI Plug&Play instrument drivers are provided as the LabWindows/CVI compatible format (fp, c, h, DLL, etc...), they cannot be directly used from within LabVIEW

©2005 KIKUSUI ELECTRONICS CORP. All Rights Reserved.

environment. Therefore, it is necessary to convert the driver interface information to LabVIEW-compatible format (vi or llb) by importing them.

To import an instrument driver, choose **Tools | Instrumentation | Import CVI Instrument Driver** menu on LabVIEW. As you will be prompted to enter an fp (CVI Function Panel) file, select **ki4800.fp** located in the **Program Files/IVI/Drivers/ki4800** directory. Then the CVI Function Panel Converter dialog appears. Confirm the destination of the file conversion, DLL file name, and others as need (normally default is acceptable), then click the **OK** button.

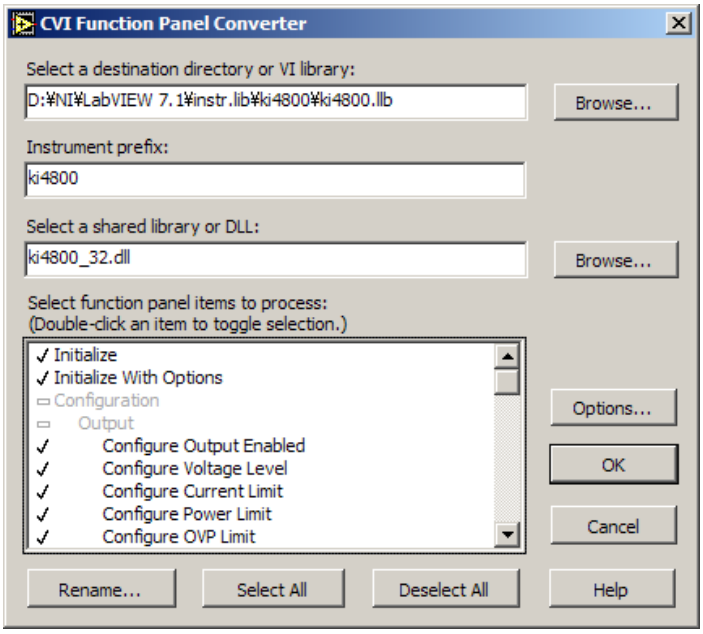


Figure 2-1 CVI Function Panel Converter

After the conversion complete, ki4800.llb (and some other support files) will be generated in the **instr.lib** sub directly under the LabVIEW installation place. They are the instrument driver wrapper, which can be directly used from LabVIEW.

The ki4800 instrument driver wrapper can be referenced through the **Instrument I/O** function palette on the Block Diagram window.

Note:

The llb file that was generated by the import work is just a driver wrapper, and not the real instance of the instrument driver. Therefore, the real driver instance (DLL) is still required to install at application runtime.

If the LabVIEW 7.1's add-on software "LabVIEW Interface Generator for LabWindows/CVI Instrument Drivers" is installed, there are some differences for operation on the CVI Function Panel Converter dialog.

2-2 Adding Controls and Functions

First, create a new application. Open the Front Panel window, place an **error in** cluster and an **error out** cluster.

©2005 KIKUSUI ELECTRONICS CORP. All Rights Reserved.

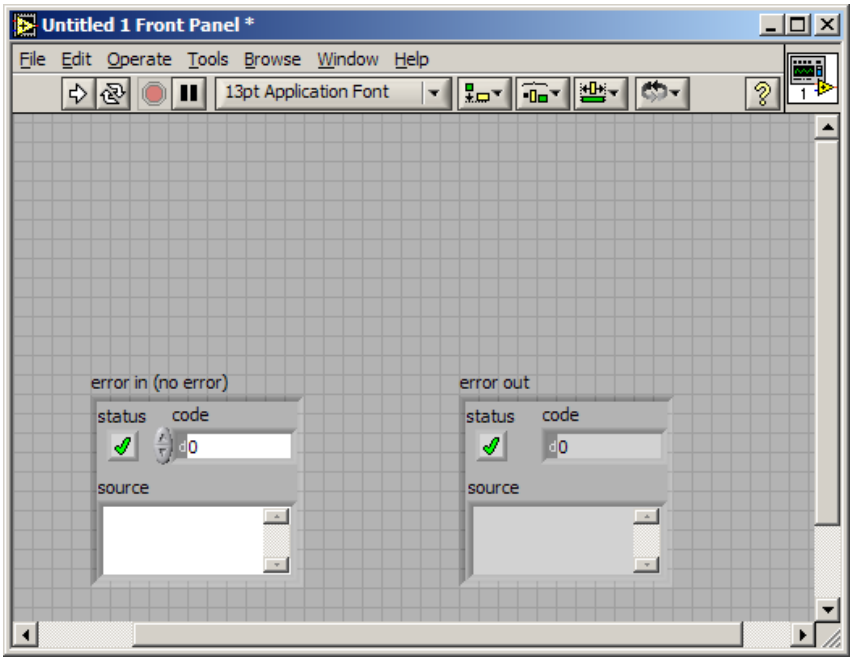


Figure 2-2 Front Panel

Next, open the Block Diagram window, then open the function palette for the ki4800 driver (wrapper). The function palette will be found through the context menu → **Instrument I/O → Instrument Drivers → ki4800**.

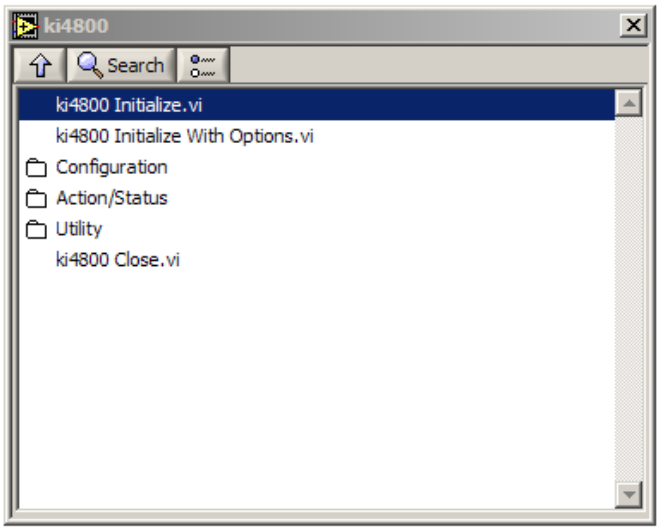


Figure 2-3 ki4800 Function Palette

Place `Initialize With Options.vi` and `Close.vi` on the Block Diagram. Furthermore, add `Configure Voltage Level.vi`, `Configure Current Limit.vi`, and `Configure Output Enabled.vi`, which are found in the **Configuration→Output** palette.

©2005 KIKUSUI ELECTRONICS CORP. All Rights Reserved.

