Document Version 1.0

IVI Driver Getting Started Guide

For EEC400XAC series

Overview

This application note will describe the installing instructions and several programming examples for IVI Instrument Driver of EEC400XAC series. To understand more about the IVI drivers, please refer to the website of IVI Foundation. For more detail of the EEC400XAC IVI driver, please check the help document, EEC400XAC.chm, located at the path of *<Program Files>\IVI Foundation\IVI\Drivers\EEC400XAC*.

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.

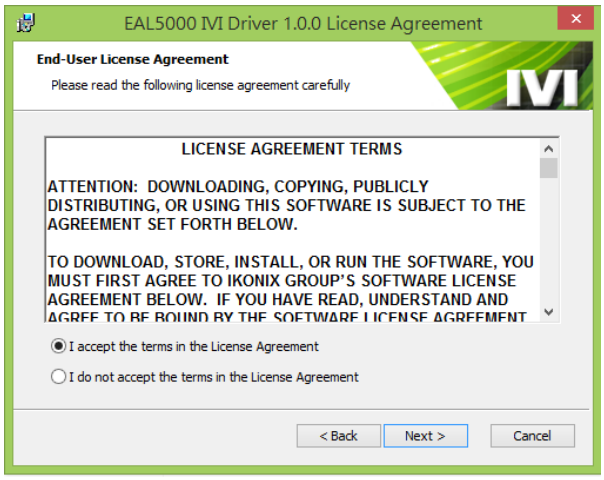
5. Getting Started with LabVIEW

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

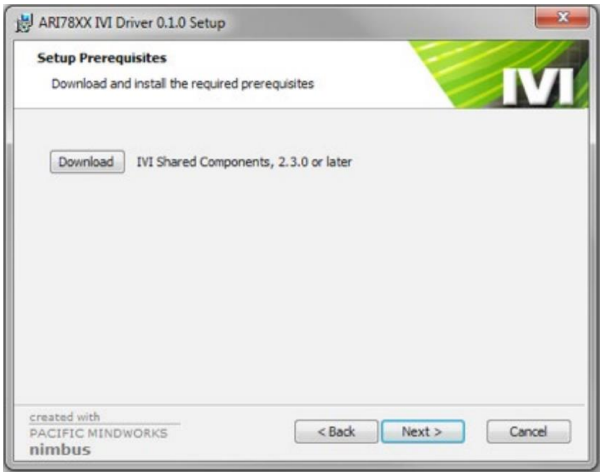
28105 N. Keith Drive
Lake Forest, IL 60045 USA
Toll Free: 1-800-858-8378 US/Canada
Phone: 1-847-367-4077 | Fax: 1-847-367-4080 | www.eecsources.com

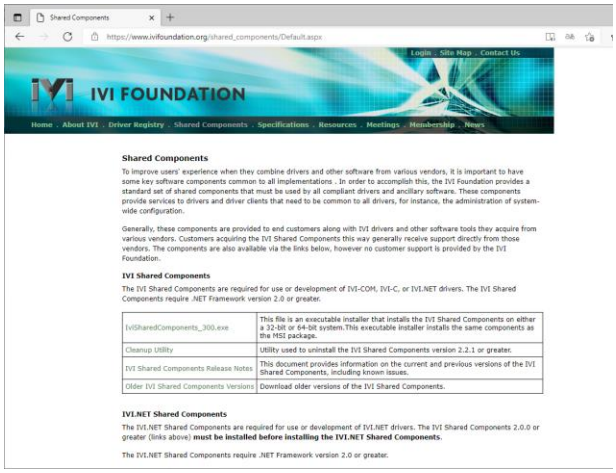
1. IVI Driver Setup

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 open.

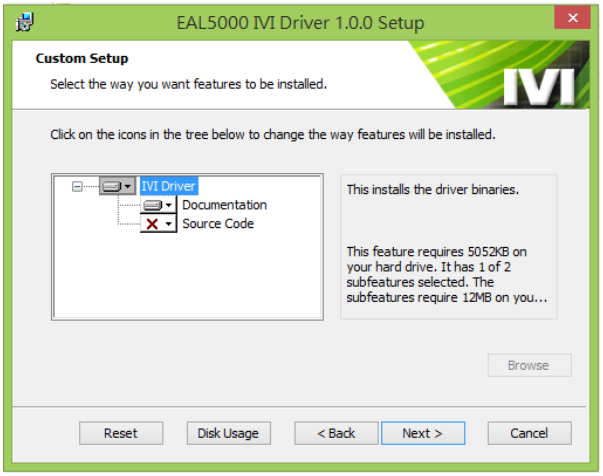


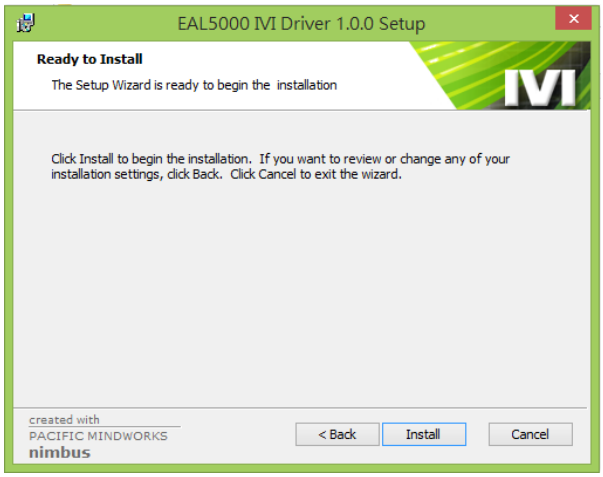


Please download the latest IVI Shared Components either 32-bit or 64-bit version. After downloading, install the shared components and continue the installation.

