Open eHS Real-Time Power Electronics Simulation Toolbox (NI PXIe-7976R)

User Manual

By OPAL-RT TECHNOLOGIES

1751 Richardson, suite 2525
Montréal (Québec) Canada H3K 1G6

www.opal-rt.com

Printed in Canada

Revision History:

The following table shows the revision history for this manual.

Version	By	Action	Action date
2016.0.0.1	Yousef B. Bedoustani, Ph.D. LabVIEW Developer	First release of LabVIEW project and manual	September 1st , 2016
2016.0.1.0	Yousef Bedoustani, Jonathan Bouchard, Nadine Hariri	Updated to reflect additional requirements & specific application information	September 29, 2016
2016.0.1.1	Jonathan Bouchard, Nadine Hariri	Clarified certain parts of document	October 6, 2016
2016.0.1.2	Jonathan Bouchard, Nadine Hariri	Updated document for NI Tools Network	October 14, 2016

Contents

ABSTRACT

This manual explains how to use the *Open eHS Real-Time Power Electronics Simulation Toolbox (NI PXIe-7976R)* develped by OPAL-RT Technologies. This toolbox uses the National Instruments PXIe platform to run the eHS core(s) and related FPGA firmware.The National Instruments platform used for this toolbox is the NI PXIe-7976R.

This toolbox leverages the eHS Gen3 core which is the latest generation of an FPGA based electrical solver developed by OPAL-RT TECHNOLOGIES for the real-time simulation of power electronic circuits. Users will learn how to generate the eHS matrix, configure the eHS Gen3 IP core and communicate with it using the provided LabVIEW libraries.

INTRODUCTION

WHAT IS THE EHS?

eHS is an FPGA-based floating-point solver that allows the user to simulate an electric circuit on FPGA automatically, without having to code in VHDL or calculate the system equations. It combines the simplicity of building electric circuit models using the SimPowerSystems™ Toolbox, PLECS Blockset, PSIM or NI Multisim 13 with the strength of OPAL-RT FPGA-based simulators to solve the currents and voltages within the circuit in real-time, with a sample time in the range of 150ns to 2.5µs.