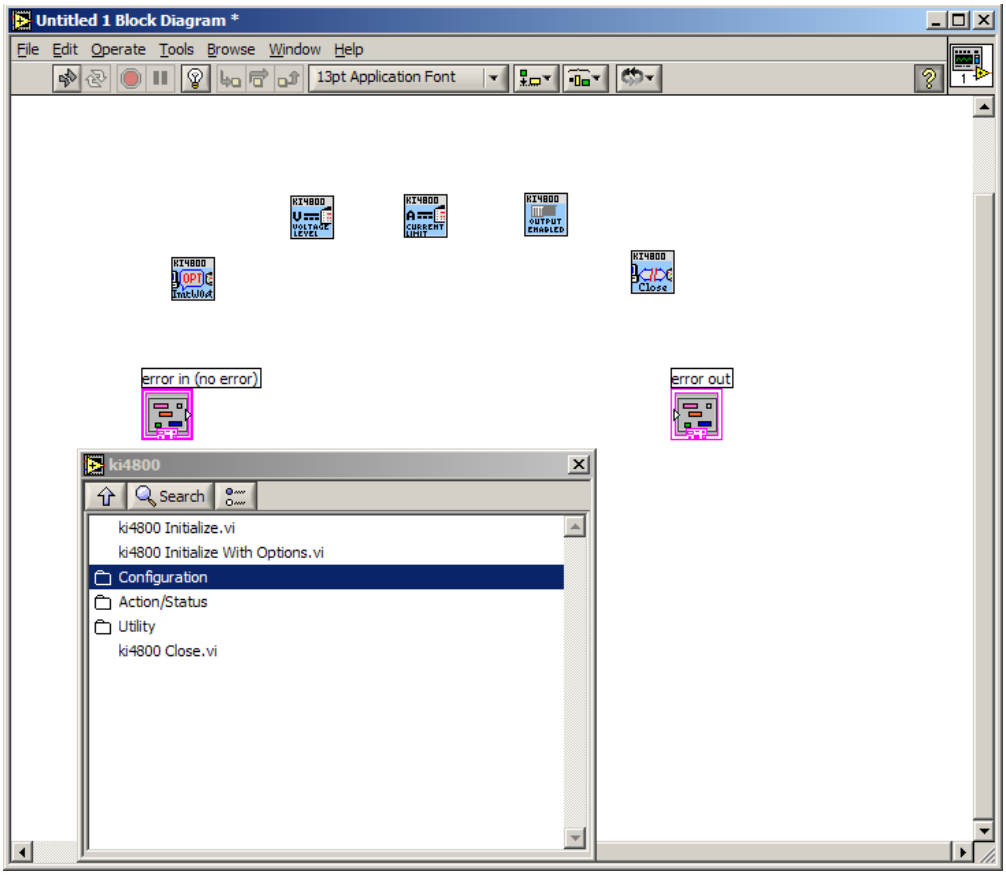


Figure 2-4 Block Diagram

2-3 Parameter Settings

Assuming the PIA4800 series Power Supply Controller is configured as GPIB address 3, pass the parameters -- `resource name`, `id query`, `reset device` to `Initialize With Options.vi`.

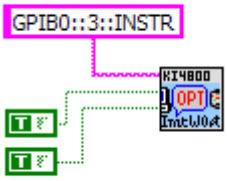
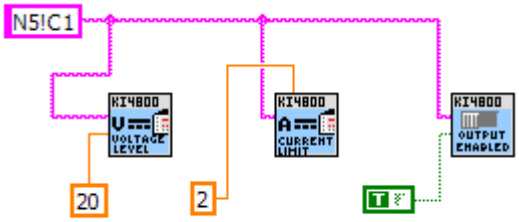


Figure 2-5 Params for Initialize With Options

Subsequently, add parameters that set voltage, current, and output. Here, we set 20V/2A and the output ON.



©2005 KIKUSUI ELECTRONICS CORP. All Rights Reserved.

Figure 2-6 Params for Configure Functions

Mind the string -- `"N5!C1"`. This is the target channel name for the DC power supply to be controlled. Details are described later.

Finally, wire the controls/functions between **error in** and **error out** clusters as like the picture below. Not only connecting error ins/outs, but make sure to connect instrument session (handle) wires also.

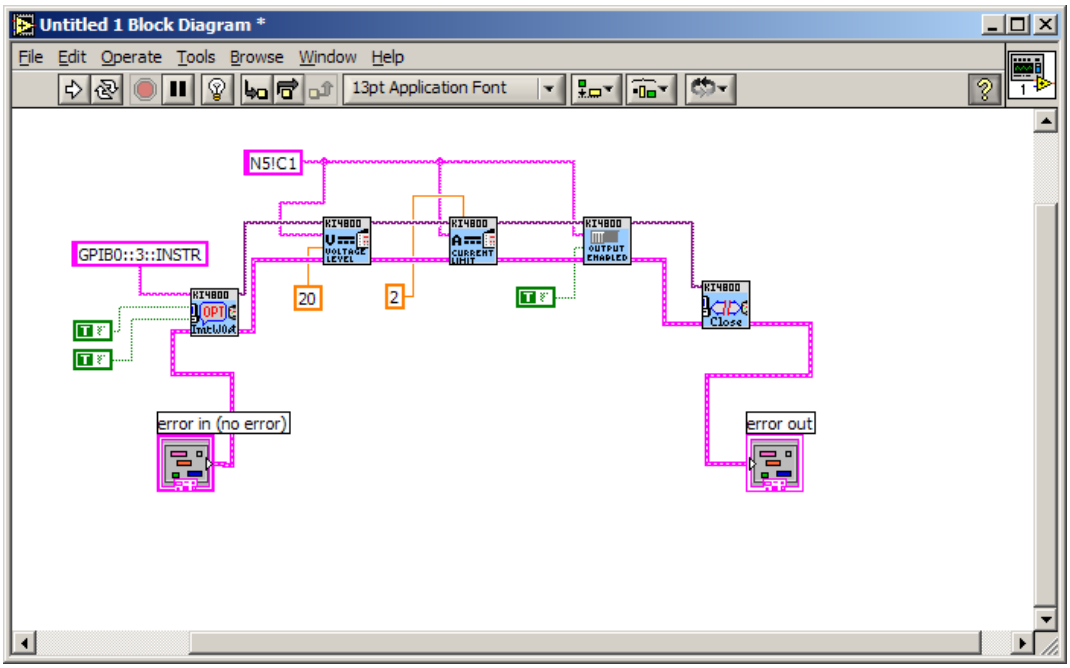


Figure 2-7 Open/Configure/Close

2-4 Program Execution

You can execute the previous codes for the time being. `Initialize With Options.vi` resets the instrument settings since the `Reset Device` parameter is set to TRUE. As you execute the program, I/O communications with the instrument immediately starts. If the instrument is actually connected and the `Initialize With Options/Close` succeeds, the error code shown on the **error out** cluster will be 0. If a communication problem has occurred or the VISA library is not configured properly, an exception is generated and its information will be shown on the **error out** cluster. By selecting Context Menu ➔**Explain Error**, you can see the detail information on the **explain error** dialogue.