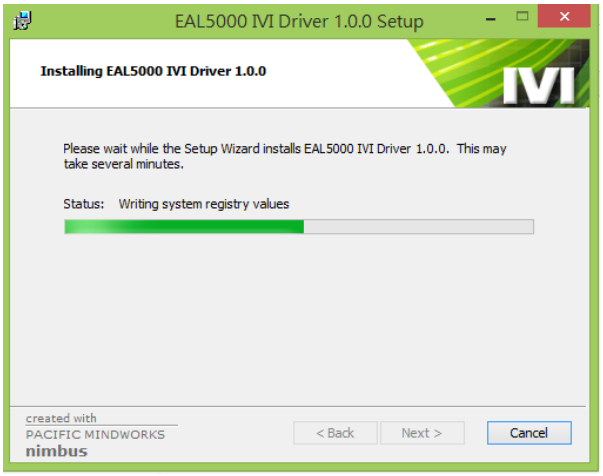
28105 N. Keith Drive
Lake Forest, IL 60045 USA
Toll Free: 1-800-858-8378 US/Canada
Phone: 1-847-367-4077 | Fax: 1-847-367-4080 | www.eecsources.com

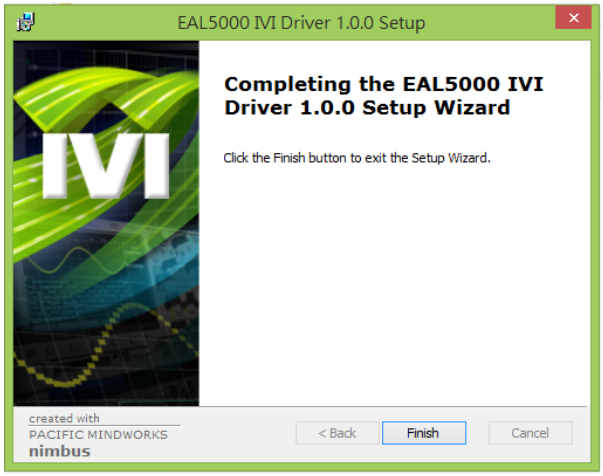
After the IVI Shared Components are installed, please follow the steps to complete installation.





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 of the *.dll file would be located in the "Bin" folder. And the necessary help documents will be in the folder of "..\Drivers\EEC400XAC".

28105 N. Keith Drive
Lake Forest, IL 60045 USA
Toll Free: 1-800-858-8378 US/Canada
Phone: 1-847-367-4077 | Fax: 1-847-367-4080 | www.eecsources.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 use IVI-C driver, either. However, we suggest that an IVI-COM interop would be easier for you to develop the program.

Requirements

- EEC400XAC IVI Driver
- IVI Shared Components, *https://www.ivifoundation.org/shared_components/Default.aspx*
- Microsoft Visual Studio or other IDEs
- An EEC400XAC series power supply, including 430XAC, 460XAC

Download the Drivers

Please go to the website of the IKONIX 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 EEC400XAC IVI Driver. A help file, ***EEC400XAC.chm***, would be located at the path of *<Program Files>\IVI Foundation\IVI\Drivers\EEC400XAC*. 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\EEC400XAC\Examples*, including C#, C++, Python and LabVIEW as well.

Development

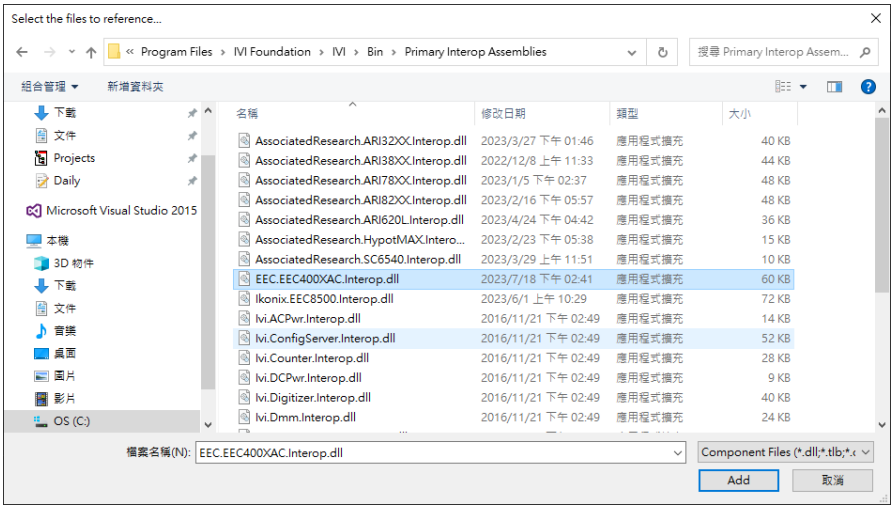
1 Create a C# project

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 ***EEC.EEC400XAC.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 EEC.EEC400XAC.Interop;
```

3 Start programming

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

```
var driver = new EEC400XAC();
driver.Initialize("ASRL3::INSTR", true, false, "QueryInstrStatus=true");
```

For more detail for the parameters of the ***Initialize()*** method, please refer to the help document, ***EEC400XAC.chm***, which is located at "*<Program Files>\IVI Foundation\IVI\Drivers\EEC400XAC*".

The first parameter ***ResourceName*** is a string type and indicates the interfaces type and address of the connection. The resource name,`"ASRL3::INSTR"`, represents a serial port with address 3. 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 EEC400XAC.

There are other parameters for the option of the ***Initialize()*** method, please refer to the ***EEC400XAC.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: 1-800-858-8378 US/Canada
Phone: 1-847-367-4077 | Fax: 1-847-367-4080 | www.eecsources.com

3.2 Create file and setup test

```
// Edit Memory in Manual mode, AC, 3phase-4wire
Console.WriteLine("Configuring Manual Mode, AC Output, 3 phases / 4wires...");
driver.System.Mode = EEC400XACModeEnum.EEC400XACModeManual;
driver.System.OutputMode = EEC400XACOutputModeEnum.EEC400XACOutputModeAC;
driver.System.Function = EEC400XACFunctionEnum.EEC400XACFunctionThreePhase4Wire;

driver.Steps.ActiveMemory = 1;
driver.Parameters.Range = EEC400XACRangeEnum.EEC400XACRangeAuto;
driver.Parameters.Voltage = 110;
driver.Parameters.Frequency = 60;
driver.Parameters.PhaseSet = EEC400XACPhaseSetEnum.EEC400XACPhaseSetA;
driver.Parameters.CurrentHighLimit = 3.2;
driver.Parameters.PhaseSet = EEC400XACPhaseSetEnum.EEC400XACPhaseSetB;
driver.Parameters.CurrentHighLimit = 2.5;
driver.Parameters.PhaseSet = EEC400XACPhaseSetEnum.EEC400XACPhaseSetC;
driver.Parameters.CurrentHighLimit = 3.0;
```

For the EEC400XAC, all of the test parameters would be within a memory. Therefore, you need to select a memory first and then setup the parameters. Also, the parameters may differ depending on the output mode and functions.

3.3 Load file and start a test