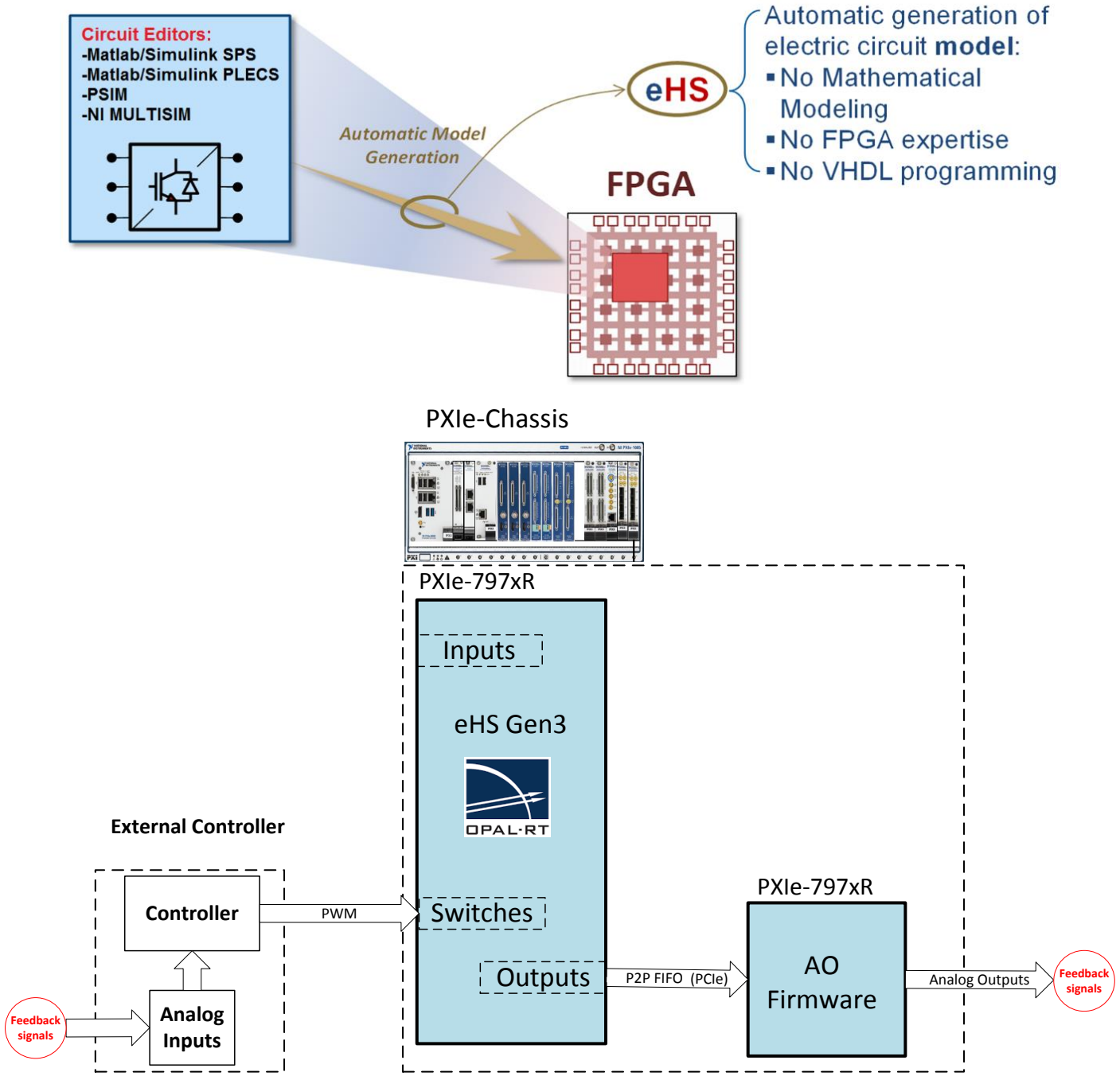


Figure 1: Real-time simulation using eHS in FlexRIO platform

KEY FEATURES

Reconfigurability

All the matrices required for solving the system are loaded into the FPGA engine when the simulation is loaded, without needing to reconfigure the engine with a specific firmware for each application. This provides the users with a simple workflow to run real-time simulation of electrical systems.

Performance

The sample time of the electrical system solved by the eHS ranges from 150ns to 2.5µs, depending on the circuit complexity. For example the one phase boost converter and three phase 2-level Inverter can be simulated with sample time of 200 ns and 250 ns respectively.

Circuit size

The maximum number of elements in the circuit that can be simulated is as follows:

- Up to 72 switching elements (**SW01** to **SW72**)
- Up to 32 current or voltage sources (**U01** to **U32**)
- Up to 32 current or voltage measurements (**Y01** to **Y32**)
- Up to 150 L/C components including L, C, LC and RLC. Note that LC and RLC branches each count as two L/C devices
- Mutual and transformer support
- Unlimited number of resistors

** eHSx64 Gen3 is the latest version of the eHS family. It runs at a 200 MHz FPGA clock frequency. Based on FPGA resources, it is possible to implement more than one eHS core within one FPGA firmware.*

INTENDED AUDIENCE AND REQUIRED SKILLS AND KNOWLEDGE

The intended user of the eHS is an R&D developer, a test integrator, or a control specialist needing an easily reconfigurable, very high speed, and low latency electrical circuit FPGA-based simulator and who does not necessarily have the knowledge and the workflow required for FPGA development.

SUPPORTED CIRCUIT EDITORS

Before starting the real-time simulation, users need to define the circuit configuration, using an appropriate circuit editor. The following circuit editors are supported by OPAL-RT TECHNOLOGIES.

- SimPowerSystemsTM Simulink Toolbox
  MATLAB®/Simulink® is a software package developed by MathWorks® that allows modeling, simulation and analysis of dynamic systems. eHS can use SimPowerSystemTM Simulink toolbox to define models to be executed. Users are expected to have a clear understanding of Simulink operation, particularly regarding model definition and simulation parameters. The SimPowerSystemsTM toolbox provides the libraries and analysis tools to model and simulate electrical systems. It is used by eHS only as a circuit description environment. The SimPowerSystemsTM simulation engine can also be used to validate the eHS results.
- PLECS Blockset
  PLECS® is the software of offline simulations of power electronic systems. It is available in two editions: PLECS Blockset for seamless integration with MATLAB®/Simulink® and PLECS Standalone, a completely independent product. OPAL-RT only supports PLECS Blockset within MATLAB®/Simulink®.

- PSIM
  PSIM is an engine for offline simulation of power electronics. PSIM uses a strong algorithm dedicated to electrical circuits (piecewise method, generic models and a fixed time-step). It is used by eHS only as a circuit description environment. The PSIM simulation engine can also be used to validate the eHS results.

- NI Multisim v13
  NI Multisim is an industry-standard, best-in-class SPICE simulation environment developed by National Instruments. It is the cornerstone of the NI circuits teaching solution to build expertise through practical application in designing, prototyping, and testing electrical circuits. It can hence be useful to use the Multisim simulation engine to validate the results of the eHS.