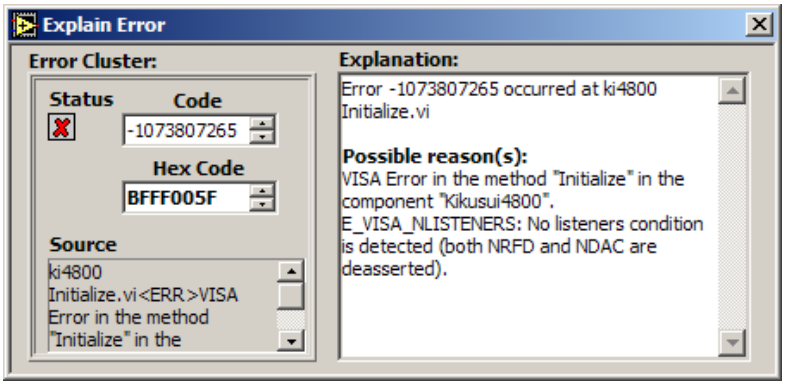


Figure 2-8 Runtime Error

Note:

The KIKUSUI PIA4800 series DC Power Supply Controller requires a couple of minutes to detect connectivity conditions of the DC power supplies through the TP-BUS. It should not be performed at

©2005 KIKUSUI ELECTRONICS CORP. All Rights Reserved.

every Initialize call. Therefore, it is necessary to configure the connectivity conditions of the DC power supplies to be controlled in advance by using Scan Utility. Make sure to run the Scan Utility when the first time to use the driver. To run the utility, choose [Start] button → All Programs → IVI → Kikusui4800 → Scan Utility menu.

You will need to run Scan Utility once again if you have changed number of DC power supplies, node addresses, communication interfaces (GPIB/RS232), and/or the port number or the GPIB address.

The configuration work with Scan Utility is specific to the IVI-COM Kikusui4800(ki4800) driver. No need to do it for other instrument drivers.

3- Description

3-1 Opening Session

To open the driver session, the `ki4800 Initialize With Options.vi` is used. Although the prefix `ki4800`, which is applied to the vi (function) is different depending for each instrument driver, this naming convention is applied for all the IVI-C instrument drivers.

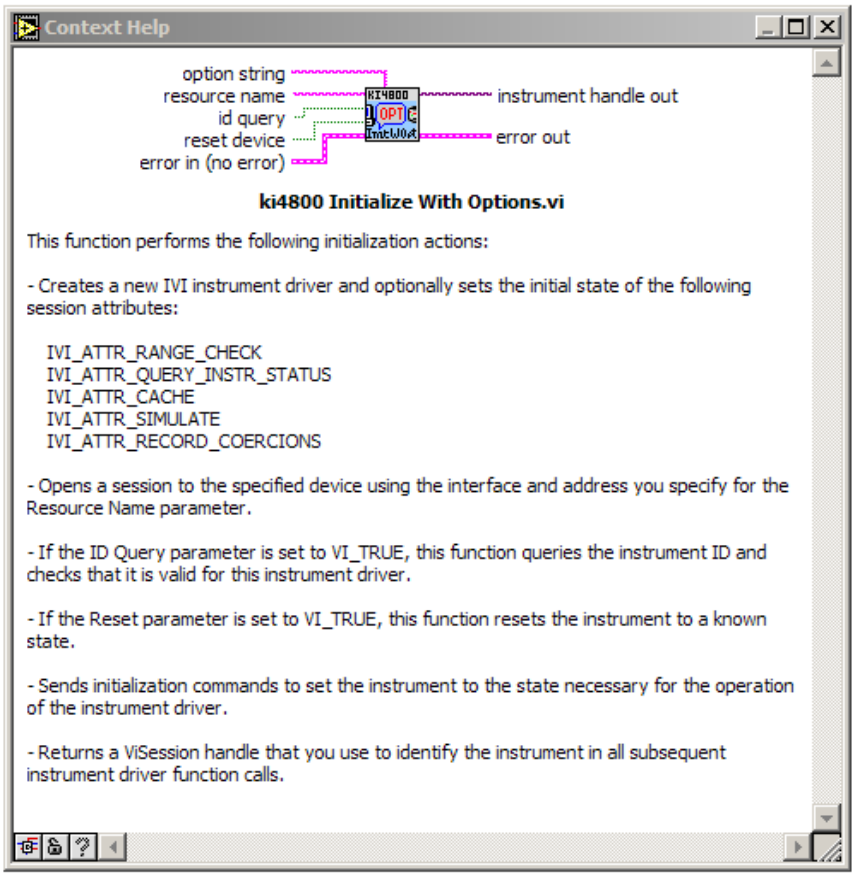


Figure 3-1 Initialize With Options.vi Help

Notes:

As a technical term of IVI-C and VXI Plug&Play instrument drivers, "<prefix>" is often used. This is a name that identifies each of specific instrument drivers, and "ki4800" is the one in this document. For example, a generic expression <prefix> Initialize.vi, designates the ki4800 Initialize.vi for the ki4800 instrument driver.

Every vi (function), except for <prefix> Initialize.vi and <prefix> Initialize With Options.vi, takes the upper left input parameter as **instrument handle in**.

©2005 KIKUSUI ELECTRONICS CORP. All Rights Reserved.

Every vi (function), except for <prefix> Close.vi, takes the upper right output parameter as **instrument handle out**, which should be wired to the **instrument handle in** parameter on the next vi.

The <prefix> Initialize.vi is remained for the compatibility to VXI Plug&Play driver specifications. The function is equivalent with <prefix> Initialize With Options.vi with an exception that the **option string** parameter cannot be specified.

Now let 's talk about parameters of the `ki4800 Initialize With Options.vi`. Every IVI instrument driver has an `Initialize With Options.vi`, which is defined by the IVI specifications. This function has the following parameters.

Table 3-1 Parameters for Initialize With Options

Parameter	Type	Description
`Resource Name`	`String`	VISA resource name string. This is decided according to the I/O interface and/or address through which the instrument is connected. If the instrument has the address 3 on the GPIB board #0, for example, it can be `GPIB0::3::INSTR`.
`Id Query`	`Boolean`	Specifying `VI_TRUE` performs ID query to the instrument.
`Reset Device`	`Boolean`	Specifying `VI_TRUE` resets the instrument settings.

`Option String`	`String`	Overrides the following settings instead of default:
		`RangeCheck`
		`Cache`
		`Simulate`
		`QueryInstrStatus`
		`RecordCoercions`
		`Interchange Check`
		Furthermore you can specify driver-specific options if the driver supports `DriverSetup` features.

`ResourceName` specifies a VISA address (resource name). If `Id Query` is VI_TRUE, the driver queries the instrument identities using a query command such as `"*IDN?"`. If `resetDevice` is VI_TRUE, the driver resets the instrument settings using a reset command such as `"*RST"`.

`Option String` has two features. One is what configures IVI-defined behaviours such as `RangeCheck`, `Cache`, `Simulate`, `QueryInstrStatus`, `RecordCoercions`, and `Interchange Check`. Another one is what specifies `DriverSetup` that may be differently defined by each of instrument drivers. Because the `Option String` is a string parameter, these settings must be written as like the following example:

```
QueryInstrStatus = TRUE , Cache = TRUE , DriverSetup=12345
```

Names and setting values for the features being set are case-insensitive. Since the setting values are ViBoolean type, you can use any of `VI_TRUE`, `VI_FALSE`, `1`, and `0`. Use commas for splitting multiple items. If an item is not explicitly specified in the `OptionString` parameter, the IVI-defined default value is applied for the item. The IVI-defined default values are VI_TRUE for `RangeCheck` and `Cache`, and VI_FALSE for others.