```
// Output and Measure
//
Console.WriteLine("Start Output...");
driver.Steps.ActiveMemory = 1;
driver.Execution.RunTest();
```

Before running output, you have to select a memory to load. And then invoke the method of ***`driver.Execution.RunTest()`*** to start a test.

3.4 Measure during test

```
int memory = 0;
int step = 0;
string status = null;
double frequency = 0;
double voltage = 0;
double current = 0;
double power = 0;
double currentPeak = 0;
double powerFactor = 0;
double reactivePower = 0;
double crestFactor = 0;
double apparentPower = 0;
double timer = 0;

for (int i = 0; i < 3; i++)
{
    driver.Display.ThreePhase4Wire.PhaseA.ReadDisplay(ref memory,
                                                      ref step,
                                                      ref status,
                                                      ref frequency,
                                                      ref voltage,
                                                      ref current,
                                                      ref power,
                                                      ref currentPeak,
                                                      ref powerFactor,
                                                      ref reactivePower,
                                                      ref crestFactor,
                                                      ref apparentPower,
```

```
                                                ref timer);
    Console.WriteLine($"PHASE-A\nMemory-{memory}, Step-{step}, Status-{status}\n"
                      + $"Frequency:{frequency}\n"
                      + $"Voltage:{voltage}\n"
                      + $"Current:{current}\n"
                      + $"Power:{power}\n"
                      + $"Peak Current:{currentPeak}\n"
                      + $"Power Factor:{powerFactor}\n"
                      + $"Reactive Power:{reactivePower}\n"
                      + $"Crest Factor:{crestFactor}\n"
                      + $"Apparent Power: {apparentPower}\n"
                      + $"Timer:{timer}\n");

    driver.Display.ThreePhase4Wire.PhaseB.ReadDisplay(ref memory,
                                                ref step,
                                                ref status,
                                                ref frequency,
                                                ref voltage,
                                                ref current,
                                                ref power,
                                                ref currentPeak,
                                                ref powerFactor,
                                                ref reactivePower,
                                                ref crestFactor,
                                                ref apparentPower,
                                                ref timer);
    Console.WriteLine($"PHASE-B\nMemory-{memory}, Step-{step}, Status-{status}\n"
                      + $"Frequency:{frequency}\n"
                      + $"Voltage:{voltage}\n"
                      + $"Current:{current}\n"
                      + $"Power:{power}\n"
                      + $"Peak Current:{currentPeak}\n"
                      + $"Power Factor:{powerFactor}\n"
                      + $"Reactive Power:{reactivePower}\n"
                      + $"Crest Factor:{crestFactor}\n"
                      + $"Apparent Power: {apparentPower}\n"
                      + $"Timer:{timer}\n");

    driver.Display.ThreePhase4Wire.PhaseC.ReadDisplay(ref memory,
                                                ref step,
                                                ref status,
                                                ref frequency,
                                                ref voltage,
                                                ref current,
                                                ref power,
                                                ref currentPeak,
                                                ref powerFactor,
                                                ref reactivePower,
                                                ref crestFactor,
                                                ref apparentPower,
                                                ref timer);
    Console.WriteLine($"PHASE-C\nMemory-{memory}, Step-{step}, Status-{status}\n"
                      + $"Frequency:{frequency}\n"
                      + $"Voltage:{voltage}\n"
                      + $"Current:{current}\n"
                      + $"Power:{power}\n"
                      + $"Peak Current:{currentPeak}\n"
                      + $"Power Factor:{powerFactor}\n"
                      + $"Reactive Power:{reactivePower}\n"
                      + $"Crest Factor:{crestFactor}\n"
                      + $"Apparent Power: {apparentPower}\n"
                      + $"Timer:{timer}\n");

    driver.Display.ThreePhase4Wire.SumPhase.ReadDisplay(ref memory,
                                                ref step,
                                                ref status,
                                                ref frequency,
                                                ref voltage,
                                                ref current,
```

```
                                                    ref power,
                                                    ref powerFactor,
                                                    ref reactivePower,
                                                    ref apparentPower,
                                                    ref timer);
    Console.WriteLine($"PHASE-Sum\nMemory-{memory}, Step-{step}, Status-{status}\n"
                        + $"Frequency:{frequency}\n"
                        + $"Voltage:{voltage}\n"
                        + $"Current:{current}\n"
                        + $"Power:{power}\n"
                        + $"Power Factor:{powerFactor}\n"
                        + $"Reactive Power:{reactivePower}\n"
                        + $"Apparent Power: {apparentPower}\n"
                        + $"Timer:{timer}\n");
    Thread.Sleep(500);
}
```

This while loop would run with the condition of state is testing. Using the methods of Measure subsystem could let you read the immediate readings.

3.5 Close the session

```
driver.Execution.AbortTest();
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 find at the path of "*<Program Files>\IVI Foundation\IVI\Drivers\EEC400XAC\Examples*". Also, there is another section describing an example of program mode with 1 phase and 3 wires configurations.

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

- EEC400XAC IVI Driver
- IVI Shared Components, *https://www.ivifoundation.org/shared_components/Default.aspx*
- Microsoft Visual Studio or other IDEs
- An EEC400XAC series power supply, including 430XAC, 460XAC

Download the Drivers

Please go to the website of the IKONIX 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 EEC400XAC IVI Driver. A help file, ***EEC400XAC.chm***, would be located at the path of *<Program Files>\IVI Foundation\IVI\Drivers\EEC400XAC*. 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\EEC400XAC\Examples*, including C#, C++ and Python as well.

