Document Version 1.0

IVI Driver Getting Started Guide

For Multiplexer SC6540,

Overview

This application note will describe the installing instructions and several programming examples for IVI Instrument Driver of Scanner. To understand more about the IVI drivers, please refer to the website of IVI Foundation. For more detail of the SC6540 IVI driver, please check the help document, SC6540.chm, located at the path of *<Program Files>\IVI Foundation\IVI\Drivers\SC6540*.

1. IVI Driver Setup

Instructions on downloading and Installing IVI Instrument Drivers from website. Download and install Shared Components from IVI Foundation Website.

2. Getting Started with C#

A tutorial using IVI driver establishes communication with the instrument by C# programming.

3. Getting Started with C++

A tutorial using IVI driver establishes communication with the instrument by C++ programming.

4. Getting Started with Python

A tutorial using IVI driver establishes communication with the instrument by Python programming.

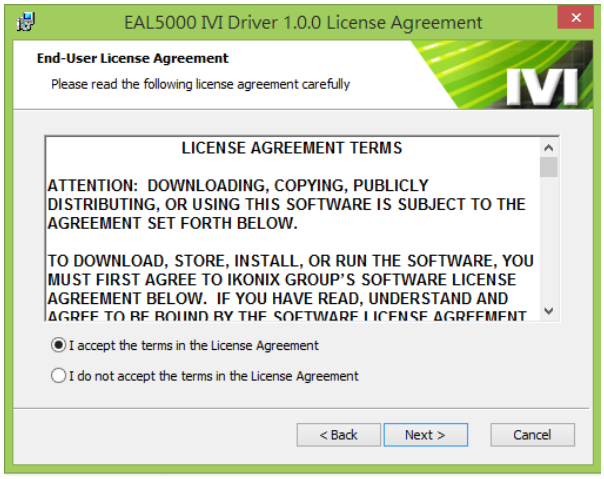
5. Getting Started with LabVIEW

A tutorial using IVI driver establishes communication with the instrument by LabVIEW programming.

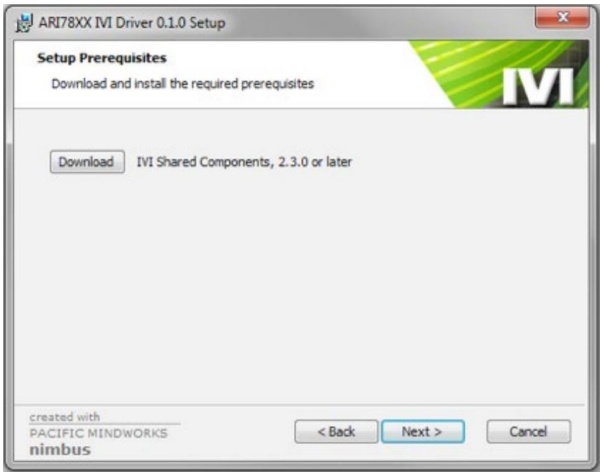
28105 N. Keith Drive
Lake Forest, IL 60045 USA
Toll Free: 1-800-858-TEST (8378) US/Canada
Phone: 1-847-367-4077 | Fax: 1-847-367-4080 | www.arisafety.com

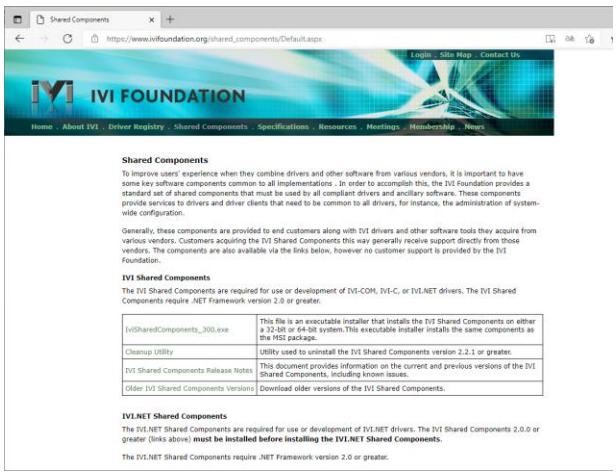
1. IVI Driver Setup

After downloading the IVI Driver, run the self-extracting setup file and you will see the installation wizard to start setup. Please follow the below instruction to complete the installation.



The setup will detect if IVI Shared Components are installed. If prompted with the following screen, click on Download, The IVI Foundation Website will be opened.

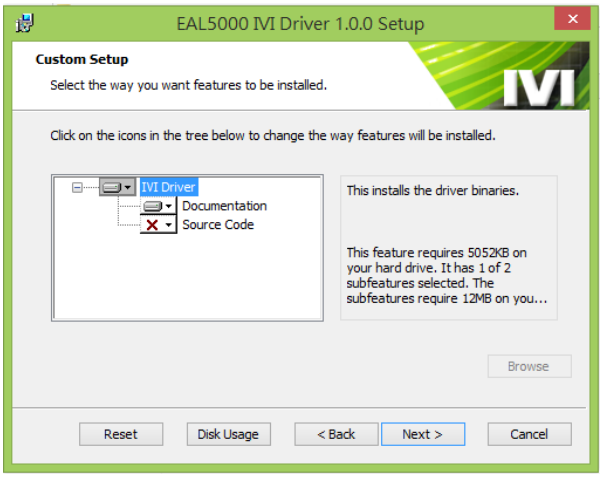


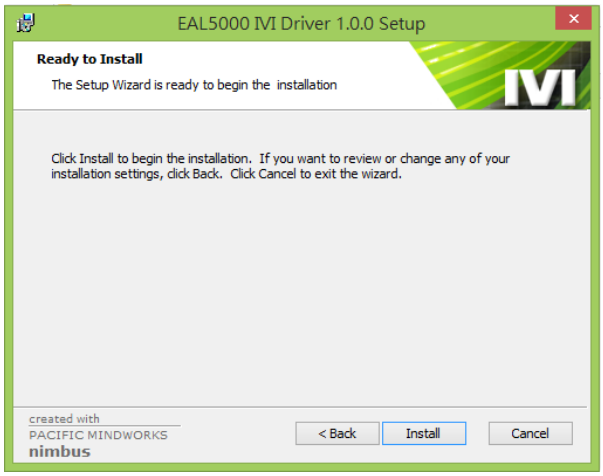


Please download the latest IVI Shared Components either 32-bit or 64-bit version. After downloading, install the shared components and continue the installation.

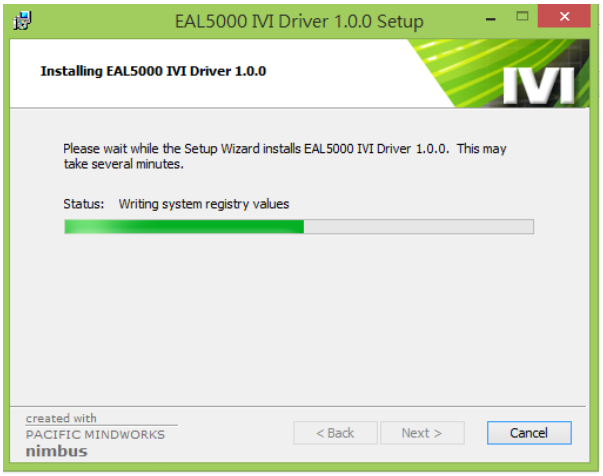
28105 N. Keith Drive
Lake Forest, IL 60045 USA
Toll Free: 1-800-858-TEST (8378) US/Canada
Phone: 1-847-367-4077 | Fax: 1-847-367-4080 | www.arisafety.com

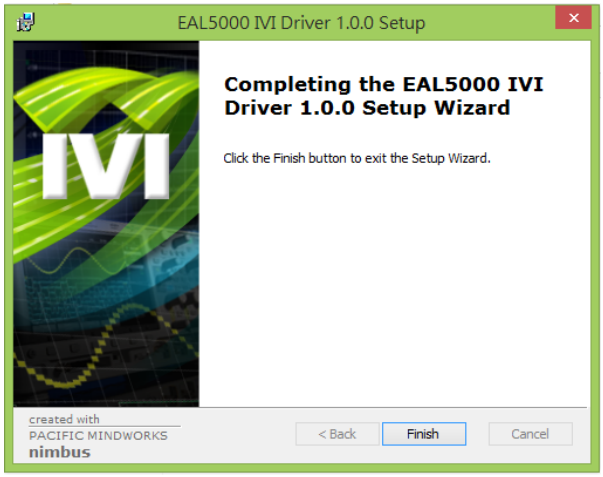
After the IVI Shared Components are installed, please follow the steps to complete installation.