Some instrument drivers may have special meanings for the `DriverSetup` parameter. It can specify items that are not defined by the IVI specifications when invoking the `InitializeWithOptions` function, and its purpose and syntax are driver-specific. Therefore, specifying the `DriverSetup` must be at the last part on the `Option String` parameter. Because the contents of `DriverSetup` are different depending on each driver, refer to driver's Readme document or online help.

©2005 KIKUSUI ELECTRONICS CORP. All Rights Reserved.

3-2 Channel Access

When supporting power supply and/or electronic load instruments, the IVI instrument driver is generally designed assuming the instrument has multiple channels. Therefore, driver functions operating instrument panel settings often have the **channel name** parameter, which specifies the channel.

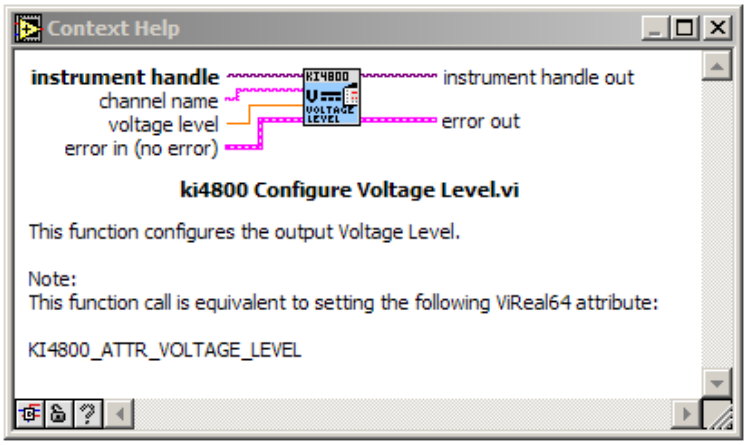


Figure 3-2 Configure Voltage Level.vi Help

As this document uses the Kikusui4800 (ki4800) driver that operates the DC power supply, we use a channel name that contains the NODE and CHANNEL. The channel name `"N5!C1"` in the above example is specific to an instrument driver, therefore different naming convention is applied on driver basis. Refer to the driver's on-line help for what channel names can be actually used.

This example sets 20V for the DC power supply, which is connected at NODE5 and CHANNEL1 of the PIA4800 series DC Power Supply Controller.

3-3 Closing Session

To close the instrument driver session, use the `ki4800 Close.vi` function.

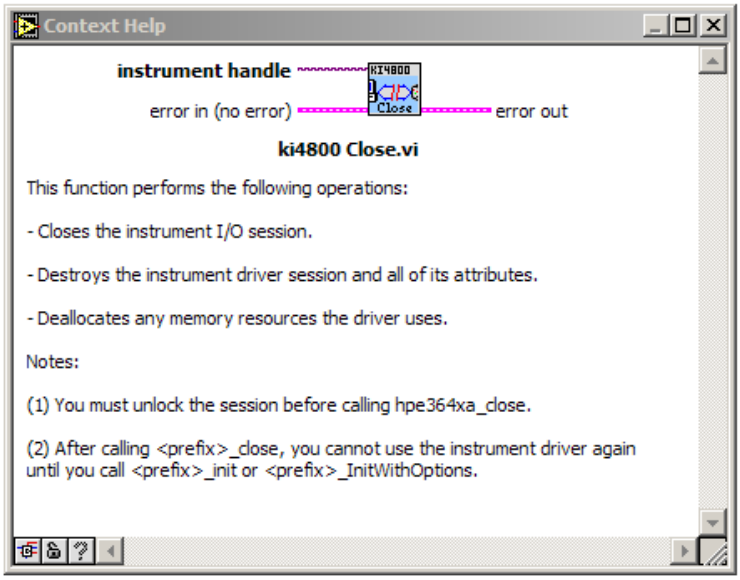


Figure 3-3 Close.vi Help

©2005 KIKUSUI ELECTRONICS CORP. All Rights Reserved.

4- Example Using Class Interfaces

Now we explain how to use class interfaces. By using class interfaces, you can swap the instruments without recompiling/relinking your application codes. In this case, however, IVI-C instrument drivers for both pre-swap and post-swap models must be provided, and these drivers both must belong to the same instrument class. There is no interchangability available between different instrument classes.

4-1 Virtual Instrument

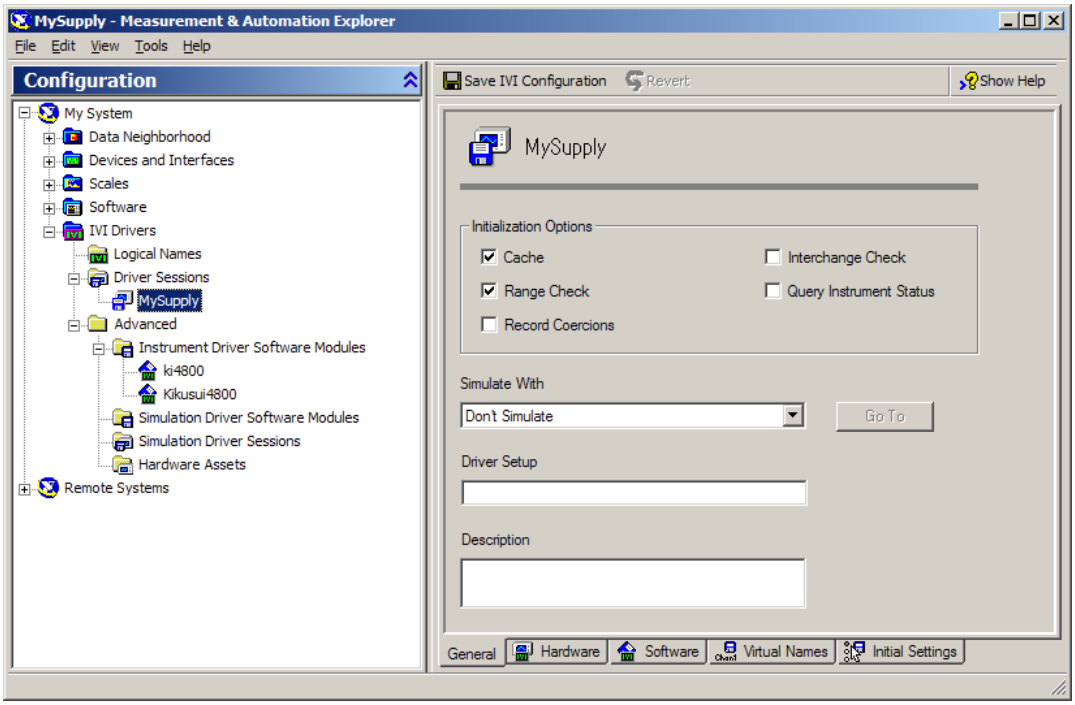
What you have to do before creating an application that utilises interchangability features is create a virtual instrument. To realise interchangability features, you should not write codes that are very specific to a particular IVI-C instrument driver (e.g. invoking the ki4800_init() function) and should not write a specific VISA address (resource name) such as "GPIB0::3::INSTR". Writing them directly in the application spoils interchangability.