Development

1 Create a C++ project

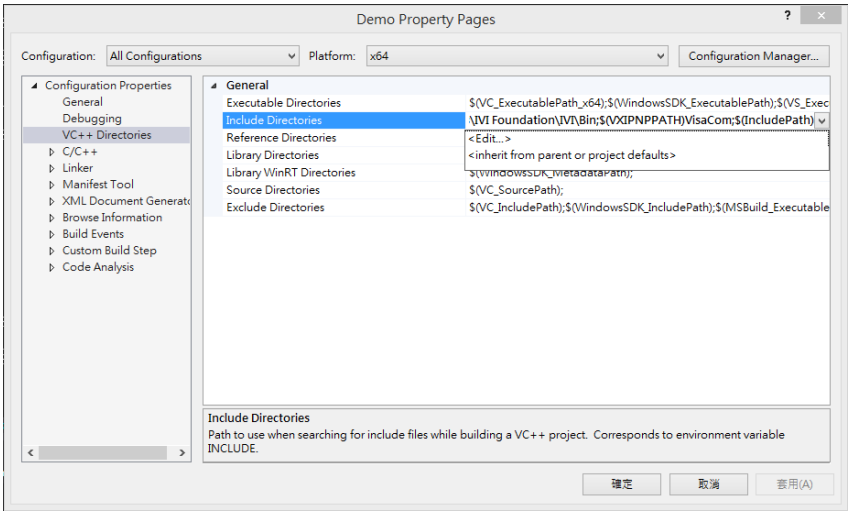
1.1 Open Visual Studio IDE and create a new C++ console project.

2 Include Directories

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

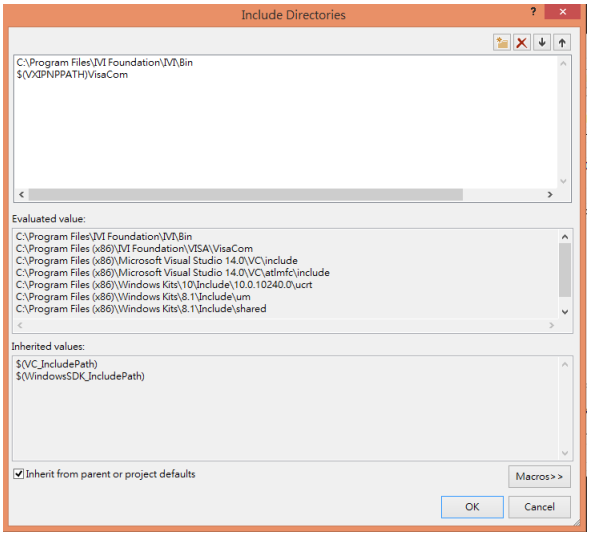
2.2 Expand the ***Configuration Properties*** and select ***VC++ Directories*** on the left menu.

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***



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

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

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

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

```
#include "stdafx.h"
#include <iostream>
#import <IviDriverTypeLib.dll> no_namespace
#import <GlobMgr.dll> no_namespace
#import <EEC400XAC_64.dll> no_namespace
#include <windows.h>
```

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);
IEEC400XACPtr driver(__uuidof(EEC400XAC));

// IIviDriverIdentity properties - Initialize required
//
driver->Initialize("ASRL3::INSTR", true, false, "QueryInstrStatus=true");
```

For more detail for the parameters of the ***Initialize***() method, please refer to the help document, ***EEC400XAC.chm*** located at "*<Program Files>\IVI Foundation\IVI\Drivers\EEC400XAC*".

The first parameter ***ResourceName*** is a string type and indicates the interfaces type and address of the connection. The resource name,`"ASRL3::INSTR"`, represents a serial port with address 3. 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 EEC400XAC.

There are other parameters for the option of the ***Initialize()*** method, please refer to the ***EEC400XAC.chm*** for more detail. For example, `"QueryInstrStatus=true"` makes the session automatically query the error status for each command was sent.

3.2 Create file and setup test

```
// Edit Memory in Manual mode, AC, 3phase-4wire
std::wcout << "Configuring Manual Mode, AC Output, 3 phases / 4wires..." << std::endl;
driver -> System -> Mode = EEC400XACModeEnum::EEC400XACModeManual;
driver -> System-> OutputMode = EEC400XACOutputModeEnum::EEC400XACOutputModeAC;
driver -> System -> Function = EEC400XACFunctionEnum::EEC400XACFunctionThreePhase4Wire;

driver -> Steps -> ActiveMemory = 1;
driver -> Parameters -> Range = EEC400XACRangeEnum::EEC400XACRangeAuto;
driver -> Parameters -> Voltage = 110;
driver -> Parameters -> Frequency = 60;
driver -> Parameters -> PhaseSet = EEC400XACPhaseSetEnum::EEC400XACPhaseSetA;
driver -> Parameters -> CurrentHighLimit = 3.2;
driver -> Parameters -> PhaseSet = EEC400XACPhaseSetEnum::EEC400XACPhaseSetB;
driver -> Parameters -> CurrentHighLimit = 2.5;
driver -> Parameters -> PhaseSet = EEC400XACPhaseSetEnum::EEC400XACPhaseSetC;
driver -> Parameters -> CurrentHighLimit = 3.0;
```

For the EEC400XAC, all of the test parameters would be within a memory. Therefore, you need to select a memory to be edited. Also, the parameters may differ depending on the output mode and functions.

3.3 Load file and start a test

```
// Output and Measure
//
std::wcout << "Start Output..." << std::endl;
driver -> Steps -> ActiveMemory = 1;
driver -> Execution -> RunTest();
```

Before running output, you have to select a memory to load. And then invoke the method of ***driver->Execution->RunTest()*** to start a test.

3.4 Measure during test