There are options for installing the source code of the IVI Driver, if it is necessary.





The IVI driver would be installed under the path of "*<Program Files>\IVI Foundation\IVI*". For the files with "*.dll" extension name would be located in the "Bin" folder. And the necessary help documents will be in the folder of "..\Drivers\SC6540".

28105 N. Keith Drive
Lake Forest, IL 60045 USA
Toll Free: 1-800-858-TEST (8378) US/Canada
Phone: 1-847-367-4077 | Fax: 1-847-367-4080 | www.arisafety.com

2. Getting Started with C#

Introduction

This chapter describes the procedures of using the IVI-COM driver of IKONIX Group by C# programming language. In this exercise, the programmer could import the driver and complete a short program controlling the device step-by step.

The C# could call IVI-C driver, either. However, we suggest that an IVI-COM interop would be easier for you to develop the program.

Requirements

- SC6540 IVI Driver
- IVI Shared Components, *https://www.ivifoundation.org/shared_components/Default.aspx*
- Microsoft Visual Studio or other IDEs
- A Main Scanner with a HV Module and a GB Module

Download the Drivers

Please go to the website of the Associated Research to download the latest version of IVI drivers or contact the vendors. Follow the steps and instructions in Chapter 1 to complete the installation.

References

On the website of IVI Foundation, there are documentations you might be interested while implementing controlling the devices. You could find the resources of developing with an IVI driver, *https://www.ivifoundation.org/resources/default.aspx*. The IVI Shared Components could be download from *https://www.ivifoundation.org/shared_components/Default.aspx*. There are several documents on the website for understanding the IVI.

In the installed directory, there are several documents for your reference understanding the SC6540 IVI Driver. A help file, ***SC6540.chm***, would be located at the path of *<Program Files>\IVI Foundation\IVI\Drivers\SC6540*. In this help file, you could find all of the provided functions and their hierarchy.

There are four types of sample code for your reference which are located at the path of *<Program Files>\IVI Foundation\IVI\Drivers\SC6540\Examples*, including C#, C++, Python and LabVIEW as well.

Development

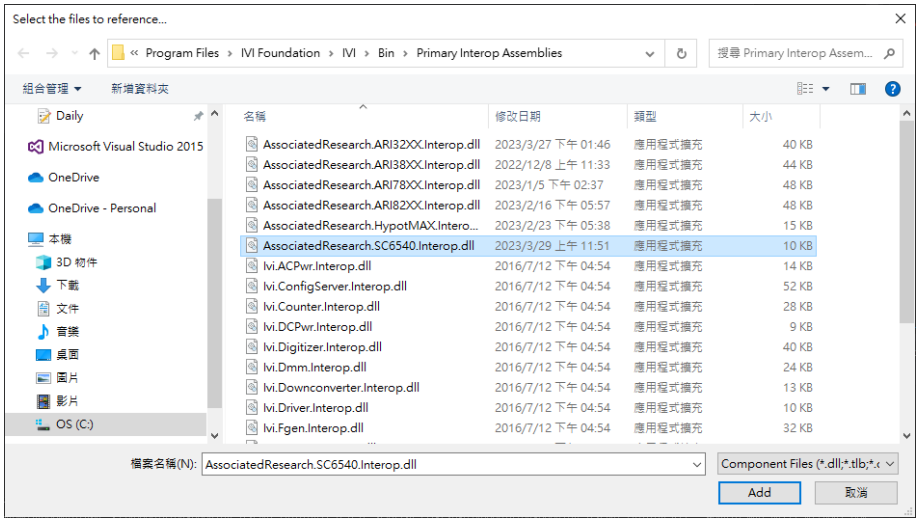
1 Create a C# project

1.1 Open Visual Studio IDE and create a new C# console project.

2 Import Libraries

2.1 Right-click on the reference and select ***Add Reference*** in the solution explorer

2.2 Click on the Browse button and go to the path of "*<Program Files>\IVI Foundation\IVI\Bin\Primary Interop Assemblies*" and choose ***AssociatedResearch.SC6540.Interop.dll*** and ***Ivi.Driver.Interop.dll.***



2.3 Declare to use the name spaces for the interop assemblies that are specified to reference in the previous section.

```
using AssociatedResearch.SC6540.Interop;
```

3 Start programming

3.1 Create an object of the driver and use the initialize method to build up the connection.

```
//Initialize
//
var driver = new SC6540();
string resourceName = "ASRL4::INSTR";
string optionString = "Cache=false, InterchangeCheck=false, QueryInstrStatus=true,
RangeCheck=false, RecordCoercions=false, Simulate=false";
driver.Initialize(resourceName, true, false, optionString);

// Disable all channels
driver.Execution.DisableAllChannels();
```

For more detail for the parameters of the ***Initialize()*** method, please refer to the help document, ***SC6540.chm***, which is located at "*<Program Files>\IVI Foundation\IVI\Drivers\SC6540*".

The first parameter ***resourceName*** is a string type and indicates the interfaces type and address of the connection. The resource name,"ASRL4::INSTR", represents a serial port with

address 4. For example, a GPIB connection could be `"GPIB0::8::INSTR"`. For TCP/IP connection, it will be in the format of `"TCPIP0::192.168.0.1::10001::SOCKET"`. The 10001 is the TCP/IP connection port of SC6540. The address depends on the configurations.
The ***optionString***, `"Cache=false, InterchangeCheck=false, QueryInstrStatus=true, RangeCheck=false, RecordCoercions=false, Simulate=false"`, configures the options for the IVI driver with enabled ***QueryInstrStatus*** which will check the status at the end of each user operation. Please refer to the ***SC6540.chm*** for more details.
After initialization, it is suggested to disable all of the channels by ***DisableAllChannels()*** method at the first step.

3.2 Configure Withstand Channels

```
// Withstand test (ACW, DCW)
// Enable Withstand (High) channels
int[] hvChannels = { 1, 2, 3 };
driver.Execution.ConfigureWithstandChannels(hvChannels);

// Enable Return (Low) channels
int[] rtnChannel = { 4, 5, 6 };
driver.Execution.ConfigureReturnChannels(rtnChannel);

// After the multiplexer was configured, the safety tester could start output for withstand
test on those connections.
Thread.Sleep(1000);

// Disable all channels
driver.Execution.DisableAllChannels();
```

For some units, we may need a multiplexer, SC6540, to extend the capacity on a HIPOT tester, such as Omnia2, HypotUltra, Hypot4 of Associated Research. Before a withstand test operated, the multiplexer needs to be configured to the test point.
Both the ***ConfigureWithstandChannels()*** and ***ConfigureReturnChannels()*** method needs an integer array as input. After, the HV and Return channels are switched, we could add a procedure performing a withstand output. And disable the relays by ***DisableAllChannels()***.