Instead, the IVI specifications define methods to realise interchangability by placing the external IVI Configuration Store. The application indirectly selects an instrument driver according to contents of the IVI Configuration Store, and accesses the indirectly loaded driver through the class driver that has no dependency to specific instrument models.

The IVI Configuration Store is normally /Program Files/IVI/Data/IviConfigurationStore.XML file and is accessed through the IVI Configuration Server DLL. This DLL is mainly used by IVI instrument drivers and some VISA/IVI configuration tools, not by end-user applications. When using LabWindows/CVI, the NI-MAX (NI Measurement and Automation Explorer) software provided by National Instruments allows you to perform IVI driver configurations.

Creating Driver Session

After launching NI-MAX, refer to the **IVI Drivers** node on the tree. Right-click on the **Driver Session** then select **Create New** menu to create a new Driver Session. Being asked for its name, give the name "`MySupply`".

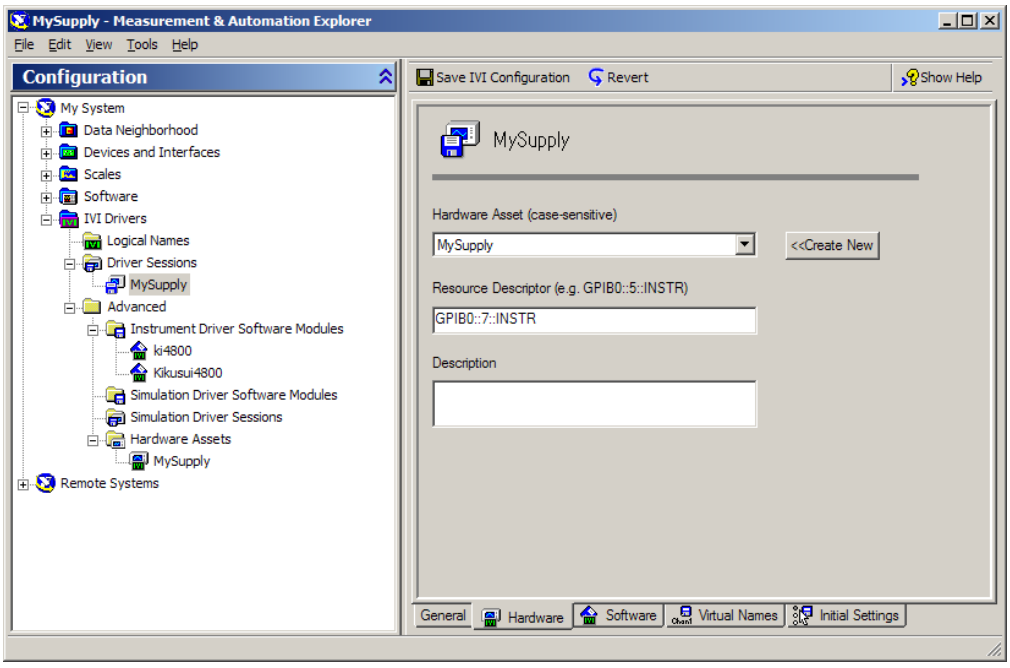


Creating Hardware Asset

Subsequently select the **Hardware** tab to show the hardware asset management screen. The hardware asset specifies what interface route your actual instrument is connected

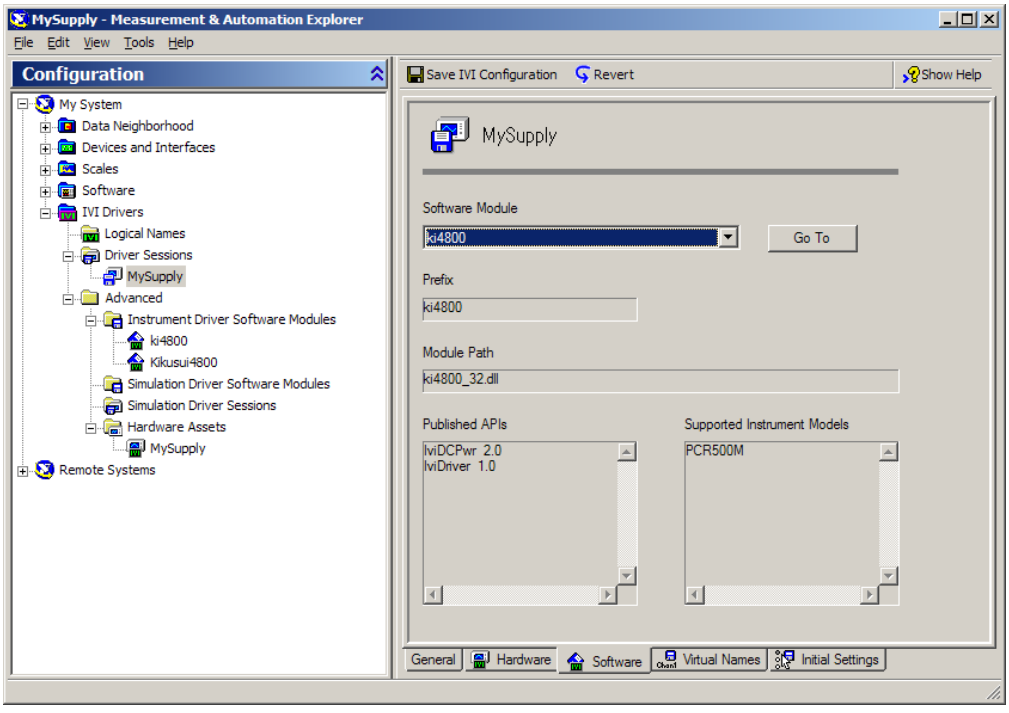
©2005 KIKUSUI ELECTRONICS CORP. All Rights Reserved.

through. Here you click the **Create New** button to create a new Hardware Asset. Being asked for its name, give the name "MySupply" again, then click the **Create** button. Furthermore specify a valid VISA address though which your instrument is connected, as **Resource Descriptor**.



Setting Linkage for Software Module

Subsequently select the **Software** tab to show the software module management screen. The software module specifies the instrument driver module (DLL module). Here select **ki4800** from the **Software Module** list.