```
long memory = 0;
long step = 0;
BSTR status = NULL;
BSTR *pStatus = &status;
double frequency = 0;
double voltage = 0;
double current = 0;
double power = 0;
double currentPeak = 0;
double powerFactor = 0;
double reactivePower = 0;
double crestFactor = 0;
double apparentPower = 0;
double timer = 0;

for (int i = 0; i < 3; i++)
{
        driver -> Display -> ThreePhase4Wire -> PhaseA -> ReadDisplay(&memory,
                                                                      &step,
                                                                      pStatus,
                                                                      &frequency,
                                                                      &voltage,
                                                                      &current,
                                                                      &power,
                                                                      &currentPeak,
                                                                      &powerFactor,
                                                                      &reactivePower,
                                                                      &crestFactor,
                                                                      &apparentPower,
                                                                      &timer);
        std::wcout << "PHASE-A" << std::endl << "Memory:" <<memory << "\t" << " Step:"
<<step << " Status:" << *pStatus << std::endl
                           <<"Frequency: "<<frequency<< std::endl
                           <<"Voltage: "<<voltage<< std::endl
                           << "Current: "<<current<< std::endl
                           <<"Power: "<<power<< std::endl
                           <<"Peak Current: "<<currentPeak<< std::endl
                           <<"Power Factor: "<<powerFactor<< std::endl
                           <<"Reactive Power: "<<reactivePower<< std::endl
                           <<"Crest Factor: "<<crestFactor<< std::endl
                           <<"Apparent Power: "<<apparentPower<< std::endl
                           <<"Timer: "<<timer<< std::endl << std::endl;
        driver -> Display -> ThreePhase4Wire -> PhaseB -> ReadDisplay(&memory,
                                                                      &step,
                                                                      pStatus,
                                                                      &frequency,
                                                                      &voltage,
                                                                      &current,
                                                                      &power,
                                                                      &currentPeak,
                                                                      &powerFactor,
                                                                      &reactivePower,
                                                                      &crestFactor,
                                                                      &apparentPower,
```

```
                                                        &timer);
        std::wcout << "PHASE-B" << std::endl << "Memory: " <<memory  << "\t" << "Step: "
<<step << "\t" << "Status: " <<status<< std::endl
                        <<"Frequency: "<<frequency<< std::endl
                        <<"Voltage: "<<voltage<< std::endl
                        <<"Current: "<<current<< std::endl
                        <<"Power: "<<power<< std::endl
                        <<"Peak Current: "<<currentPeak<< std::endl
                        <<"Power Factor: "<<powerFactor<< std::endl
                        <<"Reactive Power: "<<reactivePower<< std::endl
                        <<"Crest Factor: "<<crestFactor<< std::endl
                        <<"Apparent Power: "<<apparentPower<< std::endl
                        <<"Timer: "<<timer<< std::endl << std::endl;
        driver -> Display -> ThreePhase4Wire -> PhaseC -> ReadDisplay(&memory,
                                                        &step,
                                                        pStatus,
                                                        &frequency,
                                                        &voltage,
                                                        &current,
                                                        &power,
                                                        &currentPeak,
                                                        &powerFactor,
                                                        &reactivePower,
                                                        &crestFactor,
                                                        &apparentPower,
                                                        &timer);
        std::wcout << "PHASE-C" << std::endl << "Memory: " <<memory  << "\t" << "Step: "
<<step << "\t" << "Status: " <<status<< std::endl
                        <<"Frequency: "<<frequency<< std::endl
                        <<"Voltage: "<<voltage<< std::endl
                        <<"Current: "<<current<< std::endl
                        <<"Power: "<<power<< std::endl
                        <<"Peak Current: "<<currentPeak<< std::endl
                        <<"Power Factor: "<<powerFactor<< std::endl
                        <<"Reactive Power: "<<reactivePower<< std::endl
                        <<"Crest Factor: "<<crestFactor<< std::endl
                        <<"Apparent Power: "<<apparentPower<< std::endl
                        <<"Timer: "<<timer<< std::endl << std::endl;
        driver -> Display -> ThreePhase4Wire -> SumPhase -> ReadDisplay(&memory,
                                                        &step,
                                                        pStatus,
                                                        &frequency,
                                                        &voltage,
                                                        &current,
                                                        &power,
                                                        &powerFactor,
                                                        &reactivePower,
                                                        &apparentPower,
                                                        &timer);
        std::wcout << "PHASE-Sum" << std::endl << "Memory: " <<memory  << "\t" << "Step: "
<<step << "\t" << "Status: " <<status<< std::endl
                        <<"Frequency: "<<frequency<< std::endl
                        <<"Voltage: "<<voltage<< std::endl
                        <<"Current: "<<current<< std::endl
                        <<"Power: "<<power<< std::endl
                        <<"Power Factor: "<<powerFactor<< std::endl
                        <<"Reactive Power: "<<reactivePower<< std::endl
                        <<"Apparent Power: "<<apparentPower<< std::endl
                        <<"Timer: "<<timer<< std::endl << std::endl;
                        Sleep(500);
}
```

This while loop would run with polling the states and meters. Using the methods of Measure subsystem could let you read the immediate readings.

3.5 Close the session

```
//Close connection
std::wcout << "End of Output." << std::endl << std::endl;
driver -> Execution -> AbortTest();
driver -> Close();
std::wcout << "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\EEC400XAC\Examples*".

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

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

- EEC400XAC IVI Driver
- IVI Shared Components, *https://www.ivifoundation.org/shared_components/Default.aspx*
- Python IDE
- Cometypes Library (pip install cometypes)
- An EEC400XAC series power supply, including 430XAC, 460XAC

Download the Drivers

Please go to the website of the IKONIX 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 EEC400XAC IVI Driver. A help file, ***EEC400XAC.chm***, would be located at the path of *<Program Files>\IVI Foundation\IVI\Drivers\EEC400XAC*. 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\EEC400XAC\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 EEC400XAC_64.dll

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

cc.GetModule('EEC400XAC_64.dll')
from comtypes.gen import EEC400XACLib
```

4 Start programming

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

```
driver = cc.CreateObject('EEC400XAC.EEC400XAC', interface=EEC400XACLib.IEEC400XAC)

#   Initialize Driver and make connection
driver.Initialize('ASRL3::INSTR', True, False, 'QueryInstrStatus=true')
```

For more detail for the parameters of the ***Initialize***() method, please refer to the help document, ***EEC400XAC.chm*** located at "*<Program Files>\IVI Foundation\IVI\Drivers\EEC400XAC*".

The first parameter ResourceName is a string type and indicates the interfaces type and address of the connection. The resource name,`"ASRL3::INSTR"`, represents a serial port with address 3. 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 EEC400XAC.

There are other parameters for the option of the ***Initialize()*** method, please refer to the ***EEC400XAC.chm*** for more detail. For example, `"QueryInstrStatus=true"` makes the session automatically query the error status for each command was sent.

4.2 Create file and setup test