3.3 Configure Continuity Channels

```
// Continuity (CONT)
// Enable Continuity (High) channels
int[] contChannels = { 4, 5 };
driver.Execution.ConfigureContinuityChannels(contChannels);

// Enable Return (Low) channels
int[] contRtnChannel = { 7 };
driver.Execution.ConfigureReturnChannels(contRtnChannel);

// After the multiplexer was configured, the safety tester could start output for continuity
test on those connections.
Thread.Sleep(1000);
```

The ***ConfigureContinuityChannels()*** method takes an integer array as input. For the connection on the return circuits, we could invoke the method of ***ConfigureReturnChannels()*** which is shared with the withstand channels.

3.4 Configure Ground Bond Channel

```
//Ground Bond Test (GND)
// Enable Ground Bond test channel
driver.Execution.ConfigureGndChannel(2);

// After the multiplexer was configured, the safety or ground bond tester could start output
for ground bond test on those connections.
Thread.Sleep(1000);
```

Only one ground bond channel could be enabled at one time. Therefore, the ***ConfigureGndChannel()*** takes a integer as input parameter at once. When a ground bond channel was set, the other switches would be disabled automatically.

3.5 Configure SC6540 for Dual Check

```
// DualCheck (Withstand test + Ground bond test)
// Enable Grond Bond channel and Withstand Channel at the same time.
driver.Execution.ConfigureGndAndWithstandChannel(5, "HLOOOOL");
// After the multiplexer was configured, the safety tester could start dual check on those
connections.
Thread.Sleep(1000);
```

The DualCheck is the function allows the instrument to simultaneously run a hipot and AC Ground Bond test. By ***ConfigureGndAndWithstandChannel()***, we could configure both ground bond channel and withstand channels at the same time. This function takes one integer input as ground bond channel and one string as the raw configuration for HV module. The configurations will be H (High), L (Low) or O (Open). For example, the syntax for this command would be “HLOOOOL” which indicates the channel 1 is connected to HV and channel 2 and 7 would be connected to Return.

3.6 Close the session

```
driver.Execution.DisableAllChannels();
driver.Close();
Console.WriteLine("Done - Press Enter to Exit");
Console.ReadLine();
```

Close() would close the I/O session to the instrument.

4 Completed example

The completed sample code could be found at the path of “*<Program Files>\IVI Foundation\IVI\Drivers\SC6540\Examples*”.

3. Getting Started with C++

Introduction

This chapter describes the procedures of using the IVI-COM driver of IKONIX Group by C++ programming language. In this exercise, the programmer could import the driver and complete a short program controlling the device step-by step.

Requirements

- SC6540 IVI Driver
- IVI Shared Components, *https://www.ivifoundation.org/shared_components/Default.aspx*
- Microsoft Visual Studio or other IDEs
- A Main Scanner with a HV Module and a GB Module

Download the Drivers

Please go to the website of the Associated Research to download the latest version of IVI drivers or contact the vendors. Follow the steps and instructions in Chapter 1 to complete the installation.

References

On the website of IVI Foundation, there are documentations you might be interested while implementing controlling the devices. You could find the resources of developing with an IVI driver, *https://www.ivifoundation.org/resources/default.aspx*. The IVI Shared Components could be download from *https://www.ivifoundation.org/shared_components/Default.aspx*. There are several documents on the website for understanding the IVI.

In the installed directory, there are several documents for your reference understanding the SC6540 IVI Driver. A help file, ***SC6540.chm***, would be located at the path of *<Program Files>\IVI Foundation\IVI\Drivers\SC6540*. In this help file, you could find all of the provided functions and their hierarchy.

There are three types of sample code for your reference which are located at the path of *<Program Files>\IVI Foundation\IVI\Drivers\SC6540\Examples*, including C#, C++ and Python as well.

Development

1 Create a C++ project

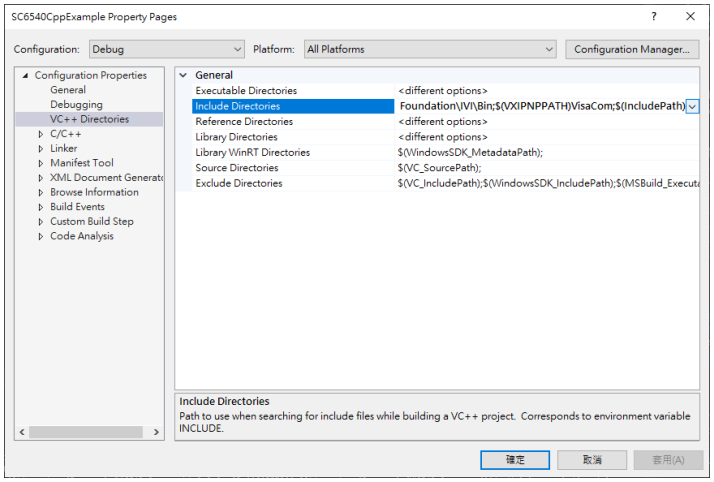
1.1 Open Visual Studio or any other IDEs and create a new C++ console project.

2 Include Directories

2.1 Right-click on the project and select ***properties***.

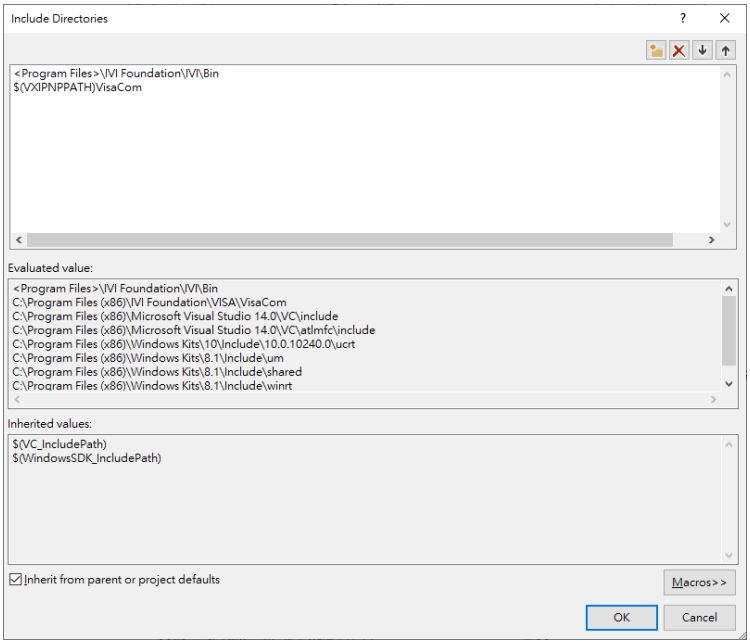
2.2 Expand the ***Configuration Properties*** and select ***VC++ Directories*** on the left menu.

2.3 Click on the drop-down column of the ***Include Directories*** and select ***<Edit..>*** to open the edit window.



2.4 Select the ***New Line*** button to add an include directories. There will be two necessary paths need to be added.

- ***<Program Files>\IVI Foundation\IVI\Bin***
- ***$(VXIPNPPATH)VisaCom***



28105 N. Keith Drive
Lake Forest, IL 60045 USA
Toll Free: 1-800-858-TEST (8378) US/Canada
Phone: 1-847-367-4077 | Fax: 1-847-367-4080 | www.arisafety.com

2.5 Click ***OK*** to complete including the directories.

2.6 Use the `#import` operator to import the necessary DLLs

```
#include "stdafx.h"
#include <iostream>
#include <string>
#import <IviDriverTypeLib.dll> no_namespace
#import <GlobMgr.dll> no_namespace
#import <SC6540.dll> no_namespace
```

3 Start programming

3.1 Create an instance of the driver by pointer and use the initialize method to build up the connection.

```
HRESULT hr = ::CoInitialize(NULL);
ISC6540Ptr driver(__uuidof(SC6540));
_bstr_t resourceName = "ASRL4::INSTR";
_bstr_t optionString = "Cache=false, InterchangeCheck=false, QueryInstrStatus=true,
RangeCheck=false, RecordCoercions=false, Simulate=false";
driver->Initialize(resourceName, true, false, optionString);

// Disable all channels
driver->Execution->DisableAllChannels();
```

For more detail for the parameters of the ***Initialize***() method, please refer to the help document, ***SC6540.chm*** located at "*<Program Files>\IVI Foundation\IVI\Drivers\SC6540*".