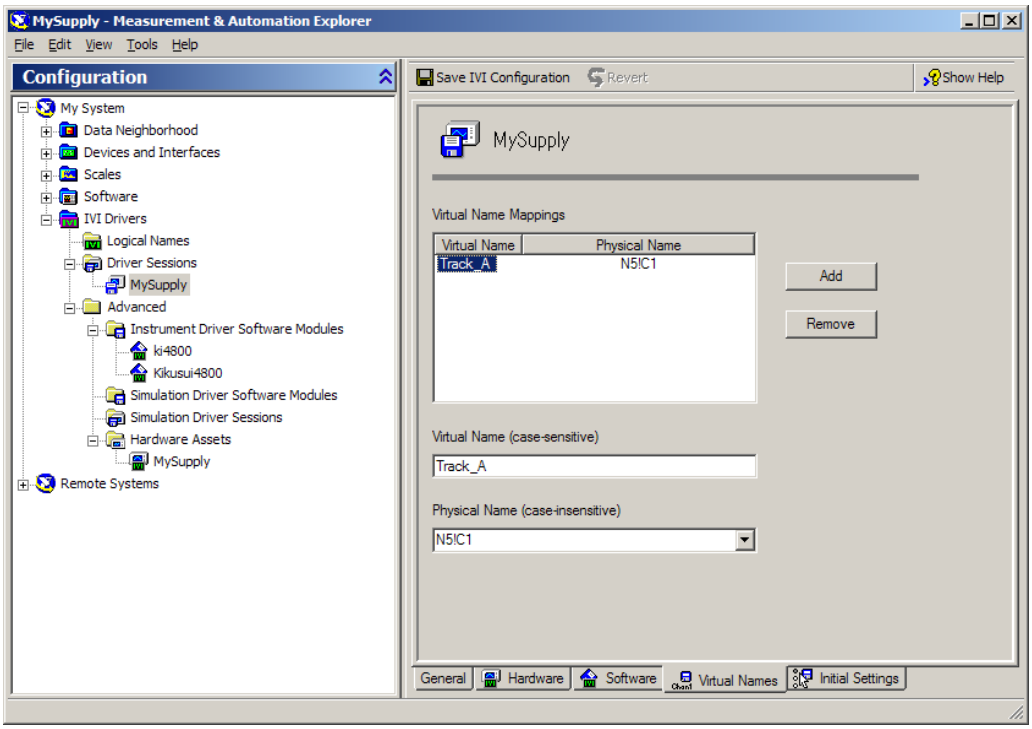


Creating Virtual Name

Subsequently select the **Virtual Names** tab to show the virtual name management screen. Normally, when channel names are related such as for power supply or electronic load drivers, valid channel names are different depending on the drivers. Therefore, these

©2005 KIKUSUI ELECTRONICS CORP. All Rights Reserved.

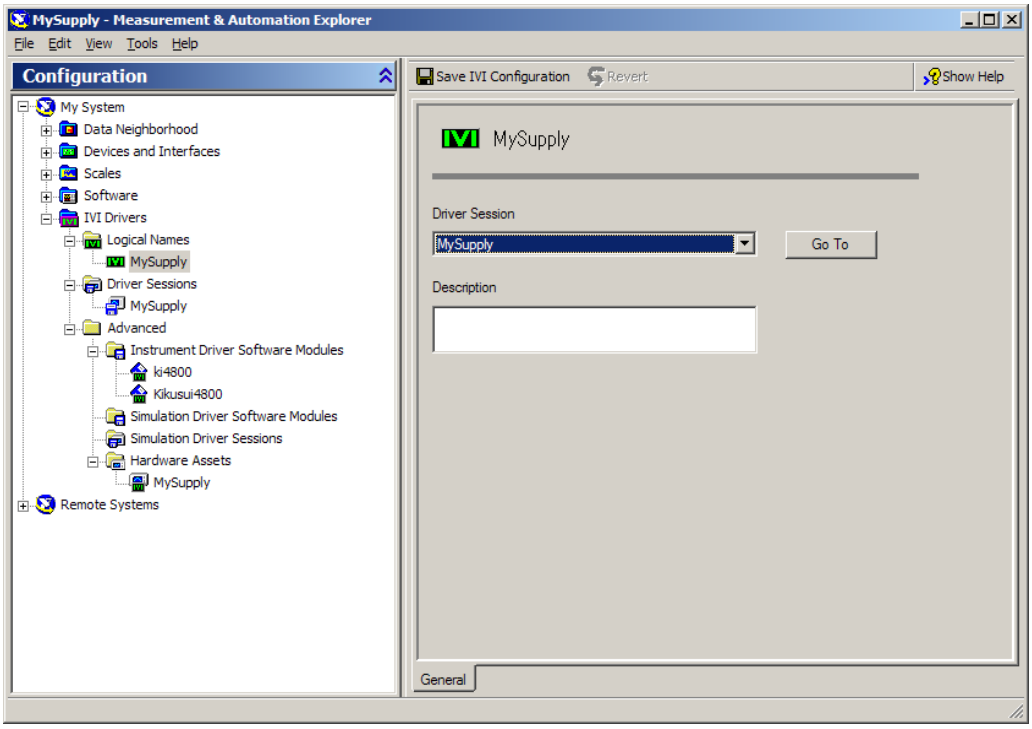
channel names also have to be virtualized. Click the **Add** button to add a virtual name, then type `"Track_A"` for **Virtual Name**. Furthermore, select an appropriate name suitable for the NODE/CHANNEL from **Physical Name**, through which your actual DC power supply is connected. The example shown below selects **N5!C1**, indicating it is connected through NODE 5, CHANNEL 1.



Creating Logical Name

Finally create a logical name. The logical name is equivalent to the name of virtual instrument configured with the NI-MAX. Refer to the **IVI Drivers** node on the tree. Right-click the **Logical Name** then select the **Create New** menu to create the new logical name. Being asked for its name, give the name `"MySupply"`. Furthermore, select `"MySupply"` from the **Driver Session** list.

©2005 KIKUSUI ELECTRONICS CORP. All Rights Reserved.



Configuration for the virtual instrument is complete. Click the **Save IVI Configuration** button placed at the upper screen on the NI-MAX to save changes.

4-2 Adding Controls and Functions

First, create a new application. Open the Front Panel window, place an **error in** cluster and an **error out** cluster.

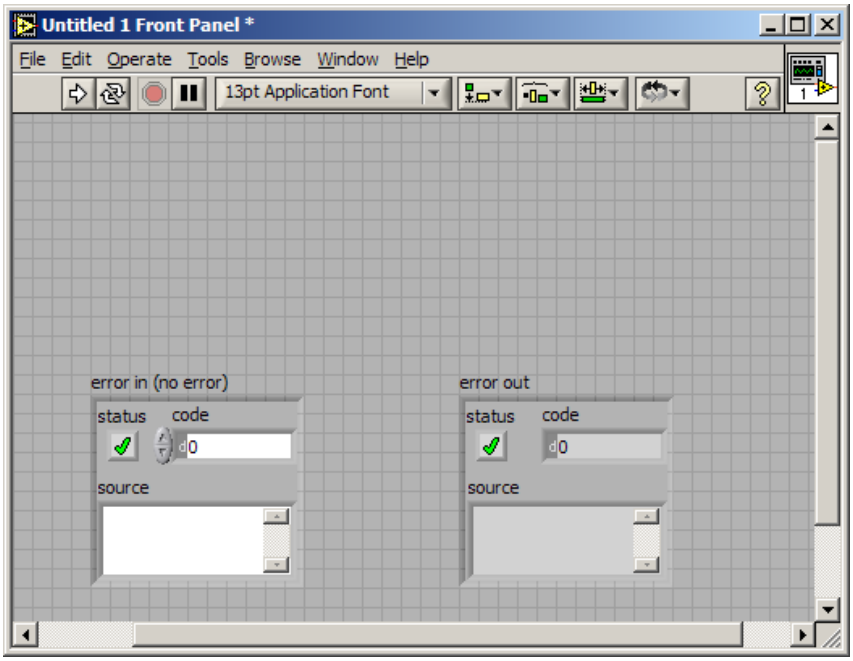


Figure 4-1 Front Panel

Next, open the Block Diagram window, then open the function palette for the IviDCPwr class driver (wrapper). The function palette will be found through the context menu → **Instrument I/O → IVI → IVI DC Power Supply**.

