



# IVI Instrument Driver Programming Guide (LabVIEW Edition)

March 2019 Revision 2.2

Product names and company names that appear in this guide are the trademarks or registered trademarks of their respective companies.

© 2019 Kikusui Electronics Corp.

---

# Contents

---

Introduction .....	3
IVI Instrument Driver Available for LabVIEW .....	3
Usable Interfaces.....	4
Programming Using Specific Interfaces .....	5
Preparing to Use Specific Interfaces .....	5
Installing the LabVIEW Instrument Driver Import Wizard.....	5
Installing the IVI Compliance Package .....	5
Importing the IVI-C Driver .....	6
Configuring the Program .....	13
Adding Functions .....	13
Setting Parameters .....	16
Executing the Program .....	18
Function Descriptions .....	19
Starting the Session.....	19
Setting the Channel Name.....	22
Closing the Session .....	23
Programming Using Class Interfaces .....	24
Configuring the Program .....	24
Creating a Virtual Instrument .....	24
Adding Functions .....	30
Setting Parameters .....	33
Function Descriptions .....	35
Starting the Session.....	35
Setting the Channel Name.....	37
Closing the Session .....	38
Switching the Instrument.....	39

## Introduction

---

This guide shows examples of using the KikusuiPwr IVI instrument driver (KIKUSUI PWR-01 Series DC power supply).

You can also use the IVI instrument driver for other manufacturers and models in a similar way.

This guide describes how to use LabVIEW 2018 (64-bit version) to create 64-bit (x64) programs that run on Windows10 (x64).

### IVI Instrument Driver Available for LabVIEW

With LabVIEW, you can import the IVI-C instrument driver to perform programming. The IVI-C instrument driver enables easier programming.

This guide recommends the IVI-C instrument driver and uses it in examples to explain programming procedures.

#### Memo

Our IVI instrument drivers include both IVI-C and IVI-COM drivers. You can use the IVI-COM instrument driver directly as a general COM component.

## Usable Interfaces

The IVI instrument driver supports the following two types of interfaces:

- **Specific interfaces**

Interfaces that are specific to their instrument drivers. You can fully utilize the instrument's features.

- **Class interfaces**

Interfaces for the instrument classes defined in the IVI specifications.

You can use the interchangeability feature, but the use of model-specific features is restricted.

This guide describes how to perform programming using each interface.

### Memo

- The instrument class to which an instrument driver belongs is documented in Readme.txt for each driver.  
The Readme document can be viewed from [Start] button > [Kikusui] > [KikusuiPwr IVI Driver 1.0.0 Documentation] menu.
- If the instrument driver does not belong to any instrument classes, you cannot use class interfaces and therefore cannot create applications that use the interchangeability feature.

# Programming Using Specific Interfaces

---

By using a specific interface, you can fully utilize the model-specific features of the instrument driver.

This section describes how to program using specific interfaces.

## Memo

When using specific interfaces, you cannot use the interchangeability feature. To use the interchangeability feature, use class interfaces. (go to page 24)

## Preparing to Use Specific Interfaces

### Installing the LabVIEW Instrument Driver Import Wizard

---

To write codes that call the IVI-C instrument driver in LabVIEW, use the LabVIEW Instrument Driver Import Wizard to import the IVI-C driver, and create the LabVIEW VI library (LabVIEW IVI-C wrapper).

Download and install the LabVIEW Instrument Driver Import Wizard from the National Instruments website in advance.

### Installing the IVI Compliance Package

---

To implement the LabVIEW IVI-C wrapper created by the Import Wizard, the IVI Compliance Package is required for the runtime environment.

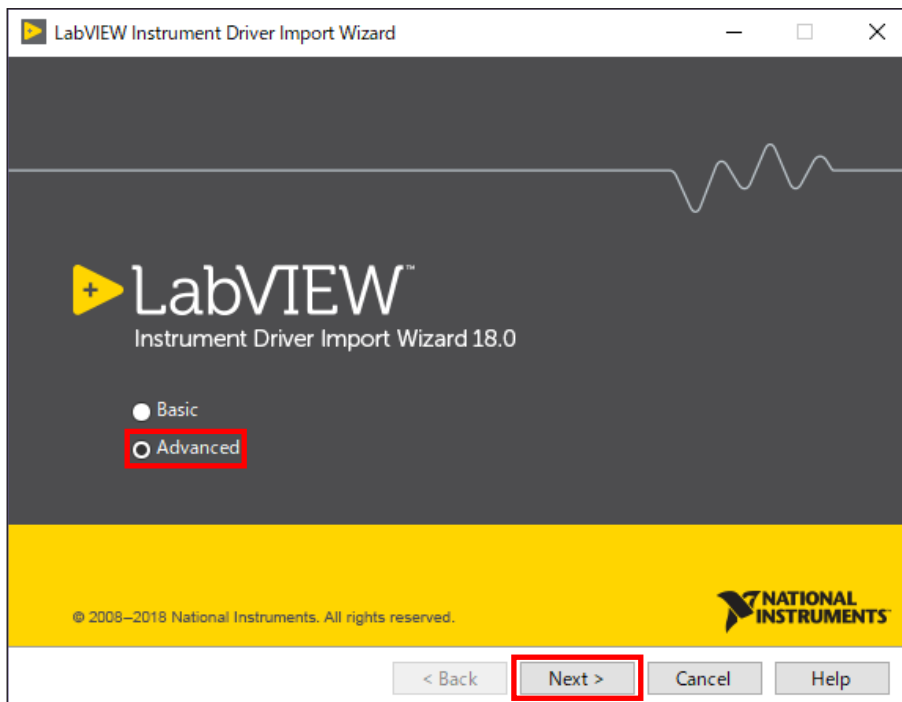
Download and install the IVI Compliance Package from the National Instruments website in advance.

## Importing the IVI-C Driver

To program using specific interfaces, the IVI-C driver must be converted to a LabVIEW-compatible format and imported.

To import the IVI-C driver, use the LabVIEW Instrument Driver Import Wizard (go to page 5).

- 1** Once the Instrument Driver Import Wizard is installed, launch LabVIEW.
- 2** Select [Tools] > [Instrumentation] > [Import LabWindows/CVI Instrument Driver] from the menu.
- 3** Select [Advanced] and click [Next >].

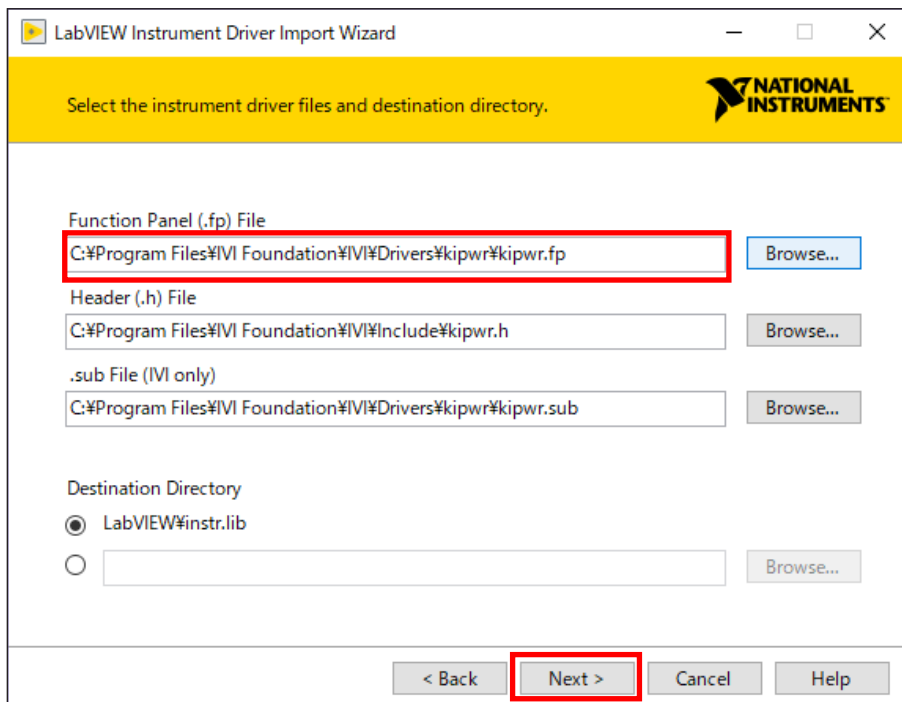


## 4 Specify the .fp (function panel) file of the IVI instrument driver under [Function Panel (.fp) File] and click [Next >].