The first parameter ***resourceName*** is a string type and indicates the interfaces type and address of the connection. The resource name,`"ASRL4::INSTR"`, represents a serial port with address 4. For example, a GPIB connection could be `"GPIB0::8::INSTR".` For TCP/IP connection, it will be in the format of `"TCPIP0::192.168.0.1::10001::SOCKET"`. The 10001 is the TCP/IP connection port of SC6540. To know more about the options of the ***Initialize()*** method, please refer to the ***SC6540.chm*** document.

After initialization, it is suggested to disable all of the channels by ***DisableAllChannels()*** method at the first step.

4.1 Configure Withstand Channels

```
// Withstand test (ACW, DCW)
// Enable Withstand (High) channels
int hvChannels[3] = { 1,2,3 };
SAFEARRAY* hvChannelsSafeArray = ConvertToSafeArray(hvChannels,
sizeof(hvChannels)/sizeof(*hvChannels));
driver->Execution->ConfigureWithstandChannels(&hvChannelsSafeArray);
SafeArrayDestroy(hvChannelsSafeArray);

//// Enable Return (Low) channels
int rtnChannels[3] = { 4,5,6 };
SAFEARRAY* rtnChannelsSafeArray = ConvertToSafeArray(rtnChannels, sizeof(rtnChannels) /
sizeof(*rtnChannels));
driver->Execution->ConfigureReturnChannels(&rtnChannelsSafeArray);
SafeArrayDestroy(rtnChannelsSafeArray);
// After the multiplexer was configured, the safety tester could start output for withstand
test on those connections->
Sleep(1000);

// Disable all channels
```

```
driver->Execution->DisableAllChannels();
```

For some units, we may need a scanner, SC6540, to extend the capacity on a HIPOT tester, such as Omnia2, HypotUltra, Hypot4 of Associated Research. Before a withstand test operated, the multiplexer needs to be configured to the test point.
Both the ***ConfigureWithstandChannels()*** and ***ConfigureReturnChannels()*** method needs an integer array as input. After, the HV and Return channels are switched, we could add a procedure performing a withstand output. And disable the relays by ***DisableAllChannels()***. There is a C++ function, ***ConvertToSafeArray()***, in the above sample which is to convert the integer array to a SAFEARRAY type.

```
//ConvertToSafeArray is the function to create a SAFEARRAY type from a std::int[]
//
SAFEARRAY* ConvertToSafeArray(int channels[], int size)
{
    CComSafeArray<int> csaData(size);
    for (int i = 0; i < size; i++)
    {
        csaData.SetAt(i, channels[i]);
    }
    return csaData.Detach();
}
```

For the COM components, it take the SAFEARRAY type as the array parameters. Therefore, we need to create it with assigned size. Also, it is suggested to release the SAFEARRAY by ***SafeArrayDestroy()*** after it is no longer been used.

4.2 Configure Continuity Channels

```
// Continuity (CONT)
// Enable Continuity (High) channels
int contChannels[2] = { 4, 5 };
SAFEARRAY* contChannelsSafeArray = ConvertToSafeArray(contChannels, sizeof(contChannels) /
sizeof(*contChannels));
driver->Execution->ConfigureContinuityChannels(&contChannelsSafeArray);
SafeArrayDestroy(contChannelsSafeArray);

// Enable Return (Low) channels
int contRtnChannel[1] = { 7 };
SAFEARRAY* contRtnChannelsSafeArray = ConvertToSafeArray(contRtnChannel,
sizeof(contRtnChannel) / sizeof(*contRtnChannel));
driver->Execution->ConfigureReturnChannels(&contRtnChannelsSafeArray);
SafeArrayDestroy(contRtnChannelsSafeArray);

// After the multiplexer was configured, the safety tester could start output for continuity
test on those connections
Sleep(1000);
```

The ***ConfigureContinuityChannels()*** method takes an SAFEARRAY as input which is converted by ***ConvertToSafeArray()***. For the connection on the return circuits, we could invoke the method of ***ConfigureReturnChannels()*** which is the same with the withstand channels.

4.3 Configure Ground Bond Channel

```
//Ground Bond Test (GND)
// Enable Ground Bond test channel
driver->Execution->ConfigureGndChannel(2);

// After the multiplexer was configured, the safety or ground bond tester could start output
for ground bond test on those connections
Sleep(1000);
```

Only one ground bond channel could be enabled at one time. Therefore, the ***ConfigureGndChannel()*** takes a integer as input parameter at once. When a ground bond channel was set, the other switches would be disabled automatically.

4.4 Configure SC6540 for Dual Check

```
// DualCheck (Withstand test + Ground bond test)
// Enable Grond Bond channel and Withstand Channel at the same time
driver->Execution->ConfigureGndAndWithstandChannel(5, "HHHLLLOO");

// After the multiplexer was configured, the safety tester could start dual check on those
connections
Sleep(1000);
```

The DualCheck is the function allows the instrument to simultaneously run a hipot and AC Ground Bond test. By ***ConfigureGndAndWithstandChannel()***, we could configure both ground bond channel and withstand channels at the same time. This function takes one integer input as ground bond channel and one string as the raw configuration for HV module. The configurations will be H (High), L (Low) or O (Open). For example, the syntax for this command would be “HLOOOOL” which indicates the channel 1 is connected to HV and channel 2 and 7 would be connected to Return.

3.2 Close the session

```
driver->Execution->DisableAllChannels();
driver->Close();
std::cout << "Done - Press Enter to Exit" <<std::endl;
std::cin.get();
```

Close() would close the I/O session to the instrument.

4 Completed example

The completed sample code could be found at the path of “*<Program Files>\IVI Foundation\IVI\Drivers\SC6540\Examples*”.

4. Getting Started with Python

Introduction

This chapter describes the procedures of using the IVI-COM driver of Ikonix Group by Python programming language. In this exercise, the programmer could import the driver and complete a short program controlling the device step-by step.

Requirements

- SC6540 IVI Driver
- IVI Shared Components, *https://www.ivifoundation.org/shared_components/Default.aspx*
- Python IDE
- Comtypes Library (pip install comtypes)
- A Main Scanner with a HV Module and a GB Module

Download the Drivers

Please go to the website of the Associated Research to download the latest version of IVI drivers or contact the vendors. Follow the steps and instructions in Chapter 1 to complete the installation.

References

On the website of IVI Foundation, there are documentations you might be interested in while implementing controlling the devices. You could find the resources of developing with an IVI driver, *https://www.ivifoundation.org/resources/default.aspx*. The IVI Shared Components could be download from *https://www.ivifoundation.org/shared_components/Default.aspx*. There are several documents on the website for understanding the IVI.

In the installed directory, there are several documents for your reference understanding the SC6540 IVI Driver. A help file, ***SC6540.chm***, would be located at the path of *<Program Files>\IVI Foundation\IVI\Drivers\SC6540*. In this help file, you could find all of the provided functions and their hierarchy.

There are three types of sample code for your reference which are located at the path of *<Program Files>\IVI Foundation\IVI\Drivers\SC6540\Examples*, including C#, C++ and Python as well.

Development

1 Install the ***Comtypes*** library

```
pip install cometypes
```

In order to call an external com DLL in Python, you will need ***comtypes*** library installed.