```
# Edit Memory in Manual mode, AC, 3phase-4wire
print("Configuring Manual Mode, AC Output, 3 phases / 4wires...")
driver.System.Mode = EEC400XACLib.EEC400XACModeManual
driver.System.OutputMode = EEC400XACLib.EEC400XACOutputModeAC
driver.System.Function = EEC400XACLib.EEC400XACFunctionThreePhase4Wire

driver.Steps.ActiveMemory = 1
driver.Parameters.Range = EEC400XACLib.EEC400XACRangeAuto
driver.Parameters.Voltage = 110
driver.Parameters.Frequency = 60
driver.Parameters.PhaseSet = EEC400XACLib.EEC400XACPhaseSetA
driver.Parameters.CurrentHighLimit = 3.2
```

```
driver.Parameters.PhaseSet = EEC400XACLib.EEC400XACPhaseSetB
driver.Parameters.CurrentHighLimit = 2.5
driver.Parameters.PhaseSet = EEC400XACLib.EEC400XACPhaseSetC
driver.Parameters.CurrentHighLimit = 3.0
```

For the EEC400XAC, all of the test parameters would be within a memory. Therefore, you need to select a memory to be edited. Also, the parameters may differ depending on the output mode and functions.

4.3 Load file and start a test

```
# Output and Measure
#
print("Start Output...")
driver.Steps.ActiveMemory = 1
driver.Execution.RunTest()
```

Before running output, you have to select a memory to load. And then invoke the method of ***driver.Execution.RunTest()*** to start a test.

4.4 Measure during test

```
for i in range(3):
    MeasurePhaseA = driver.Display.ThreePhase4Wire.PhaseA.ReadDisplay()
    print('Phase-A')
    print(MeasurePhaseA)
    MeasurePhaseB = driver.Display.ThreePhase4Wire.PhaseB.ReadDisplay()
    print('Phase-B')
    print(MeasurePhaseB)
    MeasurePhaseC = driver.Display.ThreePhase4Wire.PhaseC.ReadDisplay()
    print('Phase-C')
    print(MeasurePhaseC)
    MeasurePhaseSum = driver.Display.ThreePhase4Wire.SumPhase.ReadDisplay()
    print('Phase-Sum')
    print(MeasurePhaseSum)
    time.sleep(0.5)
```

This for loop would run with polling the state and meters. Using the methods of Measure subsystem could let you read the immediate readings.

4.5 Close the session

```
# Close connection
driver.Execution.AbortTest()
print("End of Output.")
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\EEC400XAC\Examples*".

28105 N. Keith Drive
Lake Forest, IL 60045 USA
Toll Free: 1-800-858-8378 US/Canada
Phone: 1-847-367-4077 | Fax: 1-847-367-4080 | www.eecsources.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 plu&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

- EEC400XAC IVI Driver
- IVI Shared Components, *https://www.ivifoundation.org/shared_components/Default.aspx*
- National Instruments LabVIEW (This example was written in LabVIEW 2014)
- An EEC400XAC series power supply, including 430XAC, 460XAC

Download the Drivers

Please go to the website of the IKONIX 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.

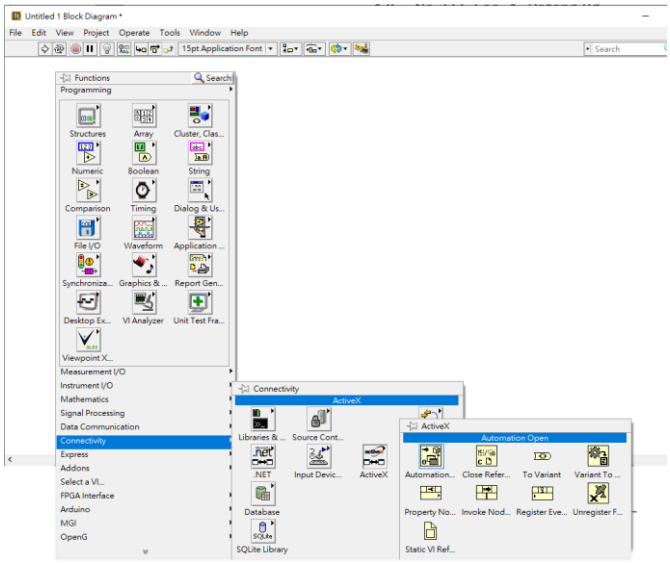
In the installed directory, there are several documents for your reference understanding the EEC400XAC IVI Driver. A help file, ***EEC400XAC.chm***, would be located at the path of *<Program Files>\IVI Foundation\IVI\Drivers\EEC400XAC*. In this help file, you could find all of the provided functions 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\EEC400XAC\Examples*, including C#, C++ and Python as well.

Development

1 Open a new vi.

2 Import the DLL component.

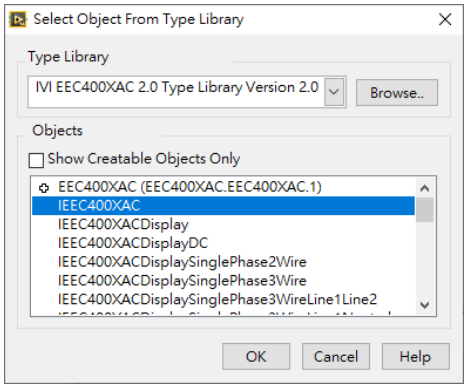


Open the ***Function Palette*** by right-clicking on the block diagram. Then select ***Connectivity*** -> ***ActiveX***. Select or drop the ***Automation Open*** function on the block diagram.

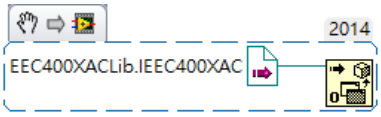
3 Right-clicking on the ***Automation Open*** and select ***Select ActiveX Class*** -> ***Browse*** will open a window for choosing the DLL.

4 Select the Browse button and select the file ***EEC400XAC.dll*** located at *<Program Files> (x86)\IVI Foundation\IVI\Bin.* The ***IVI EEC400XAC Type Library*** would be added into the ***Type Libraries*** drop down menu.