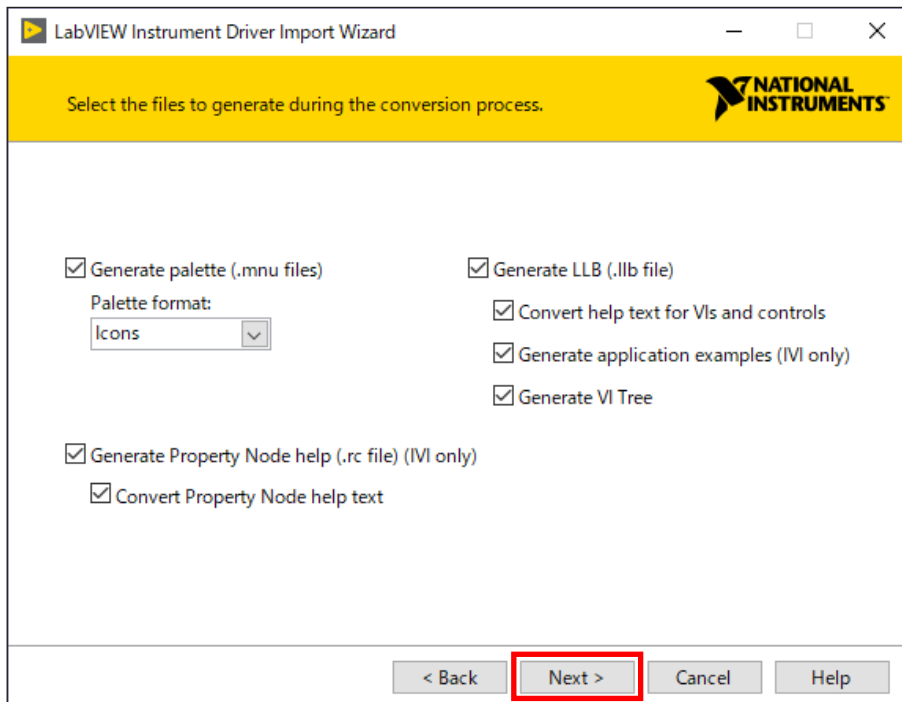
In this example, kipwr.fp is specified for using the KikusuiPwr IVI instrument driver. kipwr.fp is stored in the following directory:

C:/Program Files/IVI Foundation/IVI/Drivers/kipwr

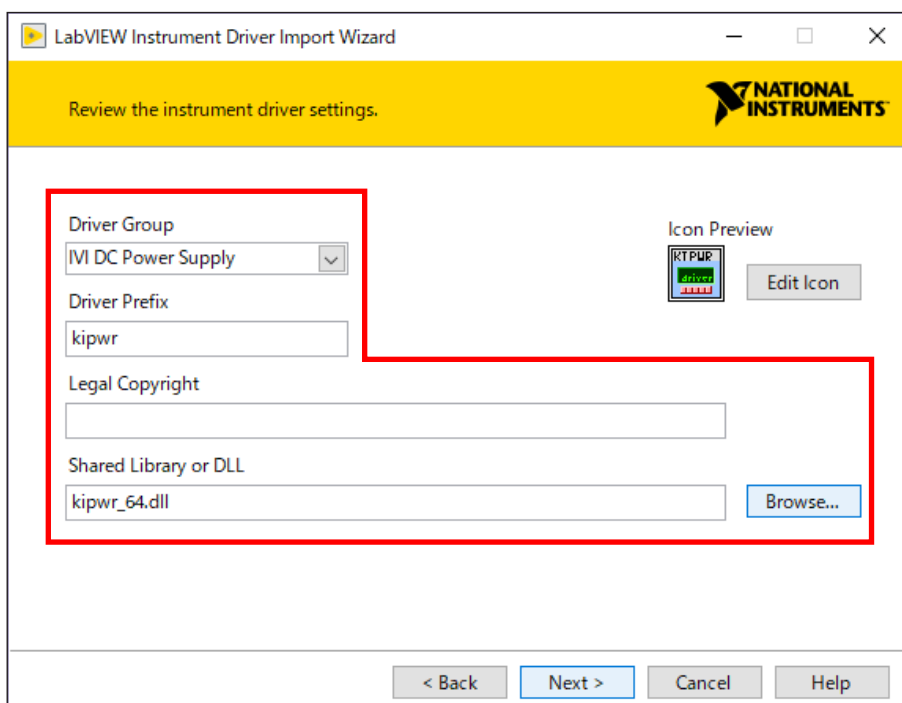
The .h file (C-language header file) and the .sub file (attribute information file) are automatically specified.



**5** Make sure that all items for the file types to be generated are selected, and click [Next >].



**6** Set each item.





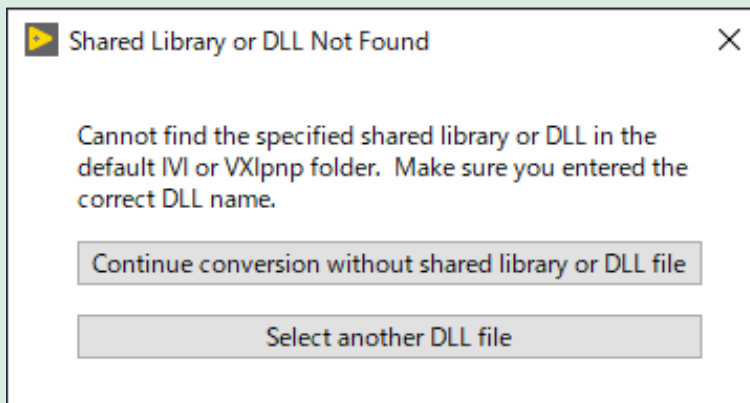
Setting Item	Description
[Driver Group]	<p>Specifies the instrument class that the IVI instrument driver belongs to. In this example, the IviDCPwr class (IVI DC power supply) is specified.</p> <p>If the IVI instrument driver to be imported does not belong to any class, select [IVI Generic].</p>
[Driver Prefix]	<p>Confirms that the .fp file selected in Step 4 is specified.</p>
[Legal Copyright]	<p>Sets the copyright notice to be included in the instrument driver.</p> <p>In this example, the field is left blank.</p>
[Shared Library or DLL]	<p>Specifies the DLL file for the driver.</p> <p>In this example, [kipwr.dll] stored in the following directory is specified.</p> <p>C:/ProgramFiles/IVI Foundation/IVI/BIN</p> <p>A file name using wildcards is entered by default. Specify the correct DLL file.</p>

## 7 Click [Next >].

The function tree screen appears.

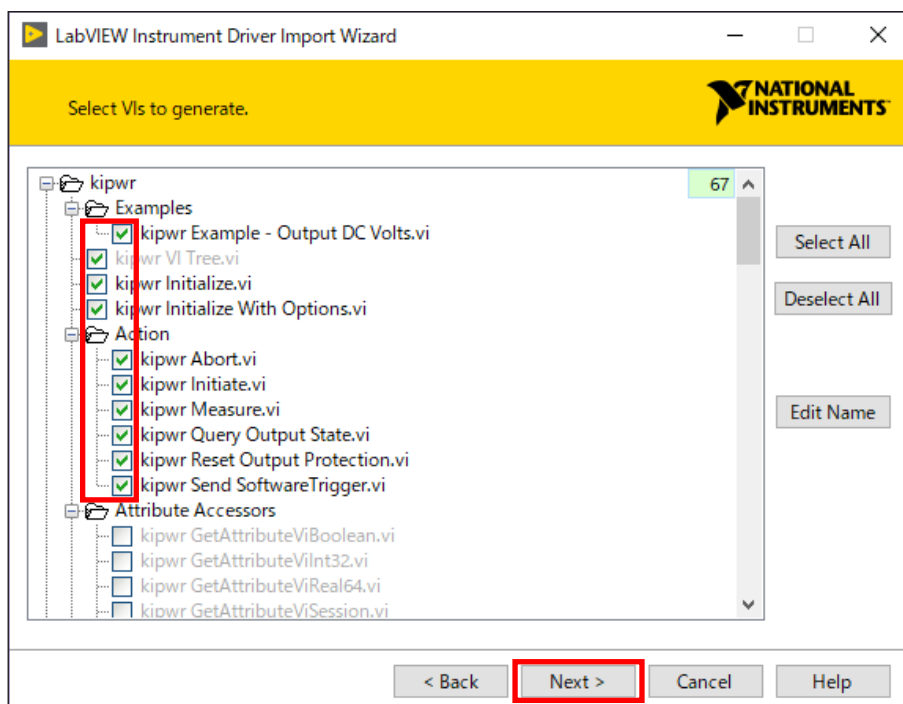
### Memo

- If [Shared Library or DLL] is specified incorrectly, the following dialog box is displayed. Click [x] to close the dialog box, and specify another DLL file or check the installation status of the IVI instrument driver again.



## 8 Select the VIs to be generated and click [Next >].

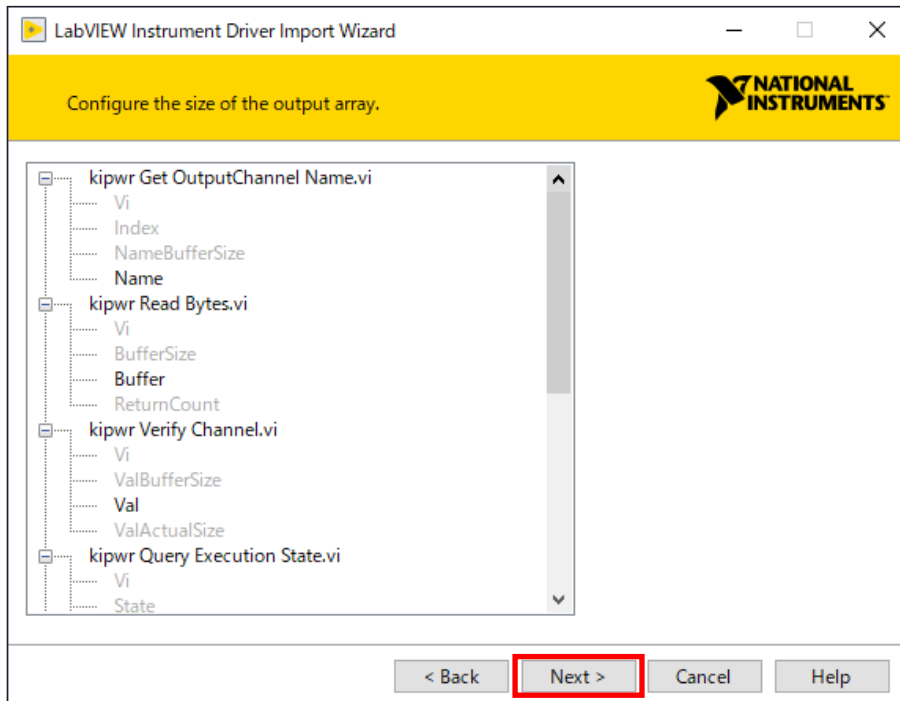
Normally, you do not need to change these settings.



## 9

**Click [Next >].**

The screen for changing the memory size used for the VI is displayed, but normally, you do not need to change these settings.