WORKFLOW FOR USING THE EHS GEN3 FOR BASIC DEVELOPMENT

1. Configure your hardware and software system.
2. Explore the template project that ships with the *Open eHS Real-Time Power Electronics Simulation Toolbox (NI PXIe-7976R)* package and the tutorials provided in the *Open eHS Real-Time Power Electronics Simulation Toolbox (NI PXIe-7976R) - User Manual*.
3. Design the eHS and AO firmware that will meet your system requirements. To accelerate the LabVIEW FPGA coding, make use of the templates that are provided as part of the *Open eHS Real-Time Power Electronics Simulation Toolbox (NI PXIe-7976R)* package.
4. Design the real-time driver that will meet your system requirements. To accelerate the LabVIEW Real-Time coding, make use of the templates provided as part of the *Open eHS Real-Time Power Electronics Simulation Toolbox (NI PXIe-7976R)* package.
5. Use one of the supported circuit editors to design your circuit.
6. Generate the circuit matrices and load them on to the target simulator using the eHS Gen3 matrix generation VI.
7. Run the LabVIEW real-time process to monitor the voltage/current measurements and update parameters.

ACCESSING THE GETTING STARTED GUIDE

In a Windows file explorer window, browse to the directory where the package is installed and navigate to the *Documentation* folder to access the *Open eHS Real-Time Power Electronics Simulation Toolbox (NI PXIe-7976R) - Getting Started Guide*.

GETTING SUPPORT FOR THE POWER ELECTRONICS REAL-TIME SIMULATOR BY OPAL-RT

For support running this application, visit www.opal-rt.com/support.

For support installing NI software or setting up NI hardware, contact NI support at http://www.ni.com/support/.

END-USER LICENSE AGREEMENT

In a Windows file explorer window, browse to the directory where the package is installed and navigate to the *Documentation* folder to locate the end-user license agreement.

NAMING CONVENTION AND SUPPORTED CIRCUIT EDITORS

COMPONENT NAMING CONVENTION

The algorithm implemented by OPAL-RT to convert the SimPowerSystems™, PLECS Blockset, PSIM, or NI Multisim circuit file into the data used by the eHS (to compute the currents and voltages in the circuit in real time) involves the analysis of the circuit file to find the supported elements. For sources, switch control signals, and measurement outputs, the corresponding elements are listed in alphabetical order and will be accessed in that order within the LabVIEW environment. Therefore, to ease the mapping process of eHS inputs and outputs to the system I/Os, it is strongly recommended to rename elements with names that are easily recognizable and whose identification from within the eHS matrix generator will be straightforward. The following naming convention is strongly recommended:

- Switching devices should be named with the prefix **SW** followed by a 2-digit index, starting from **SW01**. Elements categorized as *switching elements* are: Diodes, IGBT/Diodes, Breakers, and Ideal Switches. This prefix can be followed by other characters.
- Sources, whether independent current or voltage sources, should have the prefix **U** followed by a 2-digit index, starting from **U01**. This prefix can be followed by other characters (e.g *U01 Vdc*). To avoid confusion, the prefix for a three-phase source should contain three indices, starting with phase A (e.g. *U05 U06 U07 Vabc*).
- Measurements of any kind should have the prefix **Y** followed by a 2-digit index, starting from **Y01**. This prefix can be followed by other characters (e.g *Y01 I_load*).
- Inductances, capacitors, and resistors are not accessed directly from the eHS matrix generator and do not have any special naming convention.

SIMULINK SIMPOWERSYSTEMS™ SUPPORTED BLOCKS

In the SimPowerSystems™ Elements library, the following blocks are supported (see Figure 2):

- Series RLC Branch
- Parallel RLC Branch
- Ground
- Breaker
- Transformers
- Mutual Inductance

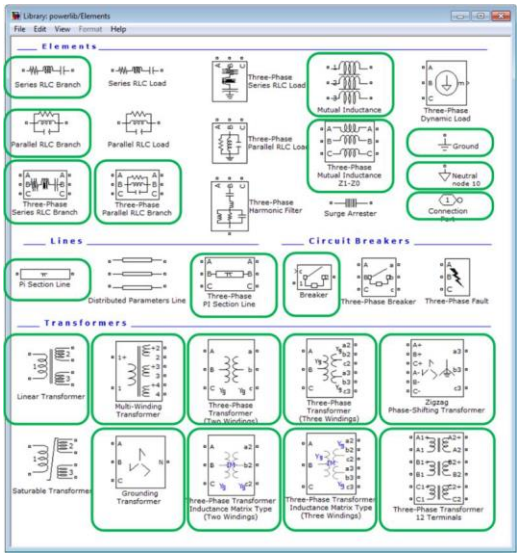


Figure 2: Elements supported in the SimPowerSystems™ Elements library

In the SimPowerSystems™ Measurements block library, the following blocks are supported (see Figure 3):