5 Select ***IEEC400XAC*** and then click ***OK*** to complete creating an object of EEC400XAC driver instance.

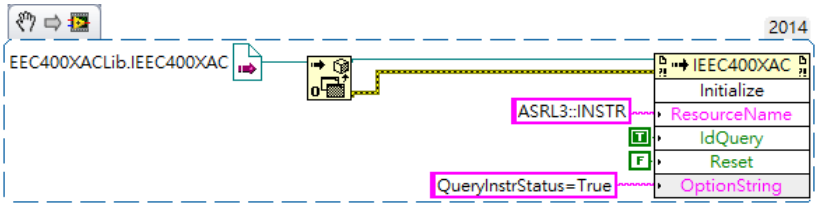


The Labview will automatically generate an ***Automation refnum*** of ***EEC400XACLib.IEEC400XAC*** control and connect to the Automation Open function.



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

6 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.

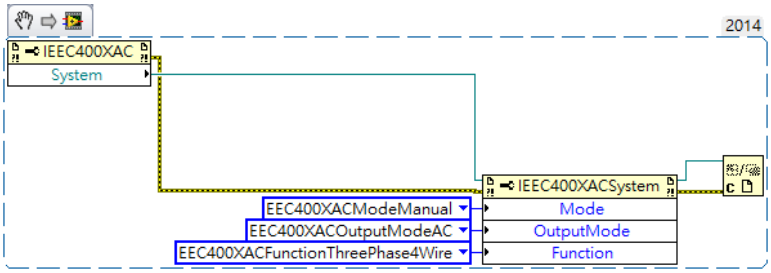


For more detail for the parameters of the ***Initialize***() method, please refer to the help document, ***EEC400XAC.chm*** located at "*<Program Files>\IVI Foundation\IVI\Drivers\EEC400XAC*".

The first parameter ***ResourceName*** is a string type and indicates the interfaces type and address of the connection. The resource name,`"ASRL3::INSTR"`, represents a serial port with address 3. 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 EEC400XAC.

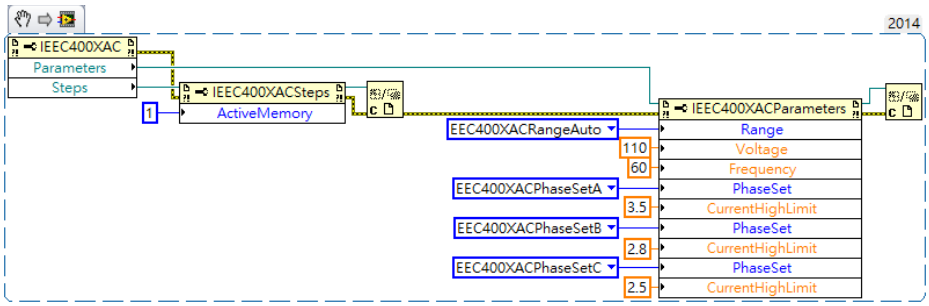
There are other parameters for the option of the Initialize() method, please refer to the ***EEC400XAC.chm*** for more detail. For example, `"QueryInstrStatus=true"` makes the session automatically query the error status for each command was sent.

6.1 Switch operation mode



Before we configure the parameters on the EEC400XAC power supply, we have to switch the operation mode. On the 400XAC, it could set to 1phase-2wires, 1phase-3wires and 3phases-4wires. Also, the 400XAC capable of offering AC or DC power source. For the control method, it is capable of switching to Manual mode, Program mode and IEC61000-4-11 procedure.

6.2 Select a memory and edit parameters

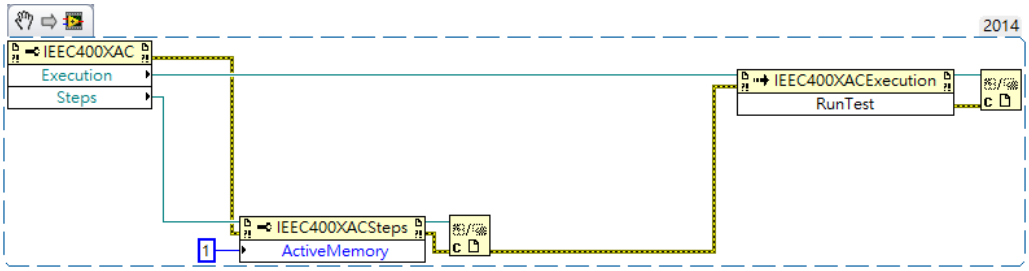


Use the ***Property Node*** to get reference of the sub-system of ***IEEC400XAC*** class. For example, in order to switch the active memory which is a property of the ***IEEC400XAC.Steps***, so we could put a property node to access the ***IEEC400XAC.Steps.ActiveMemory***. Also. we could edit the parameters with the same concepts. There are different parameters need to be setup depending on the control modes, output modes and functions.

For the EEC400XAC, all of the test parameters would be within a memory. Therefore, you need to select a memory to be edited.

Please be noted that the flow of error data could make sure that the procedure ran sequentially.