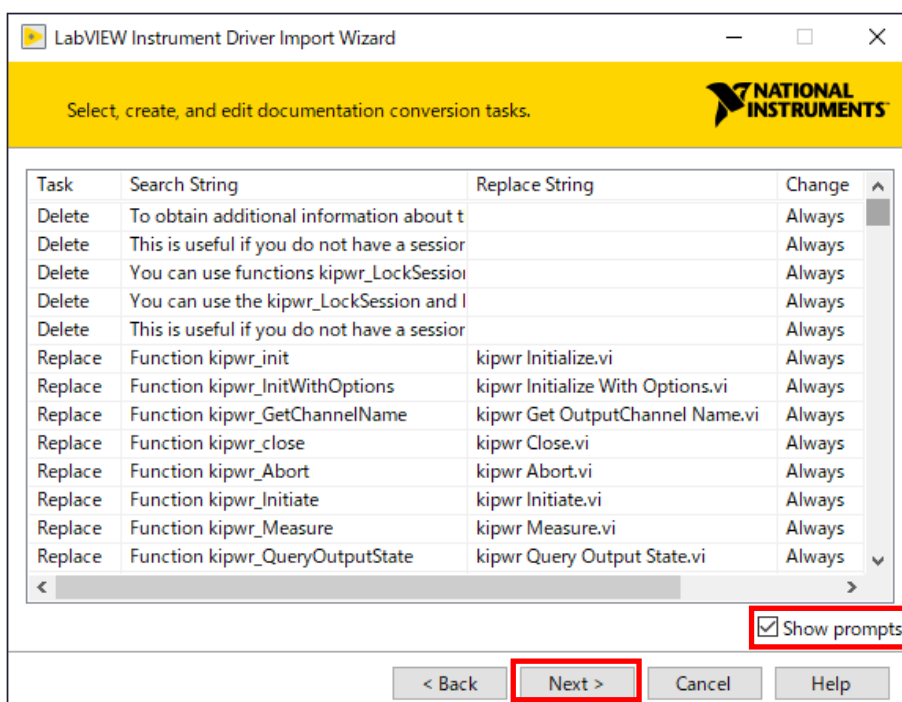


## 10

**Specify the conversion tasks for documents (mainly context and help) and click [Next >].**

Most documentation conversions replace “Function” with “VI.”

This guide recommends that you uncheck [Show prompts] to avoid displaying a confirmation prompt for every conversion, as these have low importance.



An outline of the conversion tasks appears.

# 11

## Click [Next >].

Conversion starts. When the conversion is complete, the IVI instrument driver sub-directory (in this example, the kipwr subdirectory) is created in the LabVIEW default instrument driver directory, and an IVI-C wrapper that can be used directly from LabVIEW is created.

The kipwr IVI-C wrapper can be referenced from the Instrument I/O function palette on the LabVIEW block diagram.

### Memo

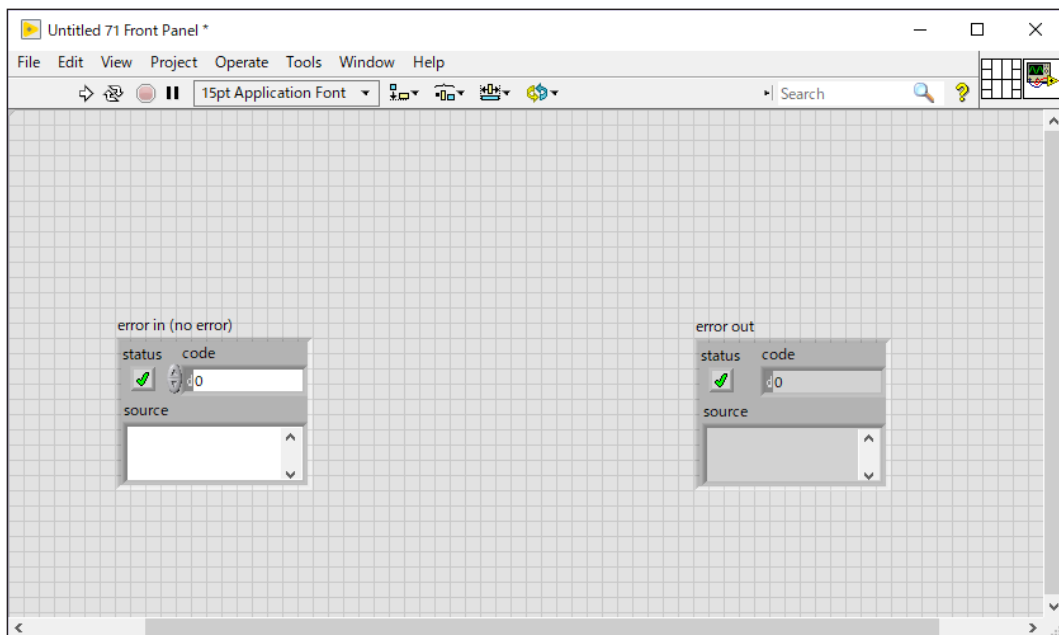
- In general, the default instrument driver directory for LabVIEW is the following:  
C:/Program Files/National Instruments/LabVIEW 2018/instr.lib
- The IVI-C wrapper is created as a file set that includes a VI library file (.llb) and multiple palette menu files (.mnu).
- The created .llb file is not a real instance of the instrument driver, but is instead a wrapper module to the IVI-C driver.  
Therefore, the IVI instrument driver must be installed on the target machine when you run the completed application.
- When installing the IVI instrument driver on the target machine, use the driver's installer, the same as installing to the development machine.  
Just copying DLL or other pieces of data does not work correctly.
- On the target machine, the IVI Compliance Package must be installed in addition to the LabVIEW Runtime Engine.

## Configuring the Program

### Adding Functions

This section describes how to add functions using specific interfaces. This example adds functions for voltage, current, and output.

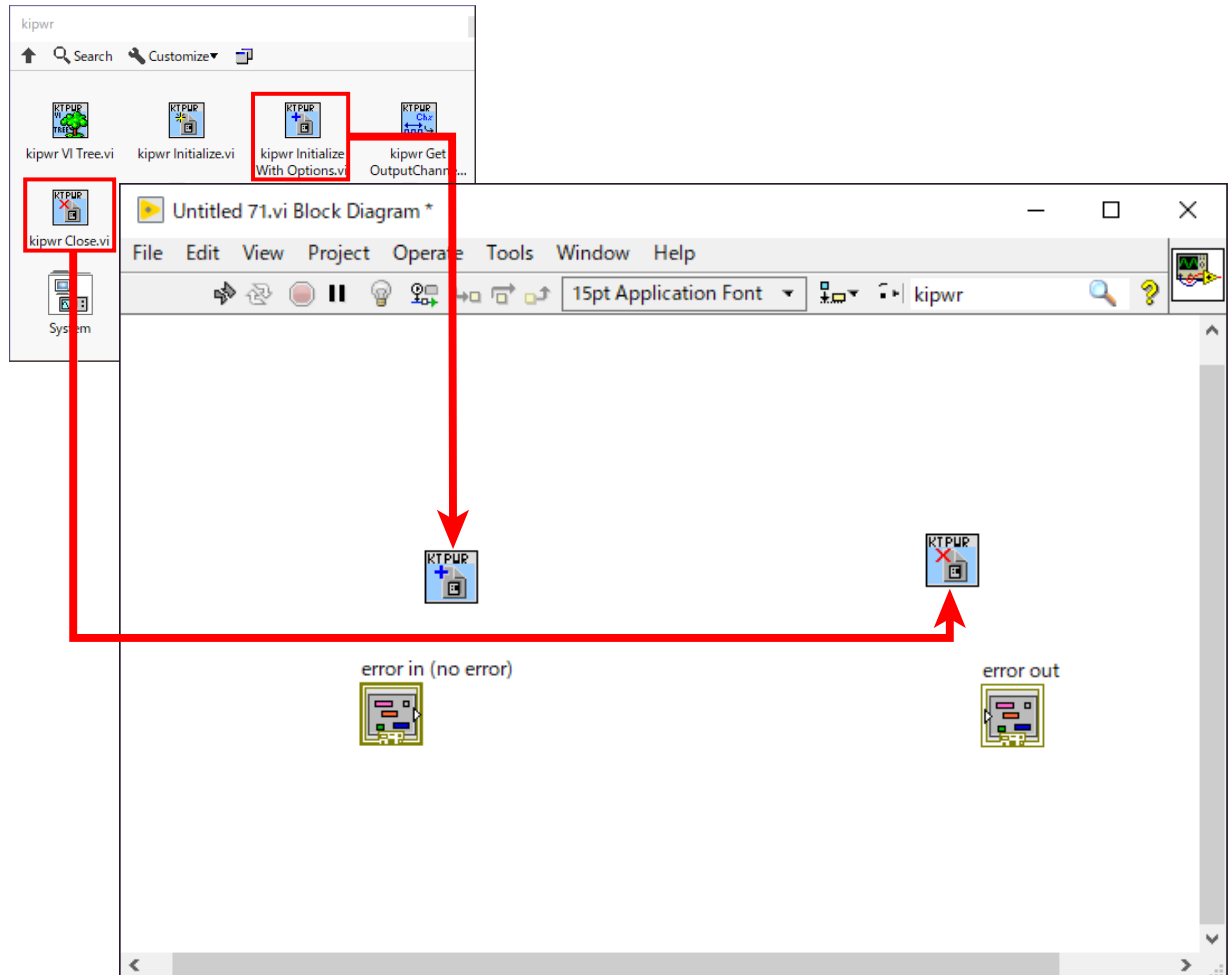
- 1 **Display the Front Panel screen, and place the [error in] cluster and the [error out] cluster.**



- 2 **Display the Block Diagram screen and right-click. Select [Instrument I/O] > [Instr Drivers] > [kipwr] to open the kipwr function palette.**