©2005 KIKUSUI ELECTRONICS CORP. All Rights Reserved.

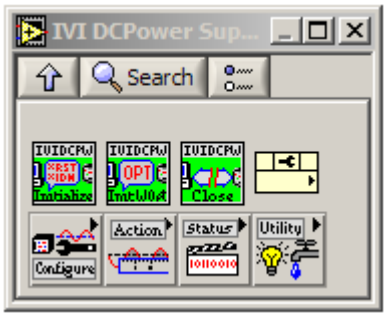


Figure 4-2 IviDCPwr Function Palette

Place `Initialize With Options.vi` and `Close.vi` on the Block Diagram. Furthermore, add `Configure Voltage Level.vi`, `Configure Current Limit.vi`, and `Configure Output Enabled.vi`, which are found in the **Configuration→Output** palette.

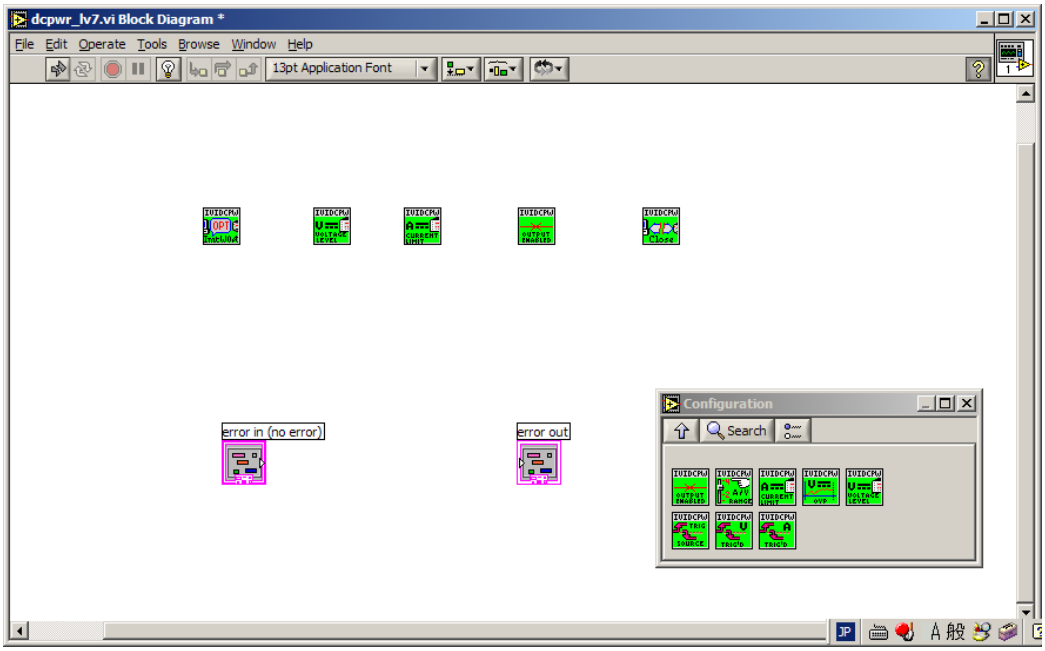


Figure 4-3 Block Diagram

Assuming the PIA4800 series Power Supply Controller is configured as GPIB address 3, pass the parameters -- `resource name`, `id query`, `reset device` to `Initialize With Options.vi`.

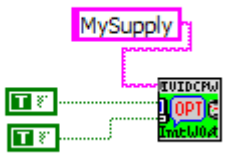


Figure 4-4 Params for Initialize With Options

Subsequently, add parameters that set voltage, current, and output. Here, we set 20V/2A and the output ON.

©2005 KIKUSUI ELECTRONICS CORP. All Rights Reserved.

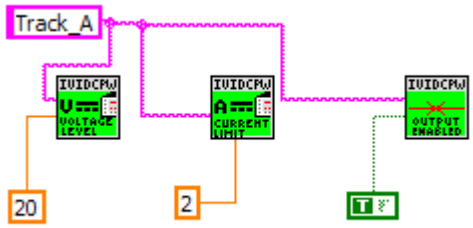


Figure 4-5 Params for Configure Functions

Mind the string -- `"Track_A"`. This is the target channel name for the DC power supply to be controlled. Details are described later.

Finally, wire the controls/functions between **error in** and **error out** clusters as like the picture below. Not only connecting error ins/outs, but make sure to connect instrument session (handle) wires also.

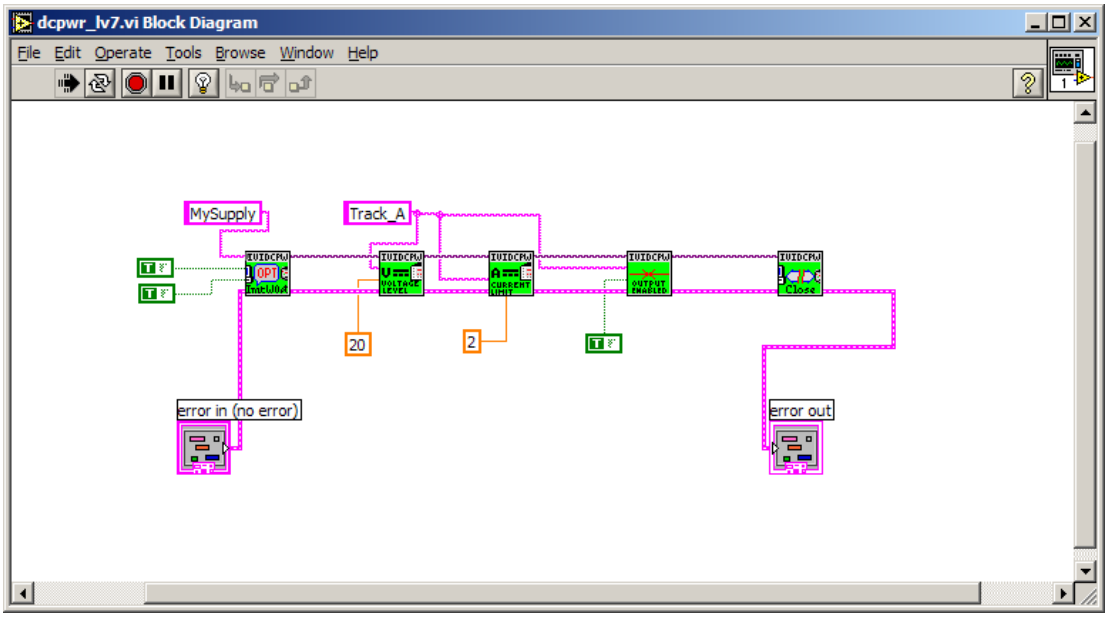


Figure 4-6 Open/Configure/Close

5- Description

5-1 Opening Session

To open the driver session, the `IviDCPwr Initialize With Options.vi` is used. The prefix `IviDCPwr`, which is applied to the vi (function), is specific to the IviDCPwr class driver.

