CANoe ASAM XIL API Support

Version 1.1
2018-06-12
Application Note AN-IND-1-019

Author Vector Informatik GmbH

Restrictions Public Document

Abstract The ASAM AE XIL API 2.0 specification defines an API for performing tests in X-In-The-Loop simulators. This document describes CANoe support for these software interfaces.

Table of Contents

1.0 Overview

CANoe provides open programming interfaces for use with test automation systems, including CANoe COM Server, the FDX protocol and CANoe MATLAB®/Simulink®. From version 8.2, CANoe also supports parts of the ASAM AE XIL API 2.0 specification. It is based on version 2.0.1 of the specification, which was the current release at the time of development.

Please note that ASAM AE HIL API 1.x support was added to CANoe in version 7.5. While this API is still supported in the latest CANoe version, Vector recommends using ASAM AE XIL API 2.0.

1.1 Use Case / Motivation

HIL simulators typically consist of various hardware components, sometimes from different manufacturers, which are usually integrated into the overall test system via proprietary software interfaces. In the ASAM AE XIL API specification, these interfaces are designated as drivers.

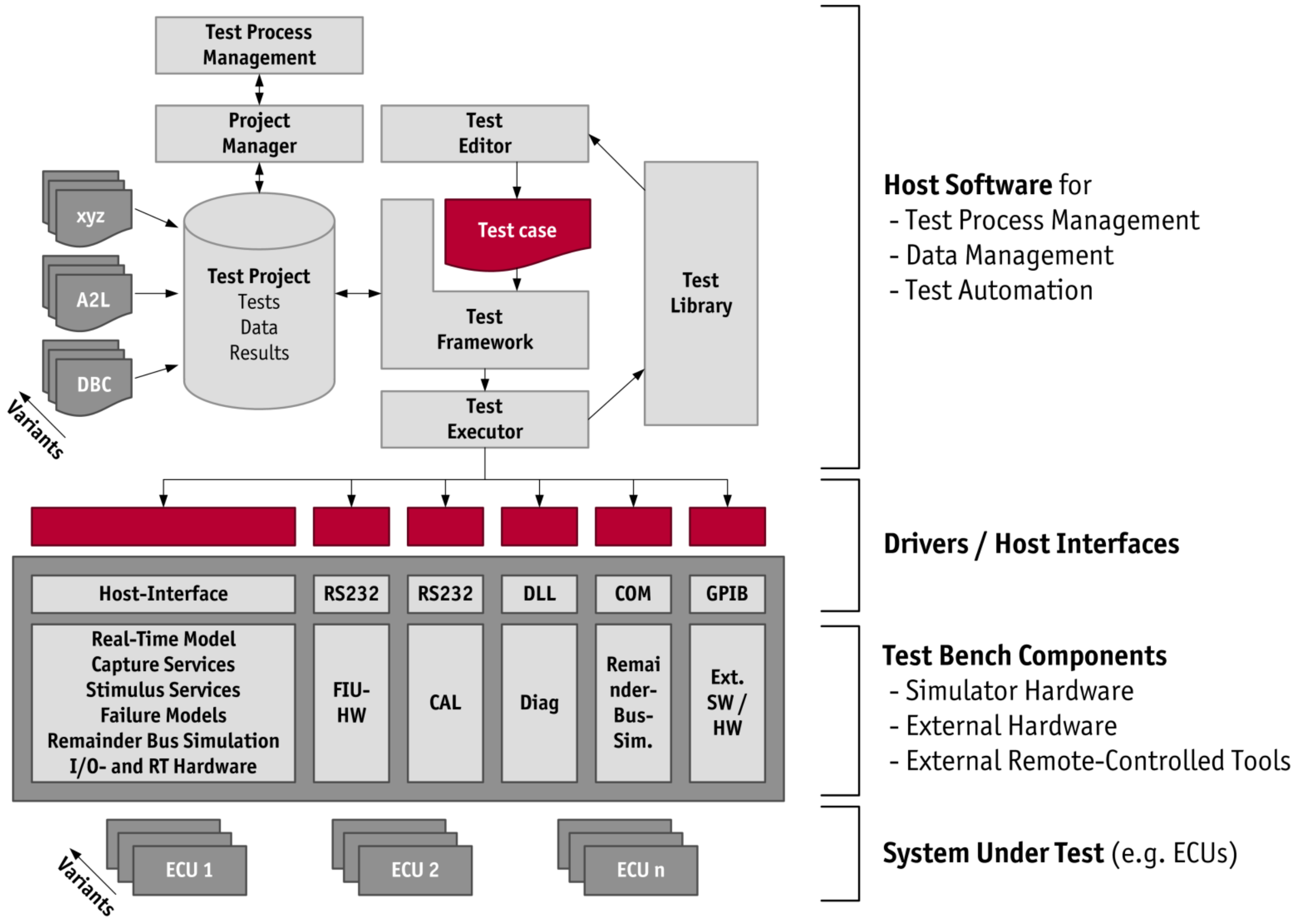


Figure 1: HIL test bench architecture (source: ASAM AE XIL API specification)

Having different driver APIs complicates the modification, extension and maintenance of the test system in a number of ways:

- There is a tight linkage between the test automation software and the hardware used. This makes changing one of these components almost impossible.
- Test cases can't be swapped across different test systems because the hardware is controlled via different drivers.
- It is time-consuming to combine test automation software from one manufacturer with another manufacturer's test hardware.
- Experience with using and programming a test system for one manufacturer does not easily translate to other manufacturers' systems. This results in additional staff training costs.