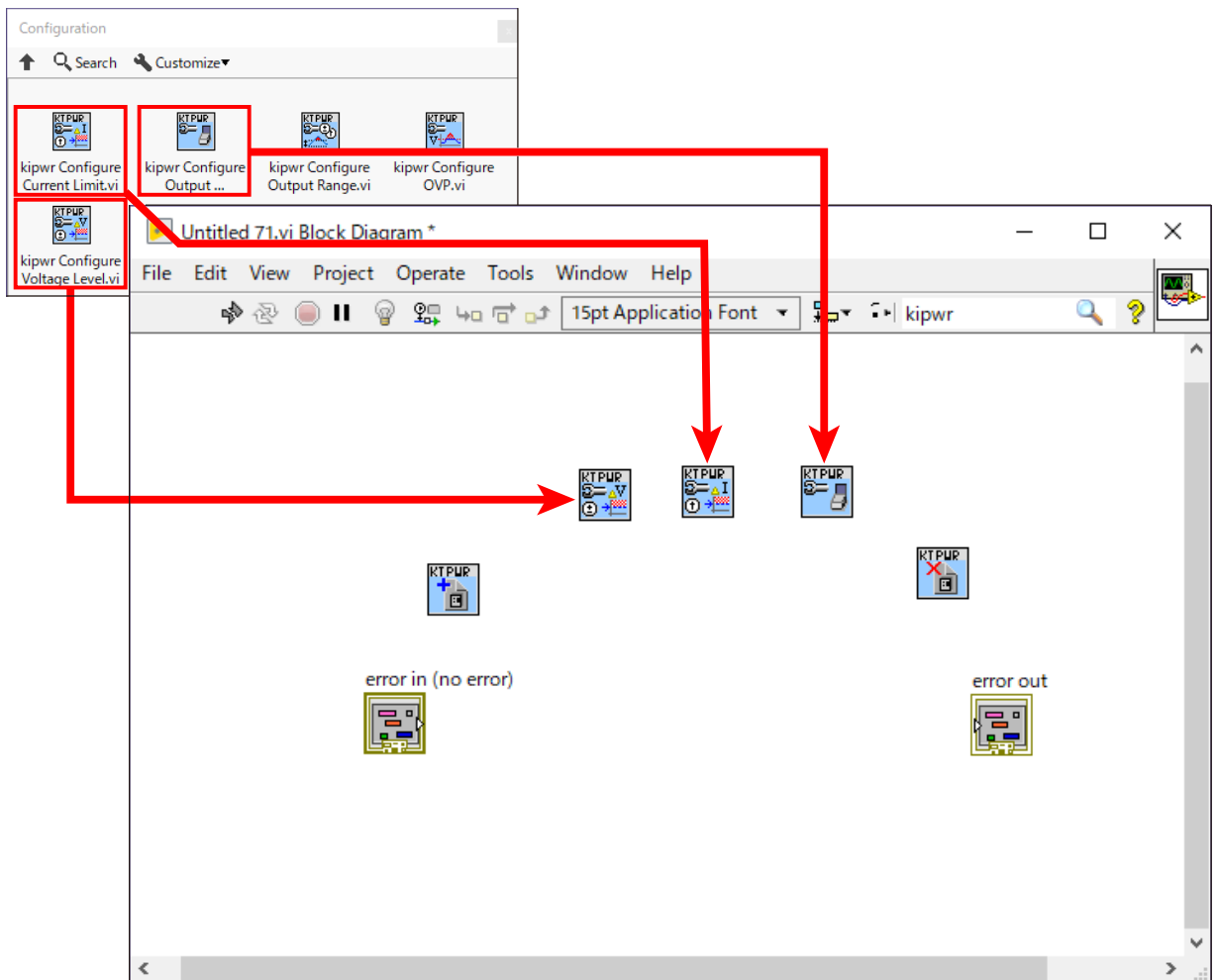
### 3 Add the following VIs to the Block Diagram screen:

- Kipwr Initialize With Options.vi
- Kipwr Close.vi



## 4 Open the [Configuration] palette and add the following VIs.

Parameter	VI
Parameter to set voltage	Configure Voltage Level.vi
Parameter to set current	Configure Current Limit.vi
Parameter to set output	Configure Output Enabled.vi



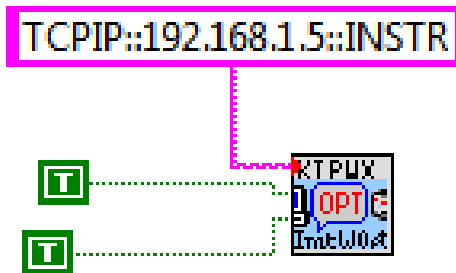
Then, set parameters to the added functions.

## Setting Parameters

This section describes how to set parameters to the added functions to configure the program. It shows an example of setting the voltage to 20 V and the current to 2 A to turn on the output.

The following explanation describes an example of setting parameters for the Kikusui PWR-01 Series DC power supply on a network that is set to the IP address 192.168.1.5.

- 1 Bridge the parameters of [resource name], [ID Query], and [Reset Device] to [Initialize With Options.vi].**



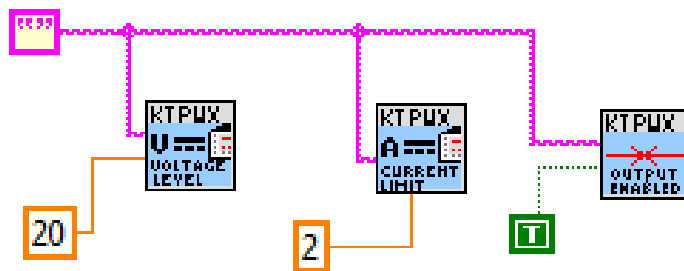
- 2 Add parameters that set voltage, current, and output to each VI.**

- 3 Bridge the character string of the DC power supply channel name to be controlled to the same three VIs as in Step 2 on a shared line.**

In this example, a blank string is set as a channel name.

A blank string can be used when there is only one channel.

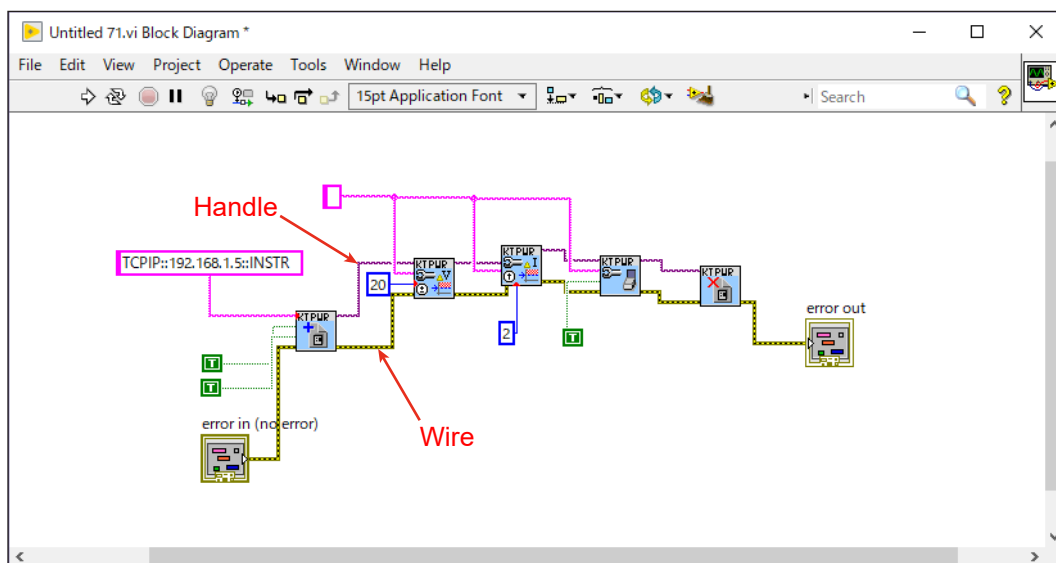
For the details on the channel names that can actually be used, refer to the driver's online help or other resources.





# 4

Connect the [error in] cluster to the [error out] cluster by wire, and connect the instrument session (handle).



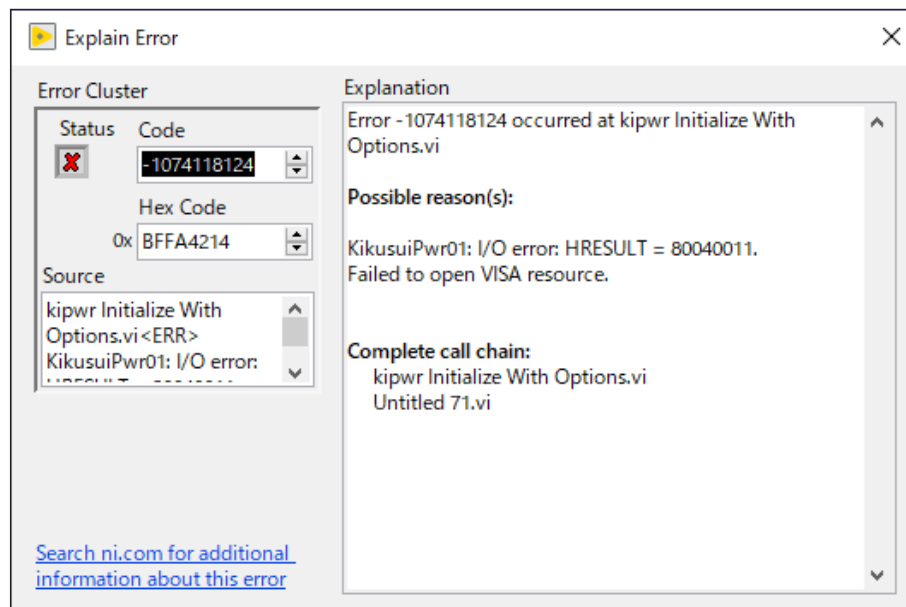
## Executing the Program

You can execute the program even with the configuration that has been set to this point.

The [Reset Device] parameter in Initialize With Options.vi is specified as “TRUE” by default. Therefore, when the program is executed, the instrument is reset and then communication starts.