2 Create a Python file

2.1 Open any IDE of Python and create a new Python file.

3 Import Libraries

3.1 Import the cometypes library and SC6540_64.dll

```
import time
import comtypes
import comtypes.client as cc

# Import the IVI-COM DLL
cc.GetModule('SC6540.dll')
from comtypes.gen import SC6540Lib
```

4 Start programming

4.1 Create an object of the driver and use the initialize method to build up the connection.

```
driver = cc.CreateObject('SC6540.SC6540', interface=SC6540Lib.ISC6540)
# Initialize Driver and make connection
optionString = 'Cache=false, InterchangeCheck=false, QueryInstrStatus=true,
RangeCheck=false, RecordCoercions=false, Simulate=false'
driver.Initialize('ASRL4::INSTR', True, False, optionString)

# Disable all channels
driver.Execution.DisableAllChannels();
```

For more detail for the parameters of the ***Initialize***() method, please refer to the help document, ***SC6540.chm*** located at "*<Program Files>\IVI Foundation\IVI\Drivers\SC6540*".

The first parameter ***ResourceName*** is a string type and indicates the interfaces type and address of the connection. The resource name,`"ASRL4::INSTR"`, represents a serial port with address 4. For example, a GPIB connection could be `"GPIB0::8::INSTR".` For TCP/IP connection, it will be in the format of `"TCPIP0::192.168.0.1::10001::SOCKET"`. The 10001 is the TCP/IP connection port of SC6540.

There are other parameters for the options of the ***Initialize()*** method, please refer to the ***SC6540.chm*** for more detail. For example, `"QueryInstrStatus=true"` makes the session automatically query the error status for each command was sent.

28105 N. Keith Drive
Lake Forest, IL 60045 USA
Toll Free: 1-800-858-TEST (8378) US/Canada
Phone: 1-847-367-4077 | Fax: 1-847-367-4080 | www.arisafety.com

4.2 Configure Withstand Channels

```
# Withstand test (ACW, DCW)
# Enable Withstand (High) channels
hvChannels = { 1, 2, 3 };
driver.Execution.ConfigureWithstandChannels(hvChannels);

# Enable Return (Low) channels
rtnChannel = { 4, 5, 6 };
driver.Execution.ConfigureReturnChannels(rtnChannel);

# After the multiplexer was configured, the safety tester could start output for withstand
test on those connections.
time.sleep(1)

# Disable all channels
driver.Execution.DisableAllChannels();
```

For some units, we may need a multiplexer, SC6540, to extend the capacity on a HIPOT tester, such as Omnia2, HypotUltra, Hypot4 of Associated Research. Before a withstand test operated, the multiplexer needs to be configured to the test point.
Both the ***ConfigureWithstandChannels()*** and ***ConfigureReturnChannels()*** method needs an integer array as input. After, the HV and Return channels are switched, we could add a procedure performing a withstand output. And disable the relays by ***DisableAllChannels()***.

4.3 Configure Continuity Channels

```
# Continuity (CONT)
# Enable Continuity (High) channels
contChannels = { 4, 5 };
driver.Execution.ConfigureContinuityChannels(contChannels);

# Enable Return (Low) channels
contRtnChannel = { 7 };
driver.Execution.ConfigureReturnChannels(contRtnChannel);

# After the multiplexer was configured, the safety tester could start output for continuity
test on those connections.
time.sleep(1)
```

The ***ConfigureContinuityChannels()*** method takes an integer array as input. For the connection on the return circuits, we could invoke the method of ***ConfigureReturnChannels()*** which is shared with the withstand channels.

4.4 Configure Ground Bond Channel

```
#Ground Bond Test (GND)
# Enable Ground Bond test channel
driver.Execution.ConfigureGndChannel(2);

# After the multiplexer was configured, the safety or ground bond tester could start output
for ground bond test on those connections.
time.sleep(1)
```

Only one ground bond channel could be enabled at one time. Therefore, the ***ConfigureGndChannel()*** takes a integer as input parameter at once. When a ground bond channel was set, the other switches would be disabled automatically.

4.5 Configure SC6540 for Dual Check

```
# DualCheck (Withstand test + Ground bond test)
# Enable Grond Bond channel and Withstand Channel at the same time.
driver.Execution.ConfigureGndAndWithstandChannel(5, "HHHLLLOO");
# After the multiplexer was configured, the safety tester could start dual check on those
connections.
time.sleep(1)
```

The DualCheck is the function allows the instrument to simultaneously run a hipot and AC Ground Bond test. By ***ConfigureGndAndWithstandChannel()***, we could configure both ground bond channel and withstand channels at the same time. This function takes one integer input as ground bond channel and one string as the raw configuration for HV module. The configurations will be H (High), L (Low) or O (Open). For example, the syntax for this command would be “HLOOOOL” which indicates the channel 1 is connected to HV and channel 2 and 7 would be connected to Return.

4.6 Close the session

```
driver.Execution.DisableAllChannels();
driver.Close();
print("Done")
```

Close() would close the I/O session to the instrument.

5 Completed example

The completed sample code could be find at the path of “*<Program Files>\IVI Foundation\IVI\Drivers\SC6540\Examples*”.

28105 N. Keith Drive
Lake Forest, IL 60045 USA
Toll Free: 1-800-858-TEST (8378) US/Canada
Phone: 1-847-367-4077 | Fax: 1-847-367-4080 | www.arisafety.com

5. Getting Started with LabVIEW

Introduction

This chapter describes the procedures of using the IVI-COM driver of Ikonix Group by LabVIEW programming language. In this exercise, the programmer could learn how to import the driver and complete a short program controlling the device step-by step.

Even though the programmers could control the device by IVI Driver. For the LabVIEW programmer, we suggest that using LabVIEW plug & play driver would be easier for your programming and debugging. The LabVIEW driver from Ikonix Group are all made up with commands directly, so you could clearly check how the commands were sent to instruments.

Requirements

- SC6540 IVI Driver
- IVI Shared Components, *https://www.ivifoundation.org/shared_components/Default.aspx*
- National Instruments LabVIEW (This example was written in LabVIEW 2014)
- A Main Scanner with a HV Module and a GB Module

Download the Drivers

Please go to the website of the Associated Research to download the latest version of IVI drivers or contact the vendors. Follow the steps and instructions in Chapter 1 to complete the installation.

References

On the website of IVI Foundation, there are documentations you might be interested while implementing controlling the devices. You could find the resources of developing with an IVI driver, *https://www.ivifoundation.org/resources/default.aspx*. The IVI Shared Components could be download from *https://www.ivifoundation.org/shared_components/Default.aspx*. There are several documents on the website for understanding the IVI.

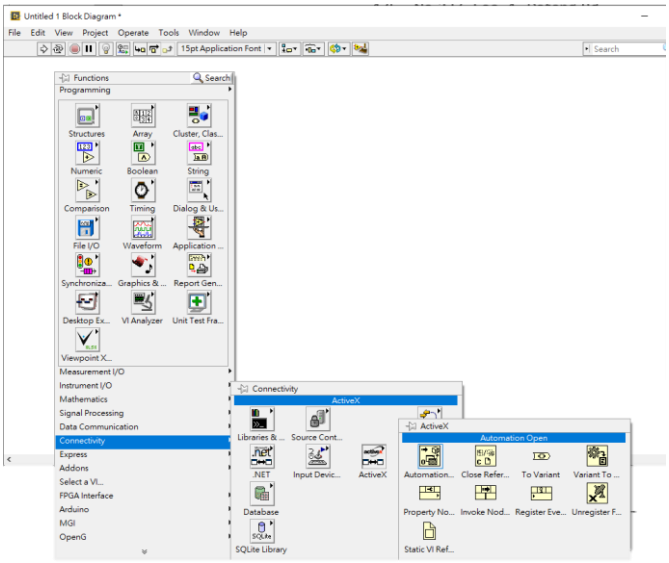
In the installed directory, there are several documents for your reference understanding the SC6540 IVI Driver. A help file, ***SC6540.chm***, would be located at the path of *<Program Files>\IVI Foundation\IVI\Drivers\SC6540*. In this help file, you could find all of the provided functions and their hierarchy.