> Migration to newer, more up-to-date test systems is hampered by compatibility issues and proprietary data formats. It is therefore not possible to meet the requirement that test scripts can be swapped by manufacturers and suppliers.

The ASAM AE XIL API specification addresses these problems by defining a uniform software interface for test hardware. The software interface is divided into different application domains such as model access, diagnostics, electrical error simulation, measurement and calibration.

1.2 CANoe as a HIL Platform

CANoe can be used in a wide range of HIL applications, which may have different requirements. In the case of small and mid-sized HIL systems with moderate real-time requirements and moderate levels of complexity, CANoe in combination with the Vector VT System provides everything needed to implement a HIL system:

> Test script control and execution platform
> Real-time-capable execution platform for simple environment models
> Signal conditioning
> Sensor/actuator simulation as interface to the SUT (system under test)
> Network communication with the SUT, including remaining bus simulation
> Automatic and/or autonomous execution (standalone mode) if needed

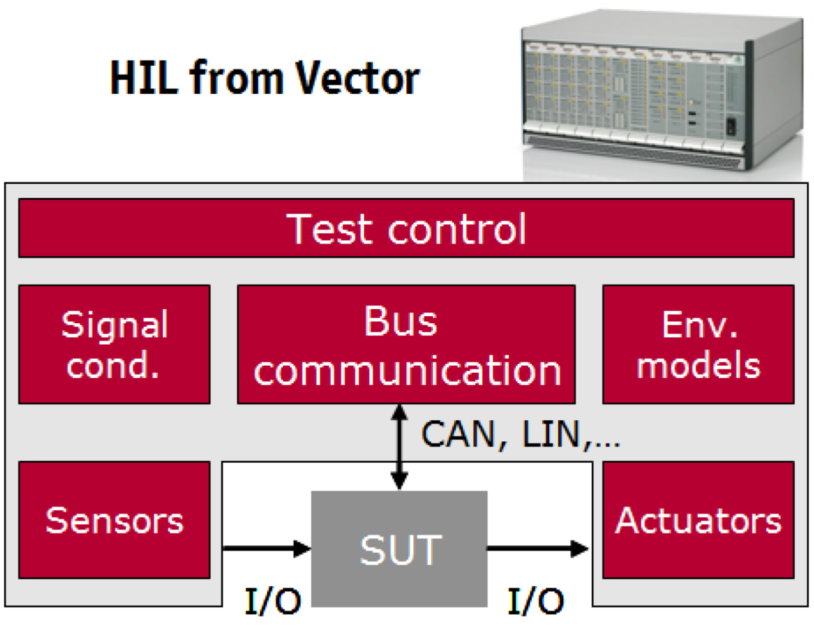


Figure 2: CANoe and VT System as a mid-sized HIL system

Combining the VT System, which is tailored to the electrical requirements of automotive ECUs, with cost-effective, PC-based HIL computers makes using these standard products an attractive option for both component and integration tests.

Large HIL systems, on the other hand, meet the highest requirements in terms of real-time capability and model complexity. Such systems are often based on hardware components from several manufacturers. CANoe can also be successfully integrated into such systems, where it is typically responsible for the network interface to the SUT as well as the remaining bus simulation.