For the details on functions and parameters, refer to “Function Descriptions” (go to page 19).

- If the error code of the [error out] cluster displays “0” when the measurement instrument is actually connected, then [Initialize With Options] and [Close] have been successful.
- If communication failed, the VISA library was not set correctly, or another problem transpired, an exception occurs and is displayed in the [error out] cluster. Right-click the [error out] cluster and select [Explain Error] to check the details.



## Function Descriptions

This section describes the details and settings of the functions (VIs) that configure the program.

### Memo

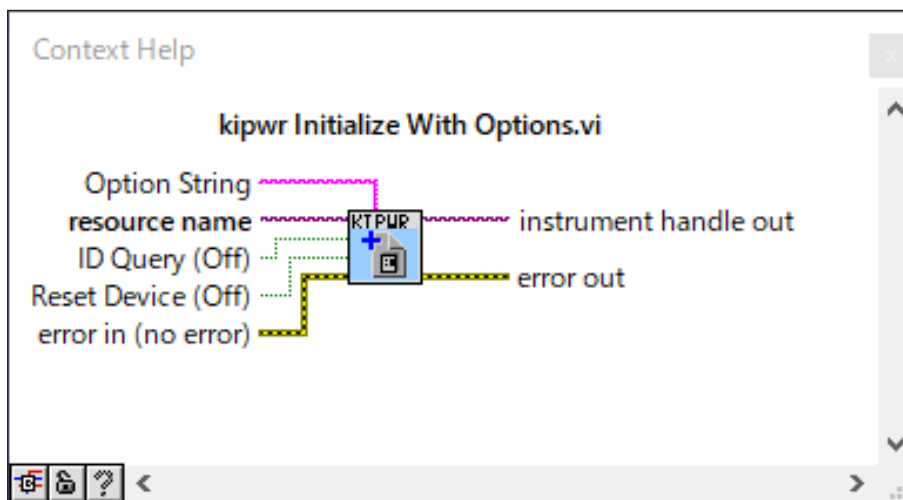
The term <prefix> is frequently used as a convention of function names in IVI-C and VXI Plug&Play instrument drivers. It indicates the identifier name given to each instrument driver.

For example, the expression “<prefix> Initialize.vi” becomes “kipwr Initialize.vi” for the kipwr instrument driver.

## Starting the Session

To start the session, use Initialize With Options.vi.

Initialize With Options.vi, which is defined in the IVI specifications, is included in all IVI instrument drivers.



The following parameters can be set for kipwr Initialize With Options.vi.

Parameter	Type	Description
resource name	String	VISA resource name string decided according to the I/O interface and address through which the instrument is connected. For example, if an instrument with the IP address 192.168.1.5 is connected through VXI-11, the resource name will be "TCPIP::192.168.1.5::INSTR".
ID Query	Boolean	Specifying VI_TRUE issues an ID query such as "*IDN?" to the instrument to inquire on the model information.
Reset Device	Boolean	Specifying VI_TRUE issues a command, such as the "*RST" command, to reset the instrument settings.
Option String	String	RangeCheck Cache Simulate QueryInstrStatus RecordCoercions InterchangeCheck Changes the abovementioned settings defined in the IVI specifications. If the [DriverSetup] parameter is supported by the instrument driver being used, this parameter can also be set.

## For Setting the Option String

Set the Option String as explained below:

- Option String is a string parameter. When setting multiple items, use commas to separate them.  
Set the parameter in the same format as the following sample string:  
QueryInstrStatus=TRUE, Cache=TRUE, DriverSetup=12345
- Function names and setting values are not case-sensitive. Uppercase and lowercase characters are not distinguished.
- The setting values are ViBoolean type. Any from “VI\_TRUE,” “VI\_FALSE,” “1,” or “0” can be specified.
- If an item is not explicitly specified, the following default values defined in the IVI specifications are applied:
  - [RangeCheck] and [Cache]: VI\_TRUE
  - Other than [RangeCheck] and [Cache]: VI\_FALSE
- Some instrument drivers do not support the [DriverSetup] parameter.  
The [DriverSetup] parameter is specified to call an item that is not defined in the IVI specifications when calling Initialize With Options.vi. Its purpose and format are driver-specific. Therefore, [DriverSetup] must be specified as the last item in [Option String]. For details on the [DriverSetup] parameter, refer to the driver’s Readme document, online help, or other resources.

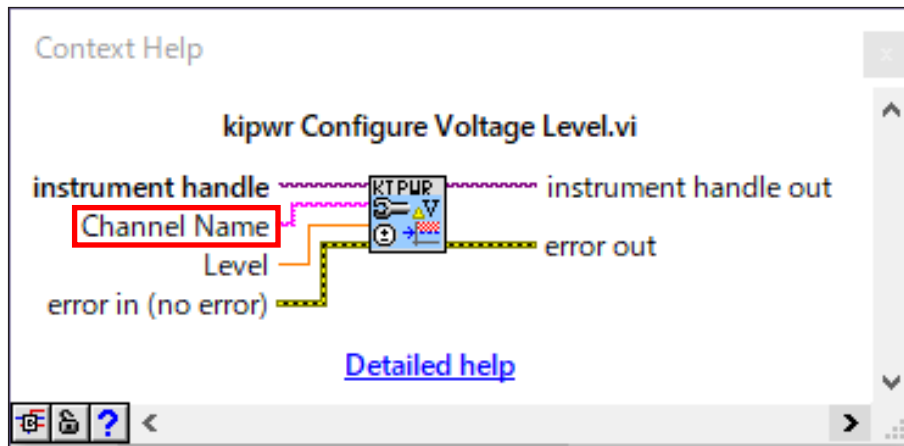
### Memo

- Every VI (driver function) except for “<prefix> Initialize.vi” and “<prefix> Initialize With Options.vi” has the [instrument handle (in)] parameter at the upper-left corner of the VI.
- Every VI (driver function) except for “<prefix> Close.vi” has the [instrument handle out] parameter at the upper-right corner of the VI.  
This output parameter connects to [instrument handle (in)] of the next VI.
- “<prefix> Initialize.vi” is left for compatibility with the VXI Plug&Play driver specifications. It is equivalent to “<prefix> Initialize With Options.vi,” with the exception that the [Option string] parameter cannot be specified.

## Setting the Channel Name

The IVI instrument driver was designed on the premise that, when supporting power supplies, oscilloscopes, and so on, the instrument has multiple channels.

Therefore, many driver functions that operate instrument panel settings may require you to specify the channel for the [Channel Name] parameter.



For example, the KikusuiPwr (kipwr) driver used as an example in this guide is a DC power supply driver.

Specify a channel name to be controlled.

The channel names that can actually be used differs depending on the driver. For details on this setting, refer to the driver's Readme document, online help, or other resources.

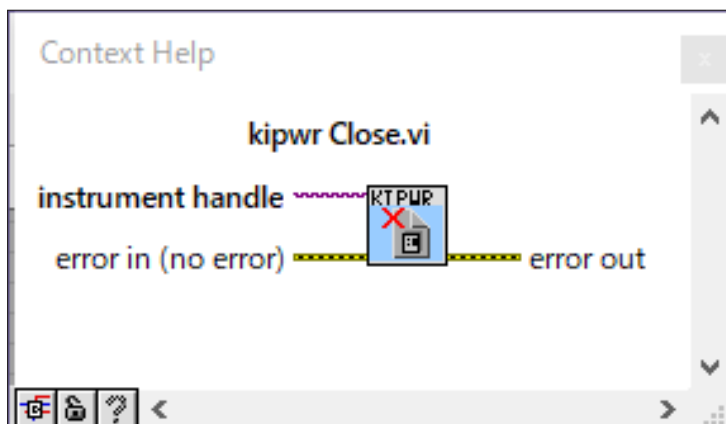
### Memo

- If there is only one channel, operation is possible even when the specified channel name is a blank string ("").
- If multiple channels exist, you must specify the channel names to be used. In general, "Output1" is specified for a power supply.
- If you operate PWR-01 Series with the multidrop extension, channel numbers are assigned sequentially from zero. Therefore, the first channel name will be "Output0."

## Closing the Session

---

To close the instrument driver's session, use Close.vi.



## Programming Using Class Interfaces

---

Programming using an IVI-defined instrument class interface enables interchangeability using an instrument class driver.

Interchangeability allows you to switch instruments without having to recompile and link the application.

### Memo

- To use interchangeability, the IVI-C instrument drivers must be provided for the models both before and after switching, and these drivers must belong to the same instrument class. Interchangeability is not possible between different instrument classes.
- For programming that uses class interfaces, the model-specific features that can be used are restricted. To fully utilize model-specific features, perform programming with specific interfaces. (go to page 5)

## Configuring the Program

### Creating a Virtual Instrument

---

To create an application that uses the interchangeability feature, you must create a virtual instrument in advance.

### Memo

Do not use descriptions that are dependent on a specific IVI-C instrument driver (e.g. a direct call of the `kipwr_init` function) or descriptions of a specific VISA address (resource name, e.g. "TCPIP::192.168.1.5::INSTR") in the application code, because doing so will spoil the interchangeability feature.

The IVI specifications provide an interchangeability feature by placing the IVI Configuration Store outside of the instrument driver and application.

Applications do not directly use model-specific instrument drivers, but instead they control through special instrument drivers called instrument class drivers.