There are three types of sample code for your reference which are located at the path of *<Program Files>\IVI Foundation\IVI\Drivers\SC6540\Examples*, including C#, C++ and Python as well.

Development

1 Open a new vi.

2 Import the DLL component.

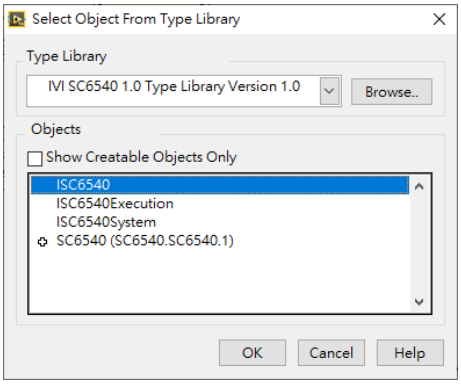


Open the ***Function Palette*** by right-clicking on the block diagram. Then select ***Connectivity*** -> ***ActiveX***. Select or drop the ***Automation Open*** function on the block diagram.

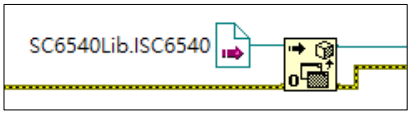
3 Right-clicking on the ***Automation Open*** and select ***Select ActiveX Class*** -> ***Browse*** will open a window for choosing the DLL.

4 Select the Browse button and select the file ***SC6540_64.dll*** located at *<Program Files>\IVI Foundation\IVI\Bin.* The ***IVI SC6540 Type Library*** would be added into the ***Type Libraries*** drop down menu.

5 Select ***ISC6540*** and then click ***OK*** to complete creating an object of SC6540 driver instance.



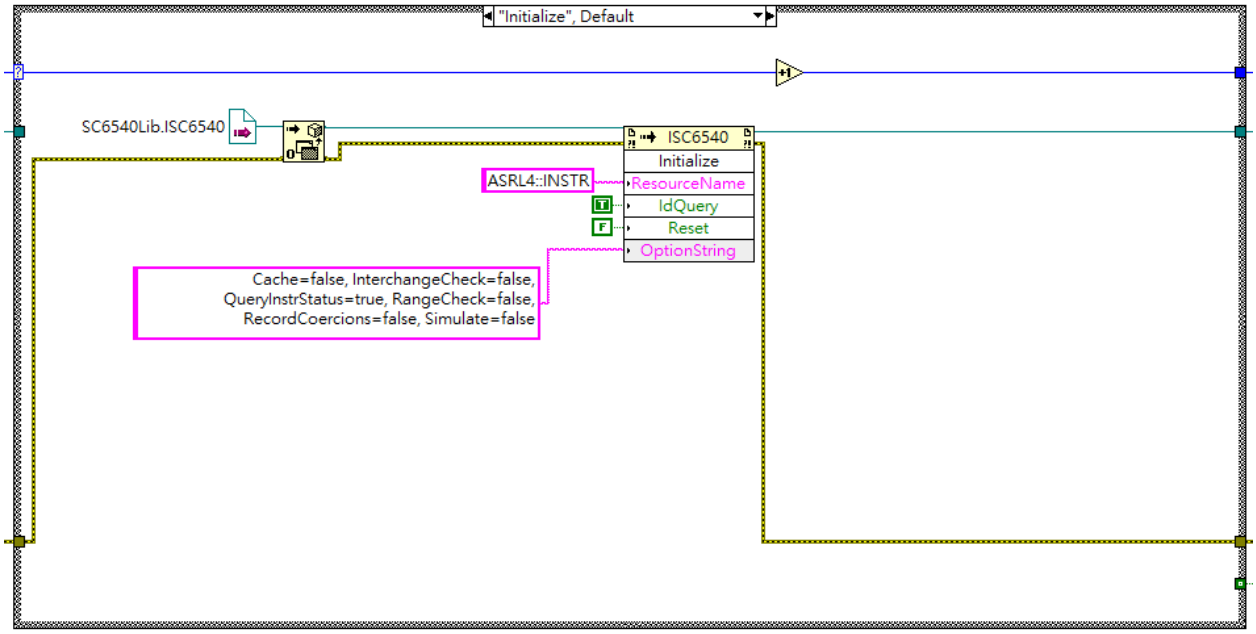
The Labview will automatically generate an ***Automation refnum*** of ***SC6540Lib.ISC6540*** control and connect to the Automation Open function.



28105 N. Keith Drive
Lake Forest, IL 60045 USA
Toll Free: 1-800-858-TEST (8378) US/Canada
Phone: 1-847-367-4077 | Fax: 1-847-367-4080 | www.arisafety.com

6 Start programming

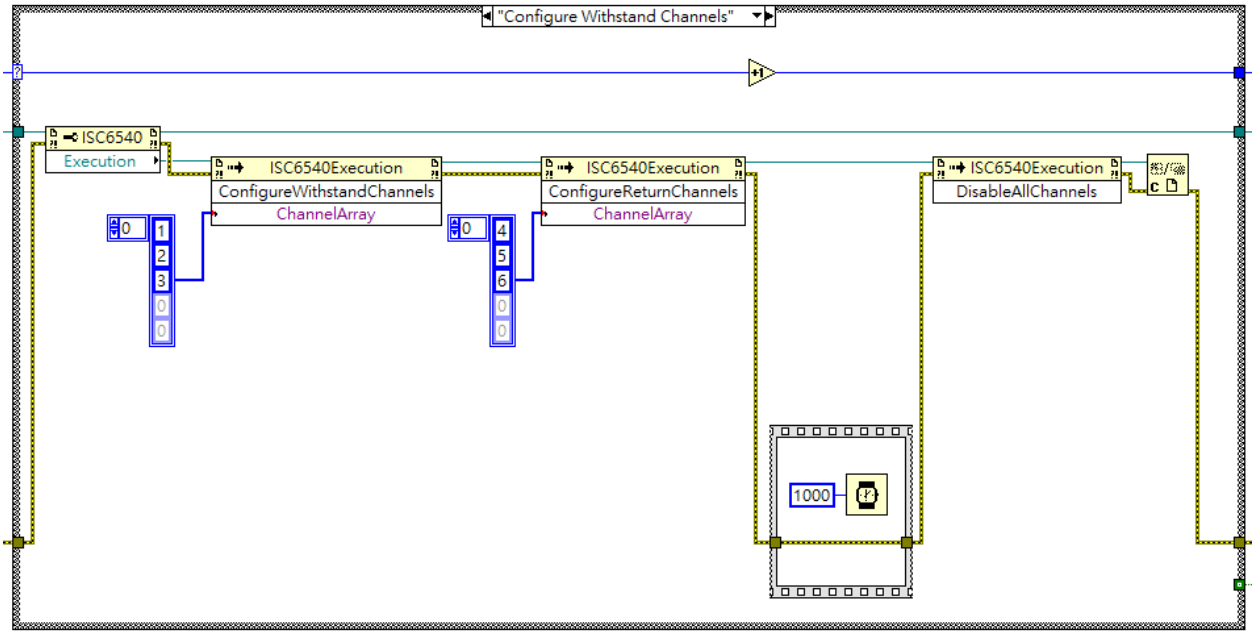
6.1 Create an ***Invoke Node*** function and connect the reference to the output of ***Automation Refnum*** and then click on the ***Method*** and select ***Initialize*** to initialize the connection with device.