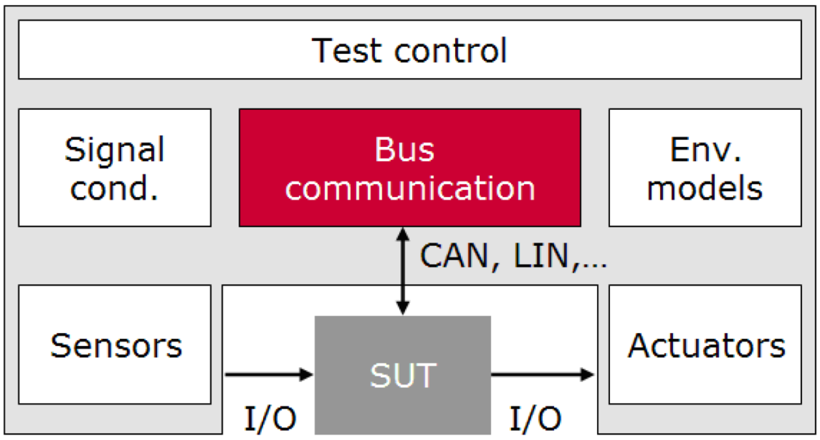


Figure 3: CANoe as a component in a larger HIL assembly

2.0 XIL API Support in CANoe

2.1 Supported APIs

In CANoe, all important values can be mapped to internal variables, i.e. so-called system variables. System variables in CANoe can be accessed via the ASAM XIL API. Reading and writing of model variables via the Model Access Port is supported via system variables. Also supported by CANoe are the Diagnostics and the EES-Port.

Other Ports (e.g. ECU-M- and ECU-C-Port) are not yet implemented in the CANoe XIL API. These ports may be added in future versions of CANoe.

2.2 Interface Concept

The CANoe XIL API is implemented as a .NET component, which implements the ASAM XIL API interfaces. Data is exchanged via CANoe's FDX protocol (read/write model variables). The CANoe FDX protocol is a UDP/IP-based software interface that enables the rapid exchange of CANoe system and environment variables as well as bus signals.

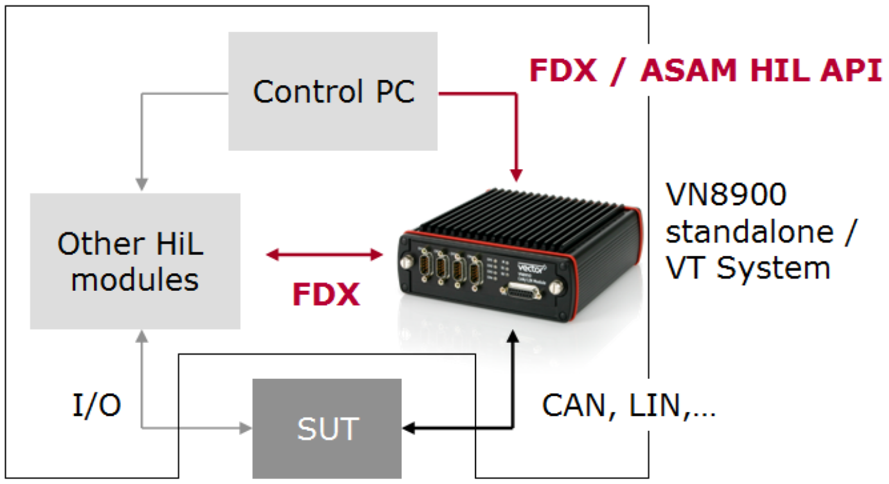


Figure 4: CANoe XIL API interface concept

2.3 Interface Configuration

CANoe FDX description files are used to configure the CANoe XIL API for accessing variables. The variables to be exchanged via FDX are arranged into data groups. The usual procedure is to define at least one group of read variables and one group of write variables in the description file. The actual settings that determine which group is read and which group is written are specified by the vendor specific port configuration files.

For this reason, the FDX description files need both to be included in the CANoe configuration (**Options|Extensions|XIL API & FDX Protocol**) and to be available in the test automation system, the XIL API application.

Furthermore, a vendor specific configuration file and a configuration for each of the used ports is required by the XIL API. These files are written in XML. Two example files are part of the Seat Tester XIL API demo configuration shipped with CANoe. You may reuse these files by simply adjusting some parts of them (e.g. assembly file locations).