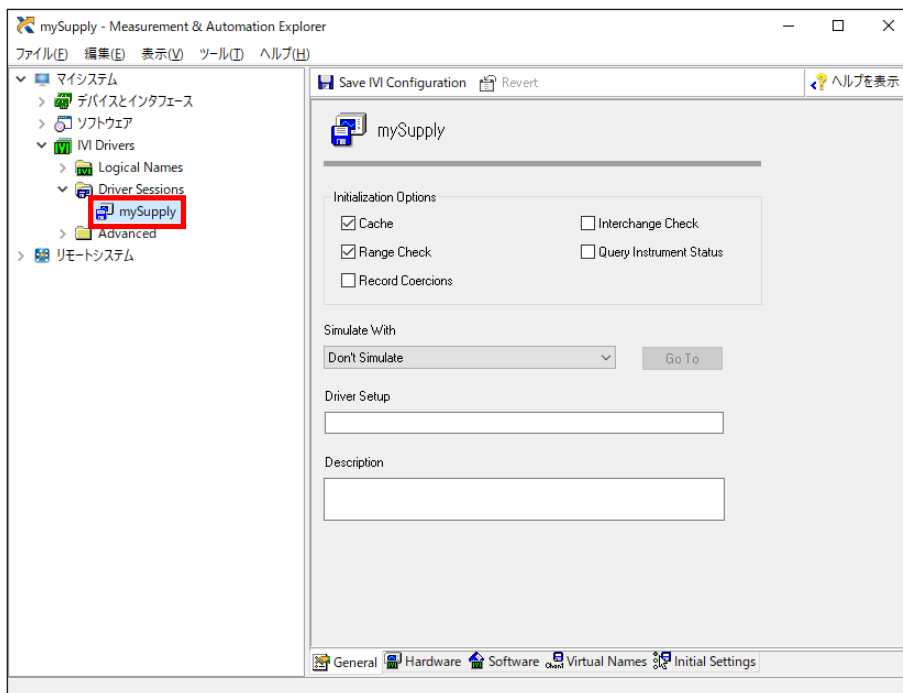
For control, select the instrument driver's DLL according to the contents of the IVI Configuration Store, and use the class driver function, which is not model-affiliated, to access the instrument driver that was loaded indirectly.

An XML file (C:/ProgramData/IVI Foundation/IVI/IviConfigurationStore.xml) is used for the IVI Configuration Store. IVI instrument drivers and some VISA/IVI configuration tools access the store through the IVI Configuration Server DLL. Applications do not normally use it. When using LabWindows/CVI, use the NI-MAX (NI Measurement and Automation Explorer) software provided by National Instruments to perform IVI driver configuration.

This section describes how to create virtual instruments using NI-MAX.

- 1 **First, create a Driver Session. Launch NI-MAX and expand the [IVI Drivers] node on the tree.**
- 2 **Right-click [Driver Session] and select [Create New (case-sensitive)].**
- 3 **Specify a name for the Driver Session.**

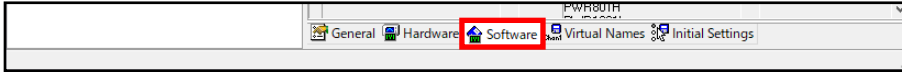
Here, it is [mySupply].





8

Next, set the **Software Module** link. Select the **[Software]** tab.



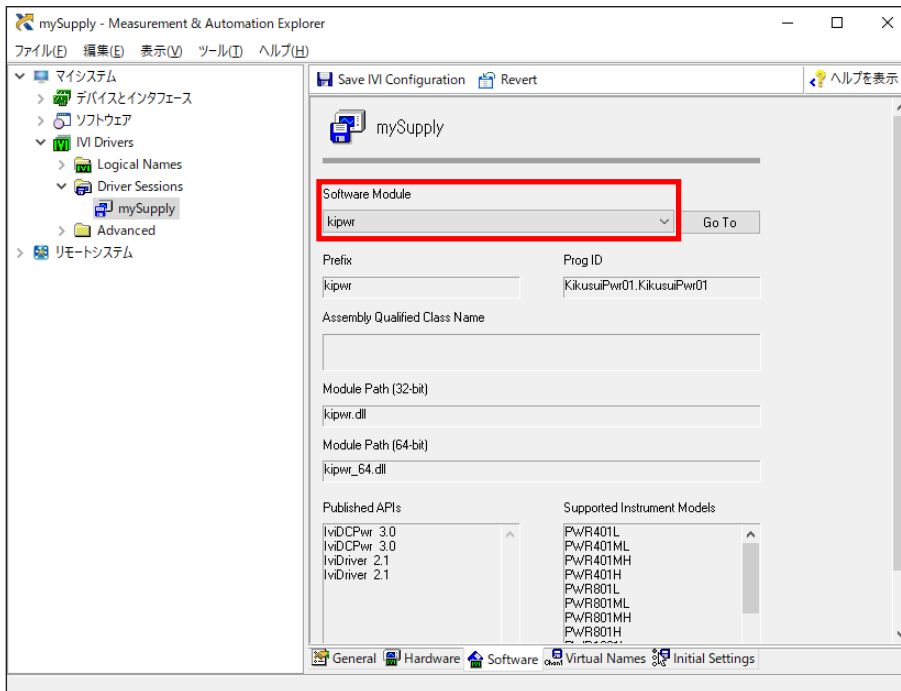
### Memo

Software Module shows the instrument driver module (DLL module).

9

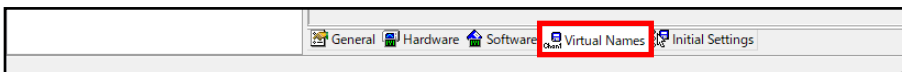
Select the desired instrument driver module from the **[Software Module]** list.

In this example, [kipwr] is selected.



10

Next, create a Virtual Name. Select the **[Virtual Names]** tab.



### Memo

For instrument drivers that require the specification of channels, valid channel names differ depending on the instrument driver.

Therefore, set the virtualization of these channel names in the Virtual Name tab.



- 13** Finally, create a Logical Name to set a link. Expand the [IVI Drivers] node in the tree on the left side of the screen.

### Memo

The Logical Name is equivalent to the name of the virtual instrument configured with NI-MAX.

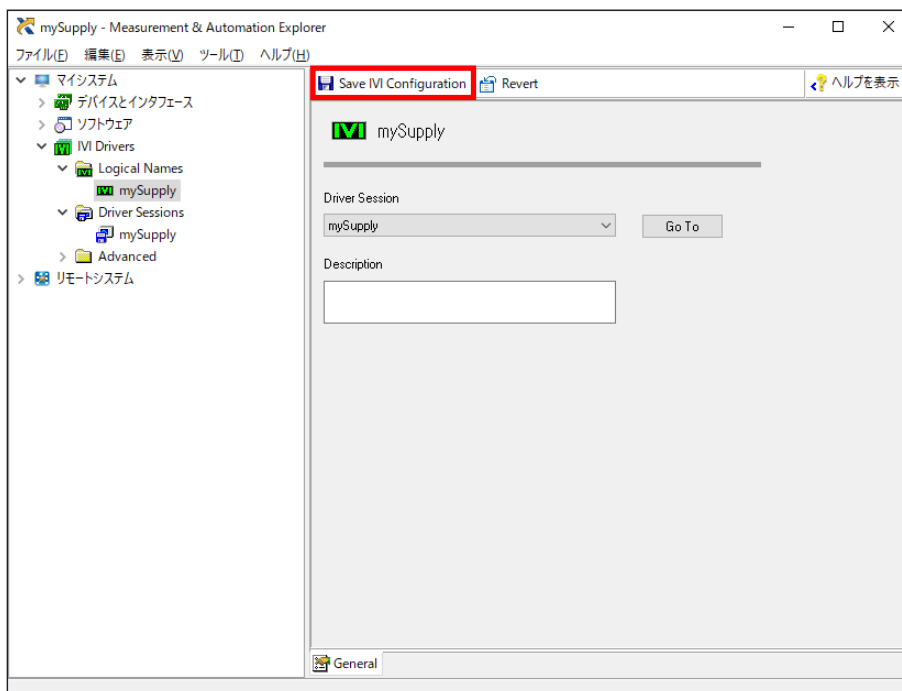
- 14** Right-click [Logical Name] and select [Create New (case-sensitive)].

- 15** Specify a name for Logical Name.

Here, it is [mySupply].

- 16** Select [mySupply] for [Driver Session].

- 17** Click [Save IVI Configuration] on the tool bar to save the setting.

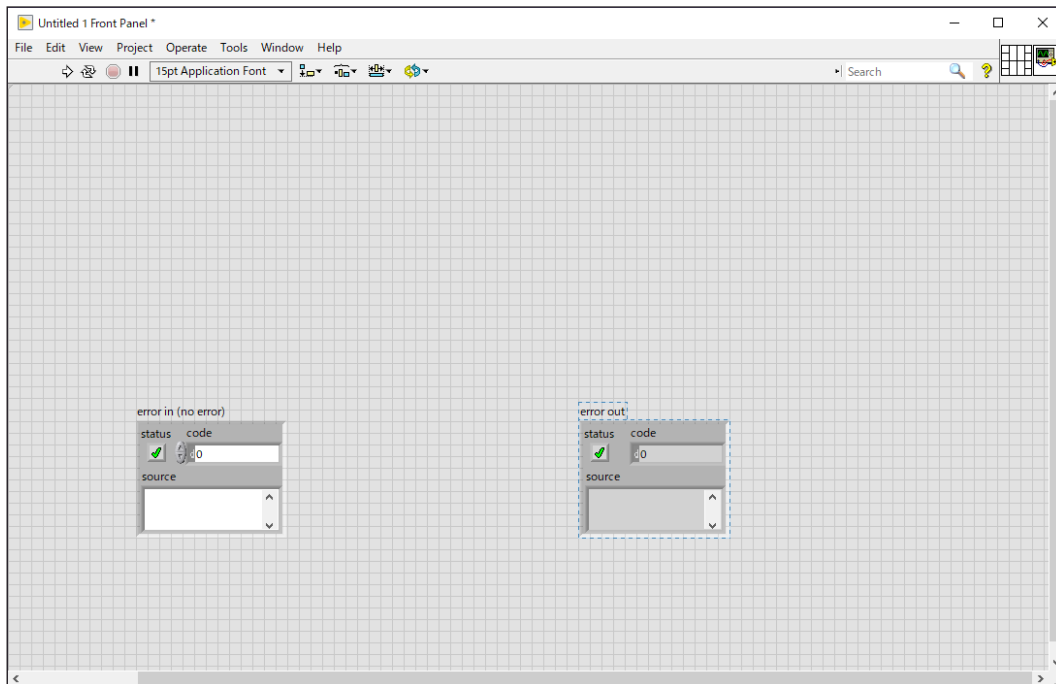


The creation of a virtual instrument is complete.