For more detail for the parameters of the ***Initialize***() method, please refer to the help document, ***SC6540.chm*** located at "*<Program Files>\IVI Foundation\IVI\Drivers\SC6540*".
The first parameter ***ResourceName*** is a string type and indicates the interfaces type and address of the connection. The resource name,`"ASRL4::INSTR"`, represents a serial port with address 4. For example, a GPIB connection could be `"GPIB0::8::INSTR".` For TCP/IP connection, it will be in the format of `"TCPIP0::192.168.0.1::10001::SOCKET"`. The 10001 is the TCP/IP connection port of SC6540.
The ***OptionString***, `"Cache=false, InterchangeCheck=false, QueryInstrStatus=true, RangeCheck=false, RecordCoercions=false, Simulate=false"`, configures the options for the IVI driver with enabled ***QueryInstrStatus*** which will check the status at the end of each user operation. Please refer to the ***SC6540.chm*** for more details.

28105 N. Keith Drive
Lake Forest, IL 60045 USA
Toll Free: 1-800-858-TEST (8378) US/Canada
Phone: 1-847-367-4077 | Fax: 1-847-367-4080 | www.arisafety.com

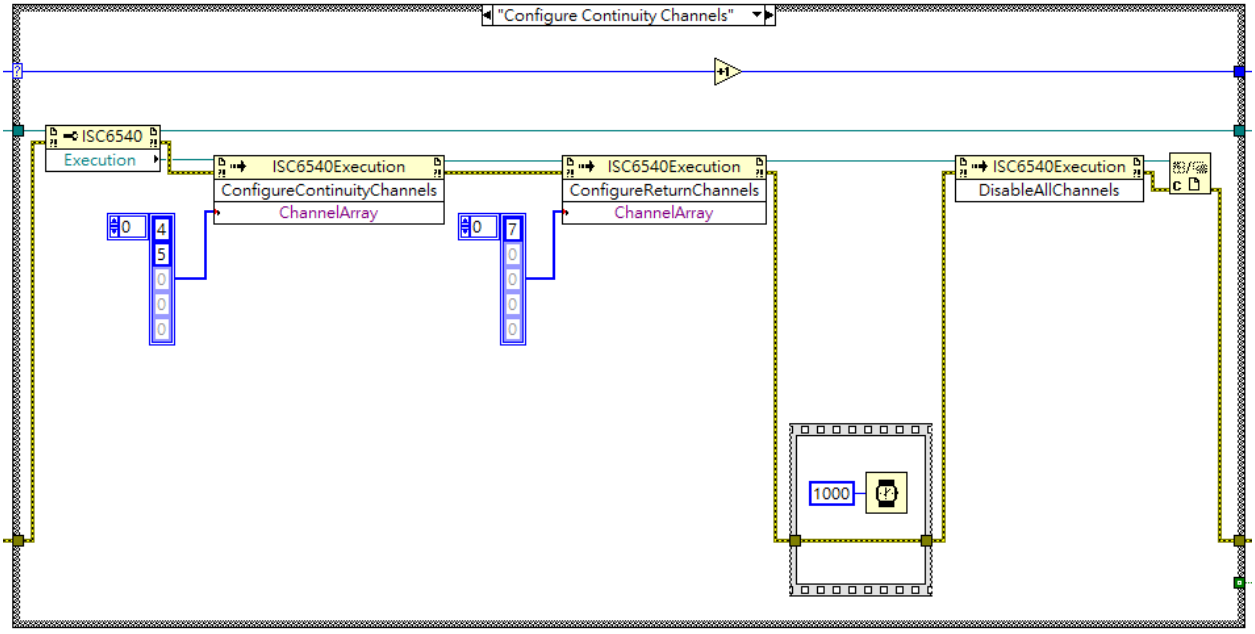
6.2 Configure Withstand Channels



For some units, we may need a multiplexer, SC6540, to extend the capacity on a HIPOT tester, such as Omnia2, HypotUltra, Hypot4 of Associated Research. Before a withstand test operated, the multiplexer needs to be configured to the test point.

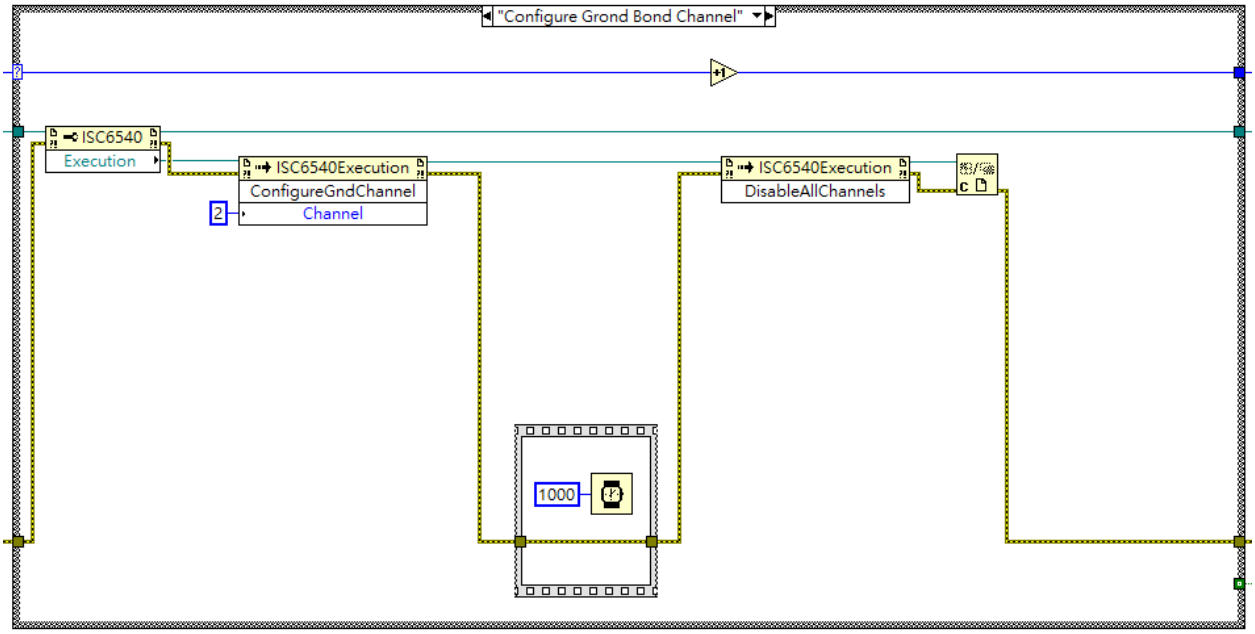
Both the ***ConfigureWithstandChannels()*** and ***ConfigureReturnChannels()*** method needs an integer array as input. After, the HV and Return channels are switched, we could add a procedure performing a withstand output. And disable the relays by ***DisableAllChannels()***.

6.3 Configure Continuity Channels



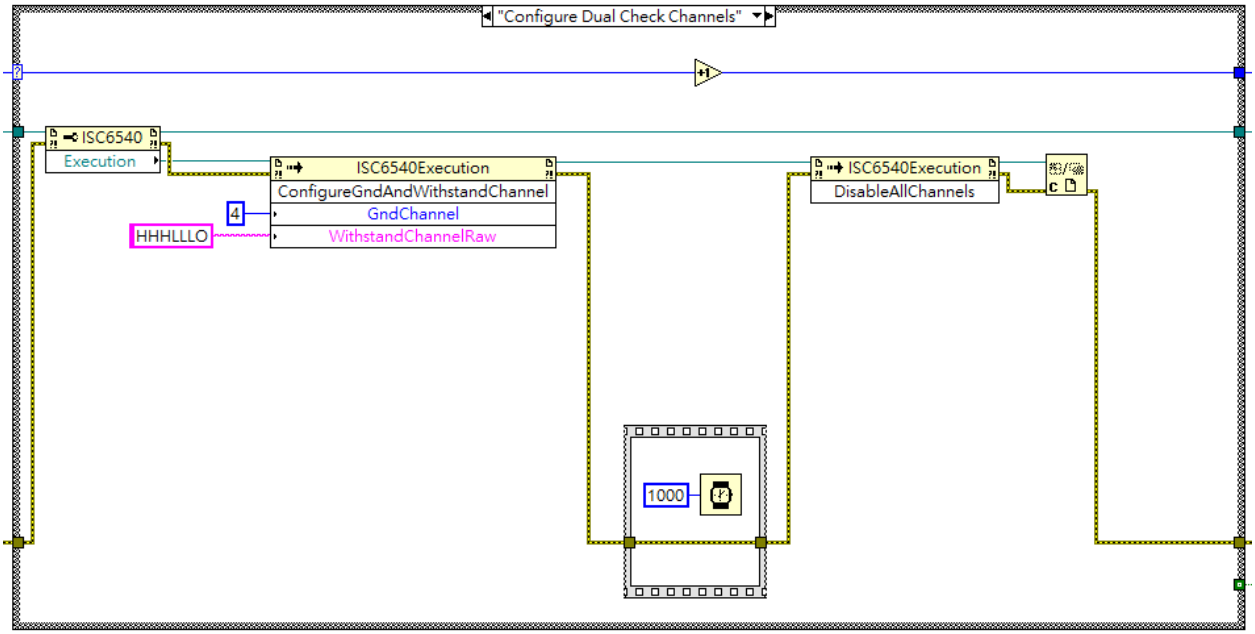
The ***ConfigureContinuityChannels()*** method takes an integer array as input. For the connection on the return circuits, we could invoke the method of ***ConfigureReturnChannels()*** which is shared with the withstand channels.