©2005 KIKUSUI ELECTRONICS CORP. All Rights Reserved.

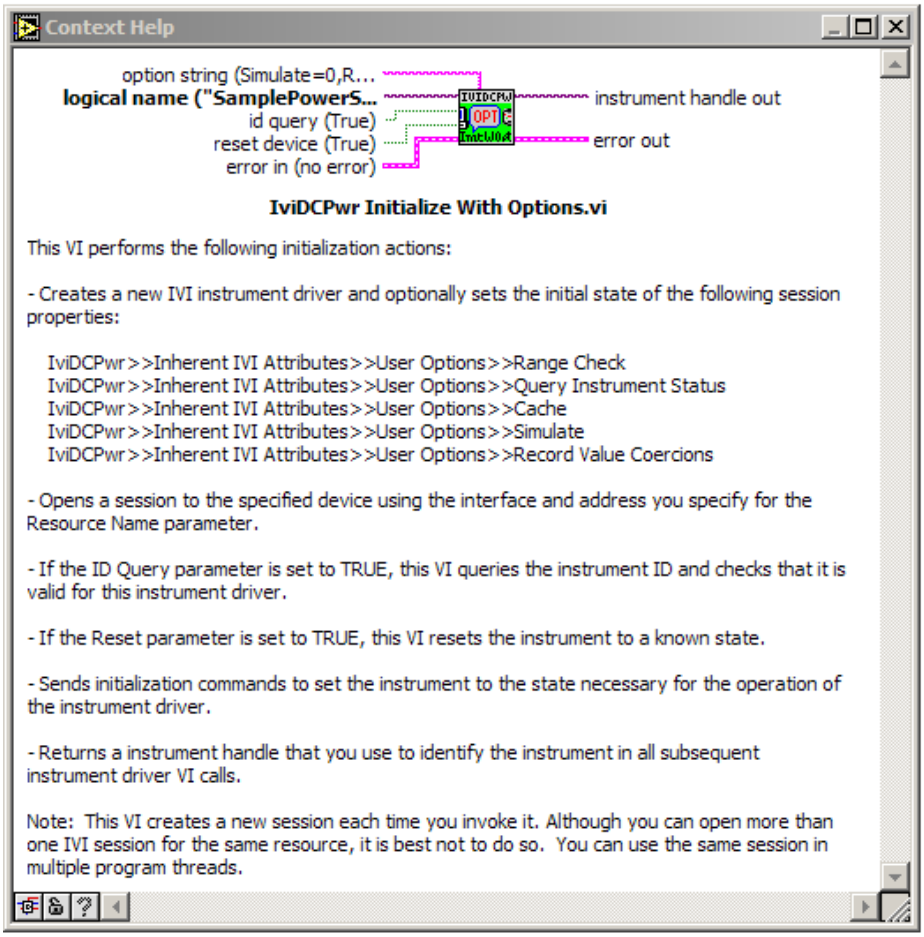


Figure 5-1 Initialize With Options.vi Help

Class drivers are different with normal instrument drivers, thus you cannot pass VISA address to the `Initialize With Options.vi` directly. Instead, pass the logical name `"MySupply"` configured in the NI-MAX. The class driver, by referencing to the logical name, searching for the appropriate instrument driver DLL (Software Module) and VISA address (Hardware Asset), then at last invokes the `ki4800 Initialize With Options.vi` indirectly.

Although the contents for OptionString are exactly the same as when using the specific driver, the default values for the case the parameter was omitted are different. The default values when using a specific driver were the ones that were defined by the IVI specifications, however, the default values when using the a class driver are the ones that are configured at the **Driver Session** in the IVI Configuration Store.

5-2 Channel Access

When supporting power supply and/or electronic load instruments, the IVI instrument driver is generally designed assuming the instrument has multiple channels. Therefore, driver functions operating instrument panel settings often have the **channel name** parameter, which specifies the channel.

©2005 KIKUSUI ELECTRONICS CORP. All Rights Reserved.

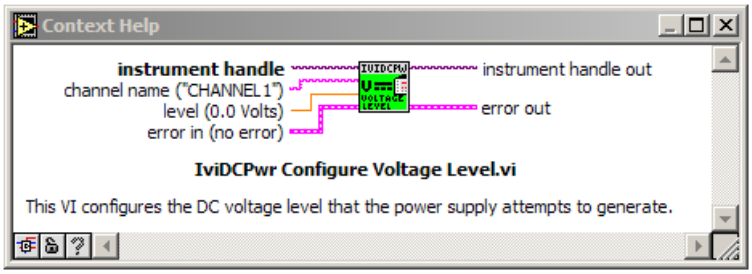


Figure 5-2 Configure Voltage Level.vi Help

When you pass a channel name to the vi, it is possible to specify a name that is usable only with a particular instrument driver (ki4800 driver in this case) such as `"N5!C1"`. However, this approach controlling the instrument with such driver-specific names makes the interchangeability being spoiled.

In above NI-MAX configuration, we added the virtual name `"Track_A"` and configured as it can be converted to the physical name `"N5!C1"`. Therefore we can use the virtual name for the channel name.

5-3 Closing Session

To close the instrument driver session, use the `IviDCPwr Close.vi` function.

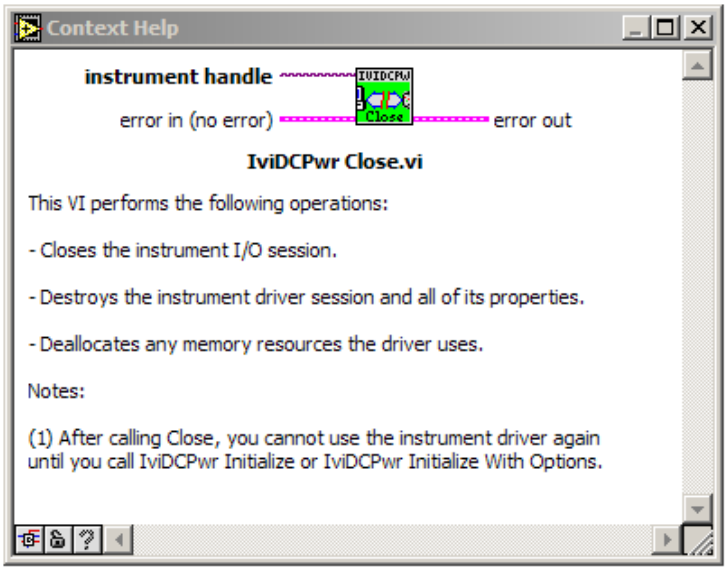


Figure 5-3 Close.vi Help

Product names and company names that appear in this guidebook are trademarks or registered trademarks of their respective companies.
©2005 Kikusui Electronics Corp. All Rights Reserved.