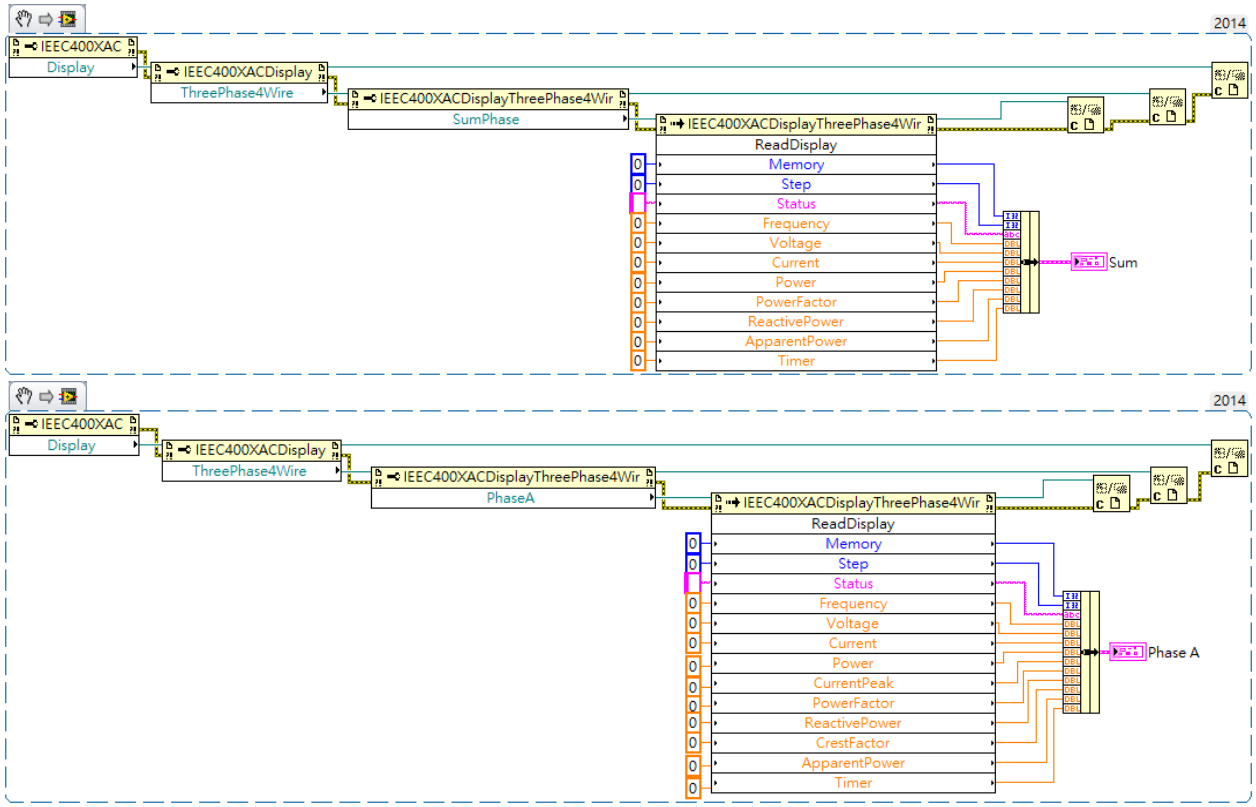
6.3 Load file and start a test



Before running output, you have to select a memory to be load. And then invoke ***`driver.Execution.RunTest()`*** method to start a test.

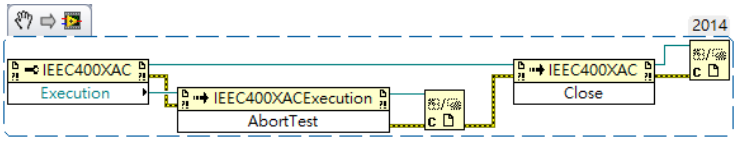
28105 N. Keith Drive
Lake Forest, IL 60045 USA
Toll Free: 1-800-858-8378 US/Canada
Phone: 1-847-367-4077 | Fax: 1-847-367-4080 | www.eecsources.com

6.4 Measure during test



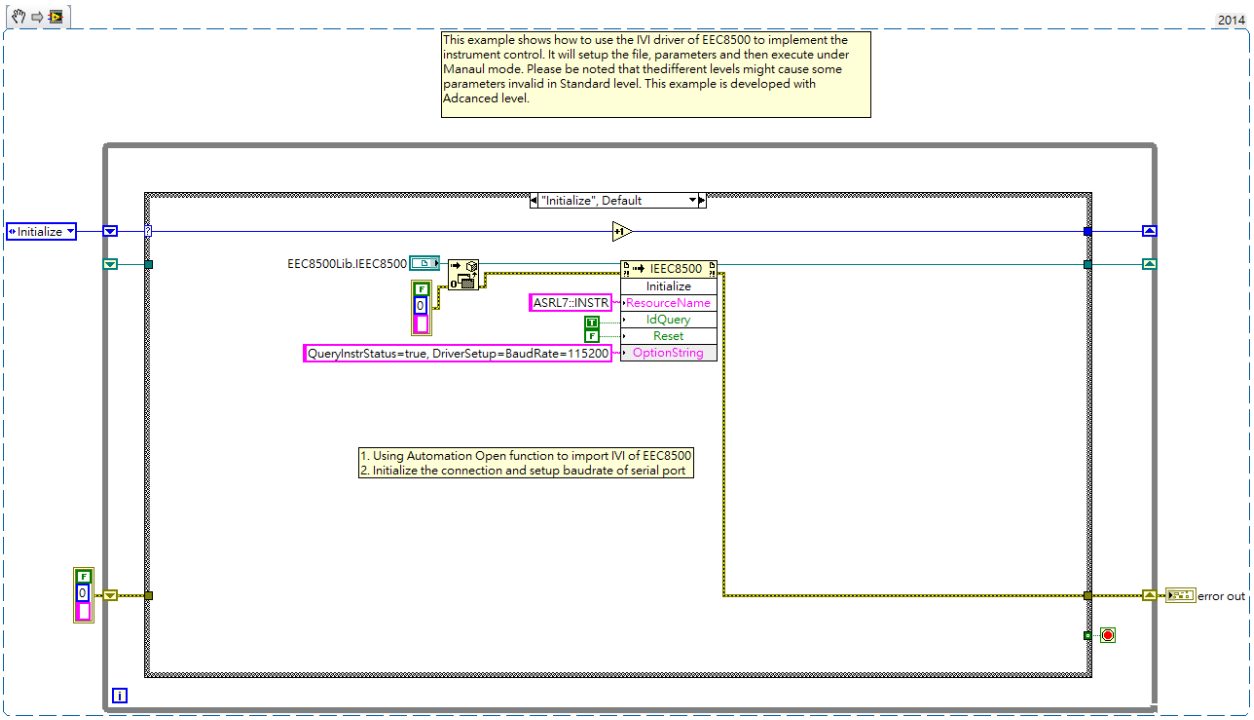
We could make a loop polling the state and meters. For the different phases, there are corresponding commands to read them. Using the methods of Display subsystem could let you get the immediate readings.

6.5 Stop and close the session



The above procedure shows how to abort the 400XAC output and close the connection. ***Close*** method in ***IEEC400XAC*** 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



The completed example for your reference which are located at the path of *<Program Files>\IVI Foundation\IVI\Drivers\EEC400XAC\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 IKONIX or contact the vendor.