6.4 Configure Ground Bond Channel



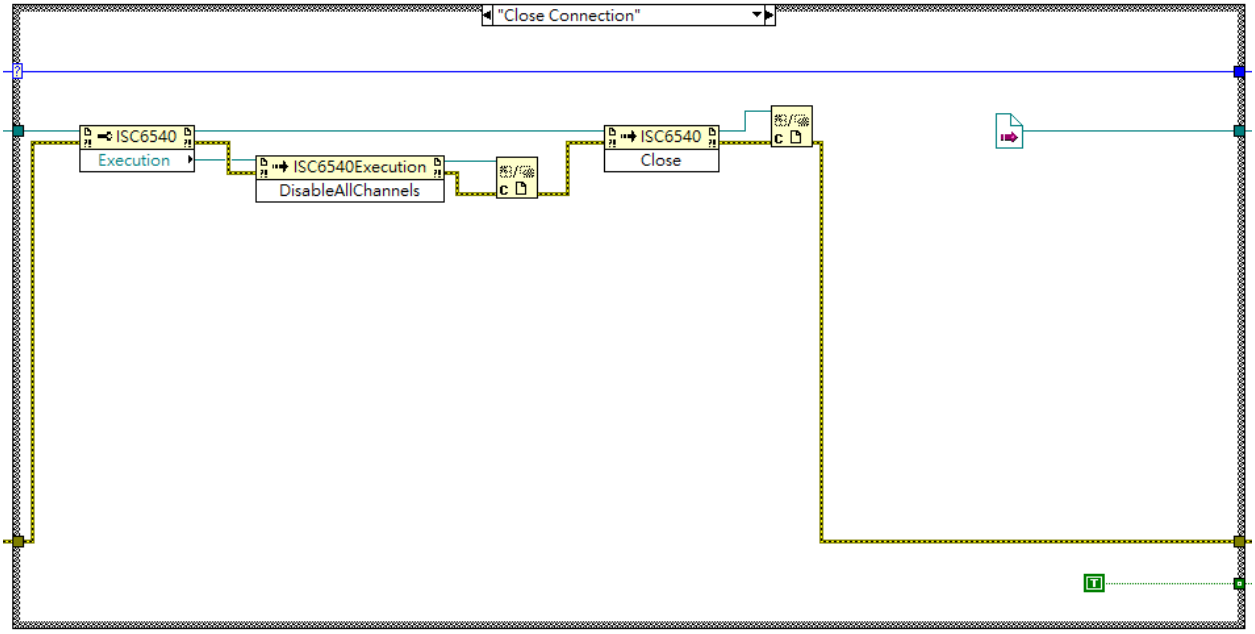
Only one ground bond channel could be enabled at one time. Therefore, the ***ConfigureGndChannel()*** takes a integer as input parameter at once. When a ground bond channel was set, the other switches would be disabled automatically.

6.5 Configure SC6540 for Dual Check



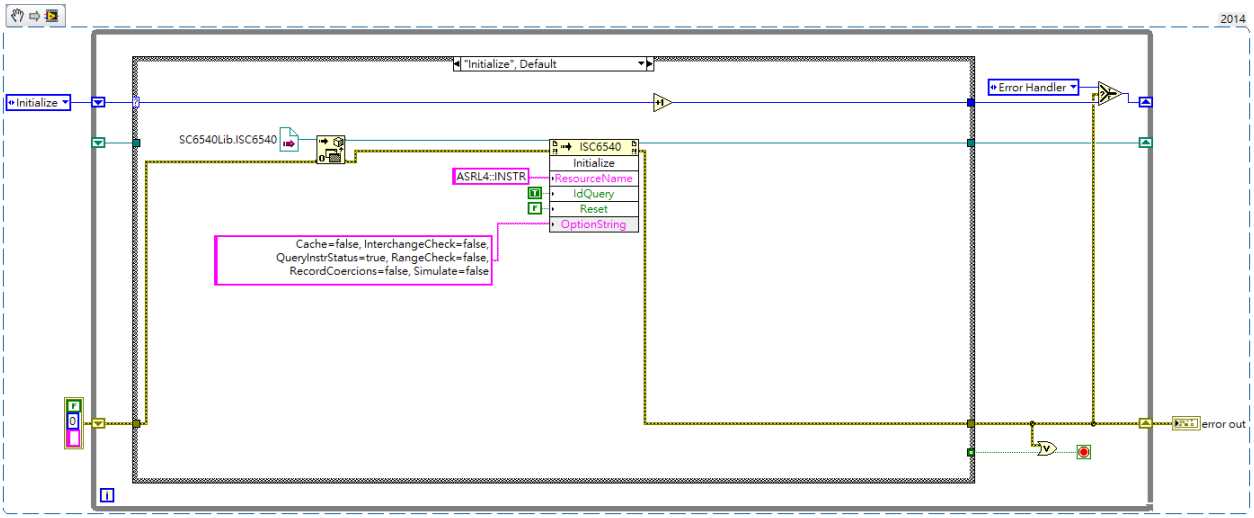
The DualCheck is the function allows the instrument to simultaneously run a hipot and AC Ground Bond test. By ***ConfigureGndAndWithstandChannel()***, we could configure both ground bond channel and withstand channels at the same time. This function takes one integer input as ground bond channel and one string as the raw configuration for HV module. The configurations will be H (High), L (Low) or O (Open). For example, the syntax for this command would be “HLOOOOL” which indicates the channel 1 is connected to HV and channel 2 and 7 would be connected to Return.

6.6 Close the session



Close() method in ***ISC6540*** class would close the I/O session to the instrument. Also, all of the references should be closed using the ***Close Reference*** function.

7 Completed example



28105 N. Keith Drive
Lake Forest, IL 60045 USA
Toll Free: 1-800-858-TEST (8378) US/Canada
Phone: 1-847-367-4077 | Fax: 1-847-367-4080 | www.arisafety.com

The completed example with the state machine design pattern for your reference which are located at the path of *<Program Files>\IVI Foundation\IVI\Drivers\SC6540\Examples*, including C#, C++ and Python as well. However, we suggest that using LabVIEW plug & play driver would be easier for LabVIEW developers. If you need a LabVIEW driver, please download it from the website of Associated Research or contact the vendor.

28105 N. Keith Drive
Lake Forest, IL 60045 USA
Toll Free: 1-800-858-TEST (8378) US/Canada
Phone: 1-847-367-4077 | Fax: 1-847-367-4080 | www.arisafety.com