3.0 Configuring the CANoe XIL API Interface

The MA.Port configuration file is required to configure the Model Access Port of CANoe. It contains the IP-address and port information of the computer running CANoe. This is necessary so that the CANoe XIL API can connect to CANoe via the FDX protocol.

This file also contains the path of the FDX description file to be used with your CANoe configuration. All FDX groups that shall be used by the XIL API have to be listed here.

The following example of how this file has to be structured is taken from the Seat Tester XIL API demo configuration shipped with CANoe:

```
<Port type="MAPort">
  <ModelFile></ModelFile>
  <IpAddress>127.0.0.1</IpAddress>
  <FdxPort>2809</FdxPort>
  <XilPort>3030</XilPort>
  <FdxFile>..//..//VendorConfigs//Fdx.xml</FdxFile>
  <FdxReadGroup>1</FdxReadGroup>
  <FdxWriteGroup>2</FdxWriteGroup>
</Port>
```

4.0 Working with the CANoe XIL API Interface

Working with the CANoe XIL API is done as specified by the ASAM XIL API Specification (see chapter 6.0). Some of the basic tasks will be explained below for clarification.

4.1 Setting up the XIL API

After you have prepared the required vendor specific XML configuration files you can instantiate the CANoe XIL API like this:

```
var factory = new TestbenchFactory();
mTestbench = factory.CreateVendorSpecificTestbench("Vector", "CANoe", "10.0");
mMaPort = mTestbench.MAPortFactory.CreateMAPort("CANoeMAPort");
mMaPort.Configure(
mMaPort.LoadConfiguration("..//Manifests//MAPortConfig.xml"), true);
mMaPort.StartSimulation();
```

Of course, the specified files and paths have to comply with your local setup and configuration files. After executing the above code the MA Port will be up, running and connected with CANoe via FDX.

4.2 Using the Model Access Port

The main purpose of the MA Port is reading and writing of variables. In addition, the port offers a wide range of different signal generators that can be used to stimulate variables. Both functionalities will be explained in the upcoming chapters.

4.2.1 Reading and Writing of Variables

Reading and writing variables via the Model Access Port is very straight forward:

```
// Read a value
IIntValue intValue = mMaPort.Read("Test::IntegerVariable") as IIntValue;
Console.WriteLine("The read value is: " + intValue.Value);
// Write a value
mMaPort.Write("Test::IntegerVariable", new IntValue(7));
```

4.2.2 Signal Generators

The signal generators provided by the XIL API offer a huge amount of functionality. The following example demonstrates only the basics of instantiating, configuring and running such a generator. For more detailed information please refer to the official ASAM XIL API specification.

```
// Create a generator
var generator = mMaPort.CreateSignalGenerator();

// Add assignments
generator.Assignments = new Dictionary<string, string>();
generator.Assignments.Add("FloatVar", "SomeNamespace::FloatVariable");

// Add signals
var signals = mTestbench.SignalFactory.CreateSignalDescriptionSet();

var constSegment1 = mTestbench.SignalFactory.CreateConstSegmentBySymbols(
  mTestbench.SymbolFactory.CreateConstSymbolByValue(5.0),
  null,
  mTestbench.SymbolFactory.CreateConstSymbolByValue(12.0));
var constSegment2 = mTestbench.SignalFactory.CreateConstSegmentBySymbols(
  mTestbench.SymbolFactory.CreateConstSymbolByValue(4.0),
  null,
  mTestbench.SymbolFactory.CreateConstSymbolByValue(8.0));
var segmentSignalDescription =
mTestbench.SignalFactory.CreateSegmentSignalDescriptionByName("FloatVar");

segmentSignalDescription.Add(constSegment1);
segmentSignalDescription.Add(constSegment2);

signals.Add(segmentSignalDescription);
generator.SignalDescriptionSet = signals;

// Download and start the generator
generator.LoadToTarget();
generator.Start();
```

4.2.3 Capturing

Capturing is not yet part of the CANoe XIL API implementation. This API will be supported in future CANoe versions.

5.0 Example

The aim is to read in two model variables (ModelOutput1 and ModelOutput2) and write two model variables (ModelInput1 and ModelInput2) via the CANoe XIL API.