## Adding Functions

This section describes how to add functions using class interfaces. This example adds functions for voltage, current, and output.

- 1 **Display the Front Panel screen, and place the [error in] cluster and the [error out] cluster.**



- 2 **Display the Block Diagram screen and right-click. Select [Instrument I/O] > [IVI Class Drivers] > [DC Power Supply] to open the IviDCPwr function palette.**

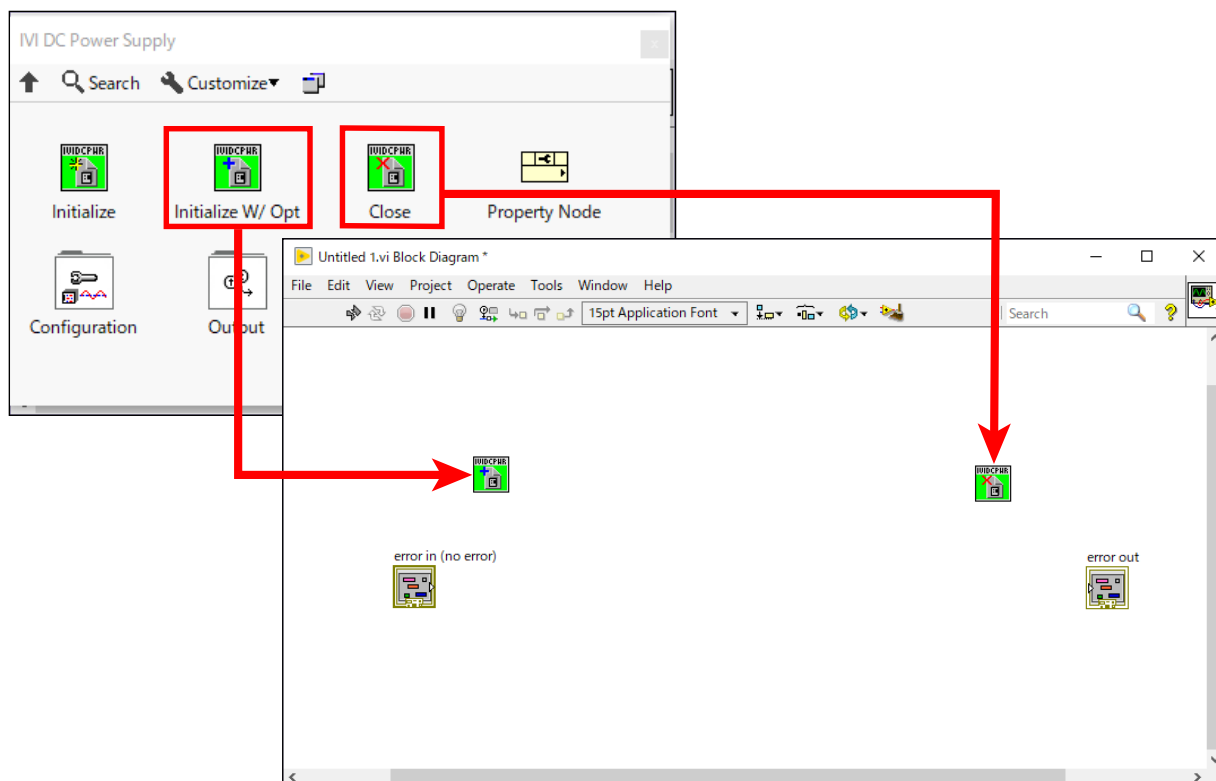
### Memo

When the IVI Class Drivers palette is not found, the appropriate version of the IVI Compliance Package may not be installed.

## 3

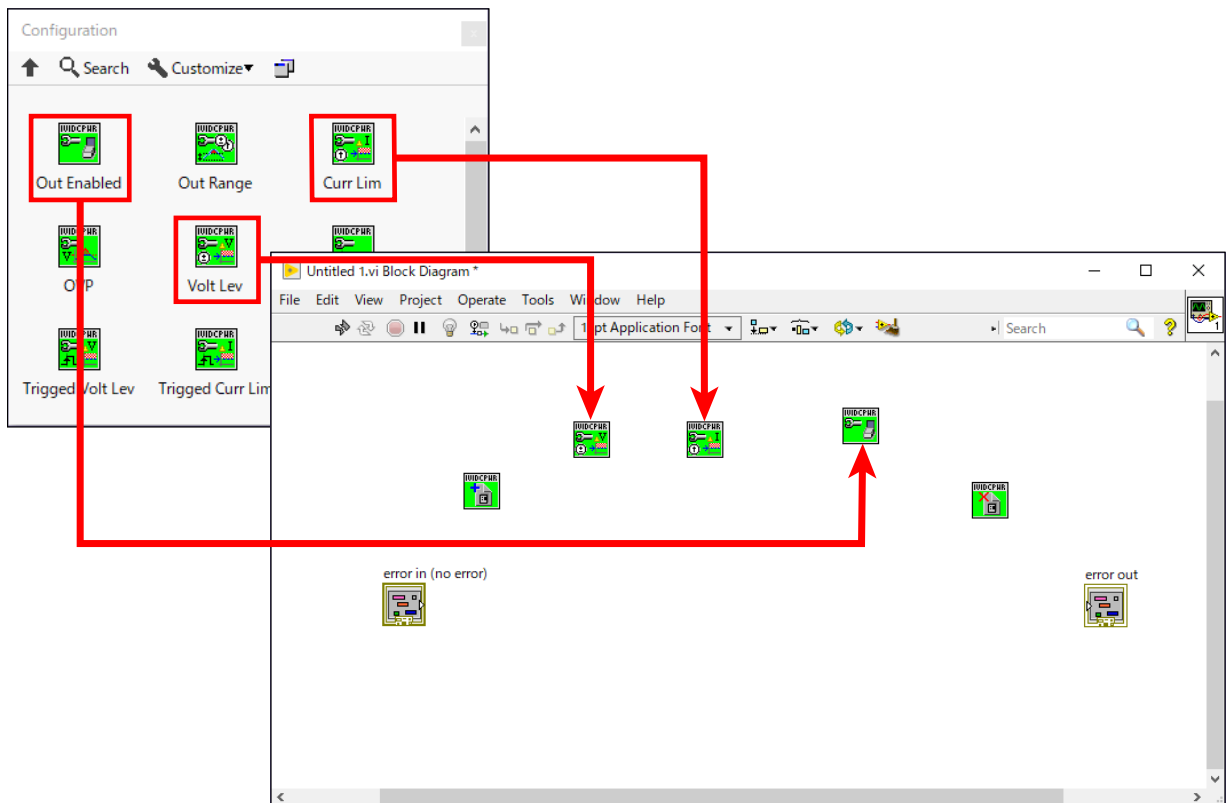
Add the following VIs to the Block Diagram screen:

- Initialize W/Opt (Initialize With Options.vi)
- Close (Close.vi)



## 4 Open the [Configuration] palette and add the following.

Parameter	VI
Parameter to set voltage	Configure Voltage Level.vi
Parameter to set current	Configure Current Limit.vi
Parameter to set output	Configure Output Enabled.vi



Then, set parameters to the added functions.



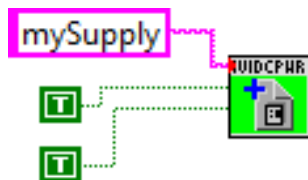
## Setting Parameters

This section describes how to set parameters to the added functions to configure the program. It shows an example of setting the voltage to 20 V and the current to 2 A to turn on the output.

### 1 Bridge the [id query] and [reset device] parameters to Initialize With Options.vi, and specify the logical name of the virtual instrument specified for [logical name] using NI-MAX.

In Initialize With Options.vi for class drivers, specify a virtual instrument (IVI logical name) instead of a VISA resource name (VISA address).

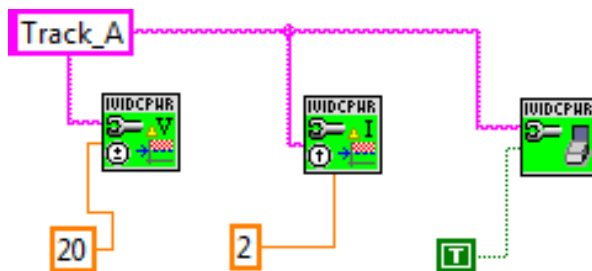
In this example, [mySupply] is used.