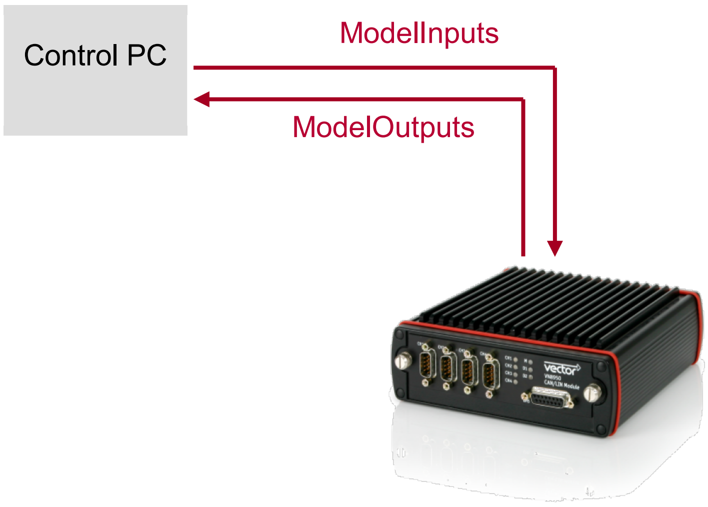
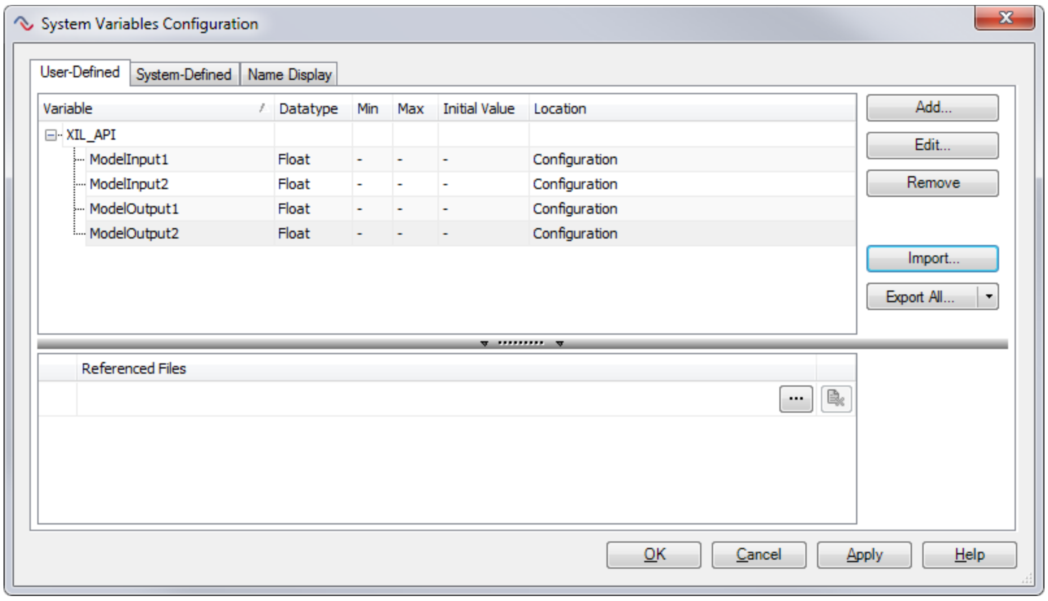


Figure 5: Input and output of model variables

Please take the following steps in order to configure the CANoe XIL API:

1. Configure the required system variables in CANoe.



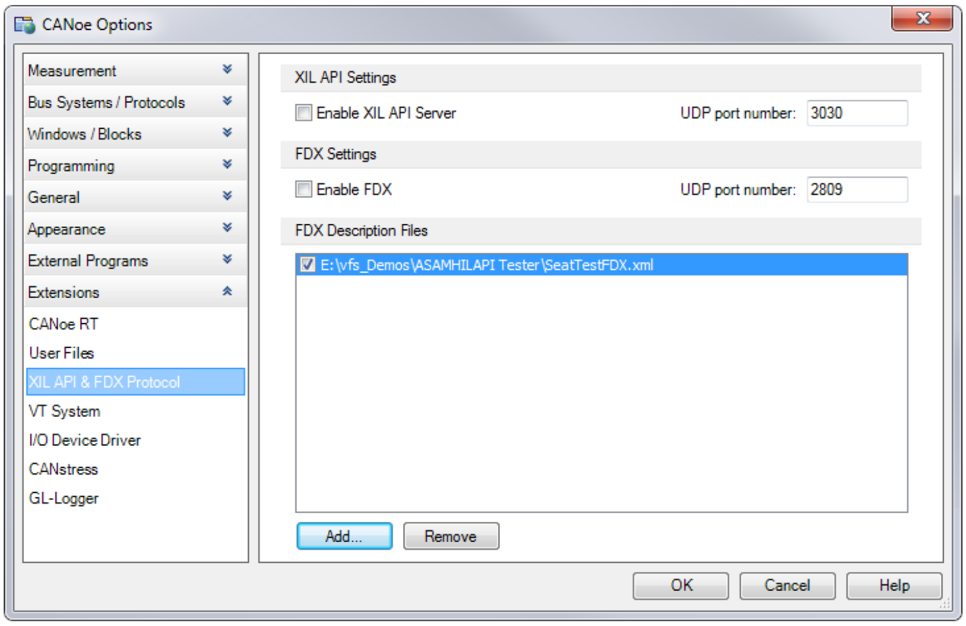
Variable	Datatype	Min	Max	Initial Value	Location
XIL_API					
ModelInput1	Float	-	-	-	Configuration
ModelInput2	Float	-	-	-	Configuration
ModelOutput1	Float	-	-	-	Configuration
ModelOutput2	Float	-	-	-	Configuration

Figure 6: Configuring the system variables

2. Create an FDX description file with one group each for the read and write variables. Please take special care to specify the memory layout of the data groups correctly (group size, data types, data values and offsets). For more information about the FDX protocol, please see chapter 6.0.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<canoefdxdescription version="1.0">
    <datagroup groupID="1" size="16">
        <identifier>ModelInputs</identifier>
        <item type="double" size="8" offset="0">
            <sysvar namespace="XIL_API" name="ModelInput1"/>
        </item>
        <item type="double" size="8" offset="8">
            <sysvar namespace="XIL_API" name="ModelInput2"/>
        </item>
    </datagroup>
    <datagroup groupID="2" size="16">
        <identifier>ModelOutputs</identifier>
        <item type="double" size="8" offset="0">
            <sysvar namespace="XIL_API" name="ModelOutput1"/>
        </item>
        <item type="double" size="8" offset="8">
            <sysvar namespace="XIL_API" name="ModelOutput2"/>
        </item>
    </datagroup>
</canoefdxdescription>
```

3. Configure the FDX description file in CANoe.



XIL API Settings

☐ Enable XIL API Server — UDP port number: 3030

FDX Settings

☐ Enable FDX — UDP port number: 2809

FDX Description Files

☑ E:\vfs_Demos\ASAMHILAPI Tester\SeatTestFDX.xml

Figure 7: Configuring FDX description

4. Prepare the vendor specific configuration files. Examples for both files can be found in the Seat Tester XIL API demo configuration shipped with CANoe. You may use these files as references and simply adjust the paths used in the files to your needs.
   a. File “Vector.imf”: This file specifies the exact XIL API implementation assembly to be used. In this case the file should point to the Vector CANoe XIL API assembly
   b. File “VectorMAPortConfig.xml”: This file contains the configuration for the Model Access Port provided by CANoe (e.g. the IP-address and port to use for UDP communication).
5. Create your test program in a Microsoft .NET language. A test script could, for example, look like this:

```
static public bool RunTest(IMAPort maPort, ITestEnvironment environment)
{
    environment.Output("Starting test...");
    IFloatValue value1 = new FloatValue();
    IFloatValue value2 = new FloatValue();
```

```
        for (int i = 0; i < 100; ++i)
        {
            IFloatValue value3 = (FloatValue)maPort.Read("HILAPI/ModelOutput1");
            IFloatValue value4 = (FloatValue)maPort.Read("HILAPI/ModelOutput2");
            value1.Value = value3.Value + value4.Value;
            value2.Value = value3.Value * value4.Value;
            maPort.Write("XILAPI/ModelInput1", value1.Value);
            maPort.Write("XILAPI/ModelInput2", value2.Value);
            environment.Wait(TimeSpan.FromMilliseconds(50));
        }

        environment.Output("Test completed");
        return true;
    }
```

6.0 Additional Resources

ASAM AE XIL API SPECIFICATION	HTTP://WWW.ASAM.NET/
VECTOR MANUAL	CANoe_FDX_Protocol_EN.pdf (CANoe FDX Protocol)
CANoe DEMO	Seat Test HILAPI Demo Configuration

7.0 Contacts

For a full list with all Vector locations and addresses worldwide, please visit http://vector.com/contact/.