### 2 Add parameters that set voltage, current, and output to each VI.

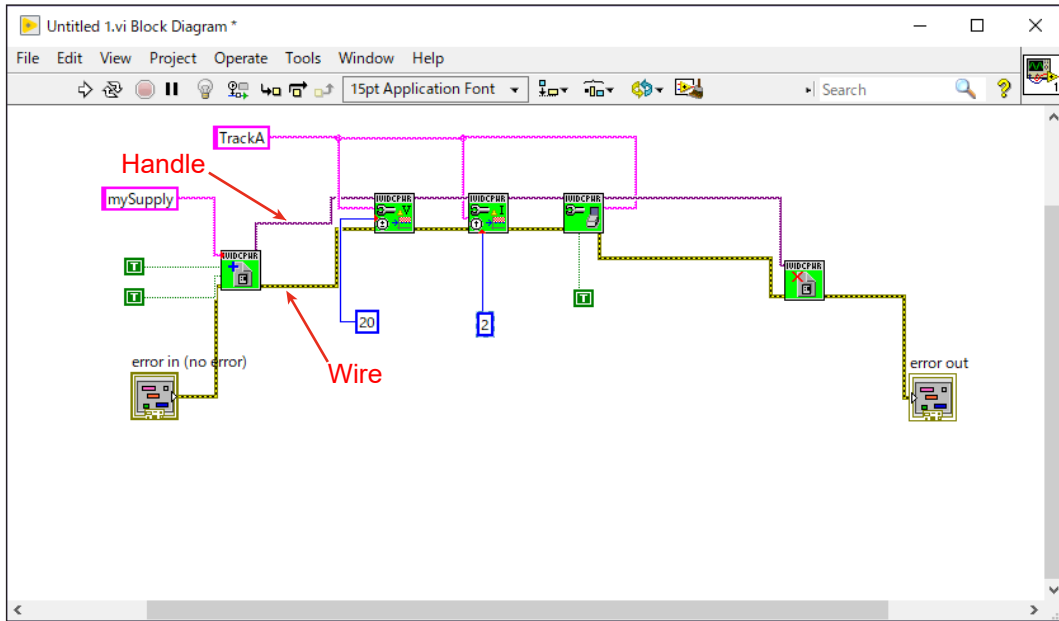
### 3 Bridge the character string of the DC power supply channel name to be controlled to the same three VIs as in Step 2 on a shared line.

For programming using class drivers, specify the virtual name of the channel that is specified for the virtual instrument. In this example, [Track\_A] is the channel name.



# 4

Connect the [error in] cluster to the [error out] cluster by wire, and connect the instrument session (handle).



## Function Descriptions

This section describes the details and settings of the functions (VIs) that configure the program.

### Starting the Session

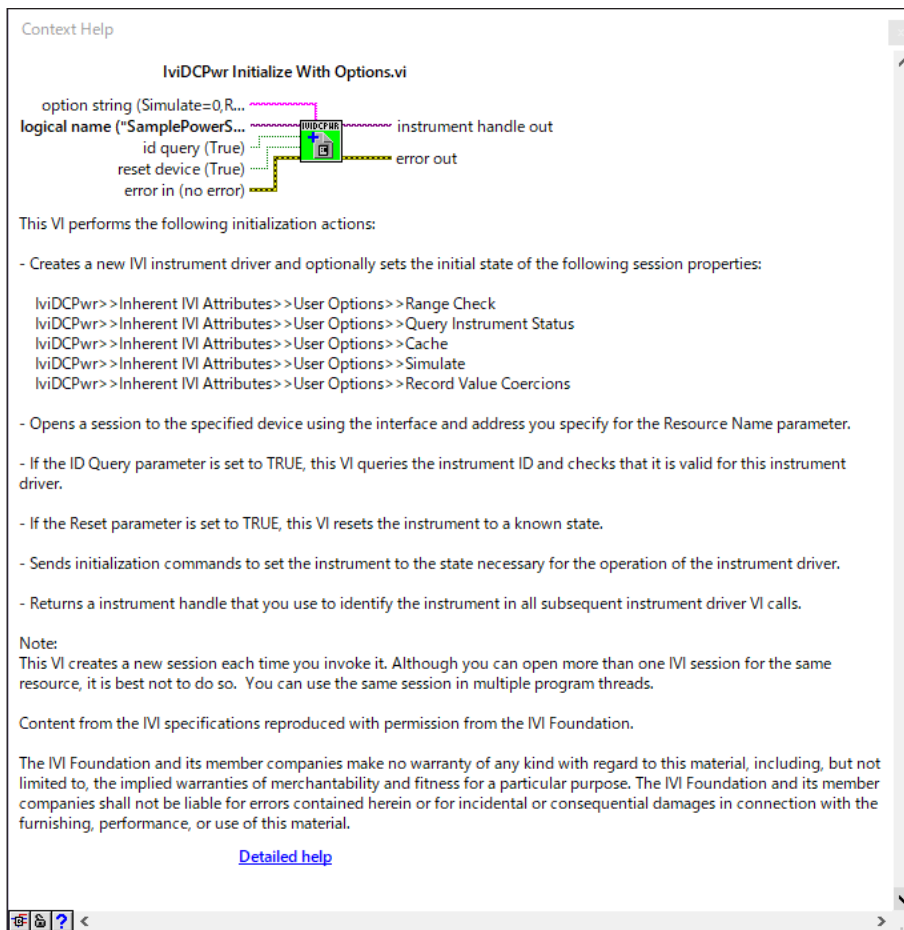
---

To start the session with class drivers, use Initialize With Options.vi.

This example uses the IviDCPwr class driver. “IviDCPwr,” as prefix to be added to VIs (functions), is specific to the IviDCPwr class driver.

#### Memo

Programs that use class drivers do not depend on instrument drivers of specific models such as kipwr (our PWR-01 Series DC power supply) and AgN57xx (N5700 Series DC power supply manufactured by Agilent).



- With class drivers, you cannot directly bridge a VISA address to Initialize With Options.vi. Specify the logical name of the virtual instrument that was created using NI-MAX.

### Memo

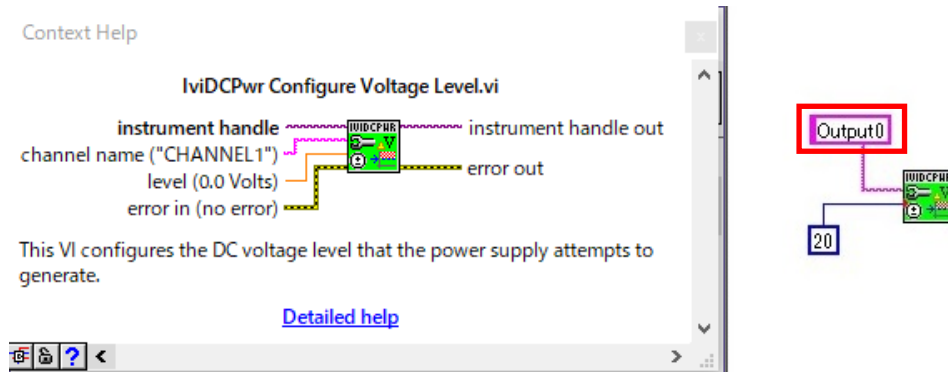
The class driver references the logical name to search for the appropriate instrument driver DLL (Software Module) and VISA address (Hardware Asset), and ultimately invokes the kipwr Initialize With Options.vi function indirectly.

- The contents bridged to Option String (Cache, Range Check, Record Coercions, Interchange Check, Query Instrument Status, and Driver Setup strings) are the same as when using the specific driver. (go to page 19)
- When the Cache, Range Check, Record Coercions, Interchange Check, Query Instrument Status, and Driver Setup strings bridged to Option String are omitted, each default value is the same as the one specified in the IVI Configuration's [Driver Session] > [General] page.

## Setting the Channel Name

The IVI instrument driver was designed on the premise that, when supporting power supplies, oscilloscopes, and so on, the instrument has multiple channels.

Therefore, many driver functions that operate instrument panel settings may require you to specify the channel for the [channel name] parameter.



The figure shown above uses a class driver, and the channel name “Output0,” which can only be applied to a particular instrument driver (the kipwr driver in this case) is specified.

Even when a channel name that depends on a particular instrument driver is specified for the class driver function, instrument measurement is possible, but interchangeability is spoiled.

For example, because “Output1” is the valid channel name for the AgN57xx instrument driver, if “Output0” is the specified channel name, then you cannot switch the instrument with AgN57xx without rewriting the program.

To actualize interchangeability in programming using class drivers, specify a virtual name for the channel in IVI Configuration, and specify the virtual name for the [channel name] of the function.

For example, in “Creating a Virtual Instrument” (go to page 24), we added the name “Track\_A” as a virtual name, and set it to convert into the physical name “Output0.” To perform programming that does not depend on a model-specific instrument driver, specify “Track\_A” for [channel name] of the Configure Voltage Level.vi mentioned above.

If you have switched the instrument driver, you can continue operation just by changing some items in IVI Configuration, without changing the application itself. (go to page 39)

### Memo

- Do not manually edit the XML file (IviConfigurationStore.xml) where the IVI Configuration setting information is stored.
- The IVI Configuration is shared between all 32-bit/64-bit measurement applications and all log-on users on the same PC.

## Closing the Session


---

To close the instrument driver's session, use Close.vi.

### ClassPrefix Close

#### IVI Inherent Function

To form the *ClassPrefix* VI name, use the prefix of the class driver that you are working with. For example, if you are working with the IviDmm class driver, use the prefix IviDmm to create the correct VI name, IviDmm Close.

instrument handle  error out

error in (no error)

## Switching the Instrument

If you have switched the instrument, you can continue operation just by changing the Driver Session of the virtual instrument (IVI Configuration).

You do not need to change the application itself.

Change the following three settings for Driver Session.

Item	Setting
[Hardware] tab > [Hardware Assets] > [Resource Descriptor]	VISA address to which the instrument is connected
[Software] tab > [Software Module]	Instrument driver to be used
[Virtual Names] tab > [Physical Names]	Physical mapping destination name of the virtual channel name

If you correctly set Driver Session according to the switched instrument, operation can continue without having to compile and link the application again.

Following the examples in this guide, if you switched the instrument from a Kikusui PWR-01 Series DC power supply (an instrument hosted by the kipwr instrument driver) to an Agilent N5700 Series DC power supply (an instrument hosted by the AgN57xx driver), change the [mySupply] settings as shown below.

Item	Change
[Hardware] tab > [Hardware Assets] > [Resource Descriptor]	VISA address to which the Kikusui PWR-01 Series DC power supply is connected  => VISA address to which the Agilent N5700 Series DC power supply is connected
[Software] tab > [Software Module]	“kipwr” => “AGN57xx”
[Virtual Names] tab > [Physical Names]	“Output0” => “Output1”

### Memo

- The interchangeability feature using IVI class drivers does not guarantee the correct operation of instruments before and after switching. Fully verify that your system correctly functions after switching the instruments.