- Current Measurement
- Voltage Measurement
- Multimeter

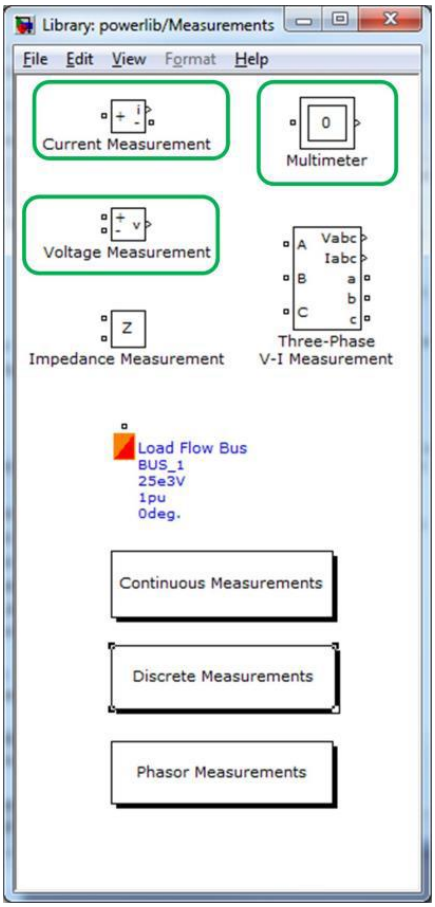


Figure 3: Elements supported in the SimPowerSystems™ Measurements block library

In the SimPowerSystems™ Power Electronics block library, the following blocks are supported (see Figure 4):

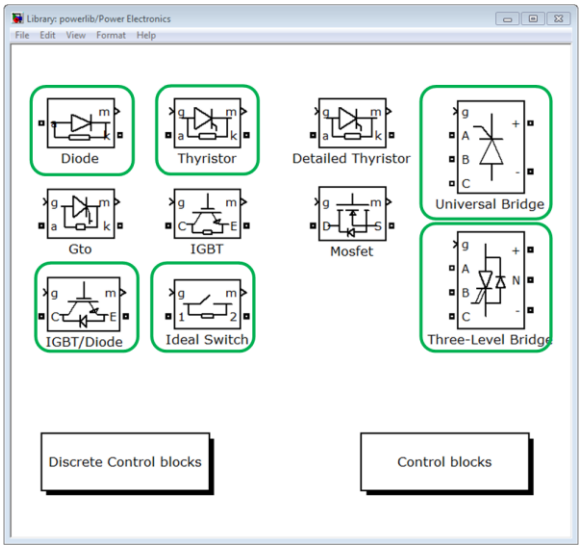


Figure 4: Elements supported in the SimPowerSystems™ Power Electronics block library

In the SimPowerSystems™ Electrical Sources block library, the following elements are supported (see Figure 5):

- DC Voltage Source
- AC Voltage Source
- AC Current Source
- Controlled Voltage Source
- Controlled Current Source
- Three-Phase Source

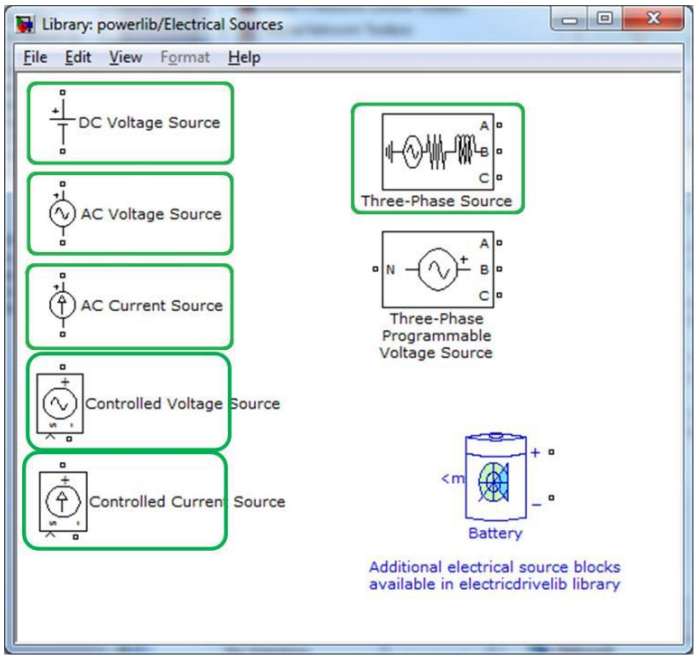


Figure 5: Elements supported in the SimPowerSystems™ Electrical Sources block library

Figure 6 shows an example of a three-phase inverter implemented with supported SimPowerSystems™ elements using the naming convention described earlier in this chapter.

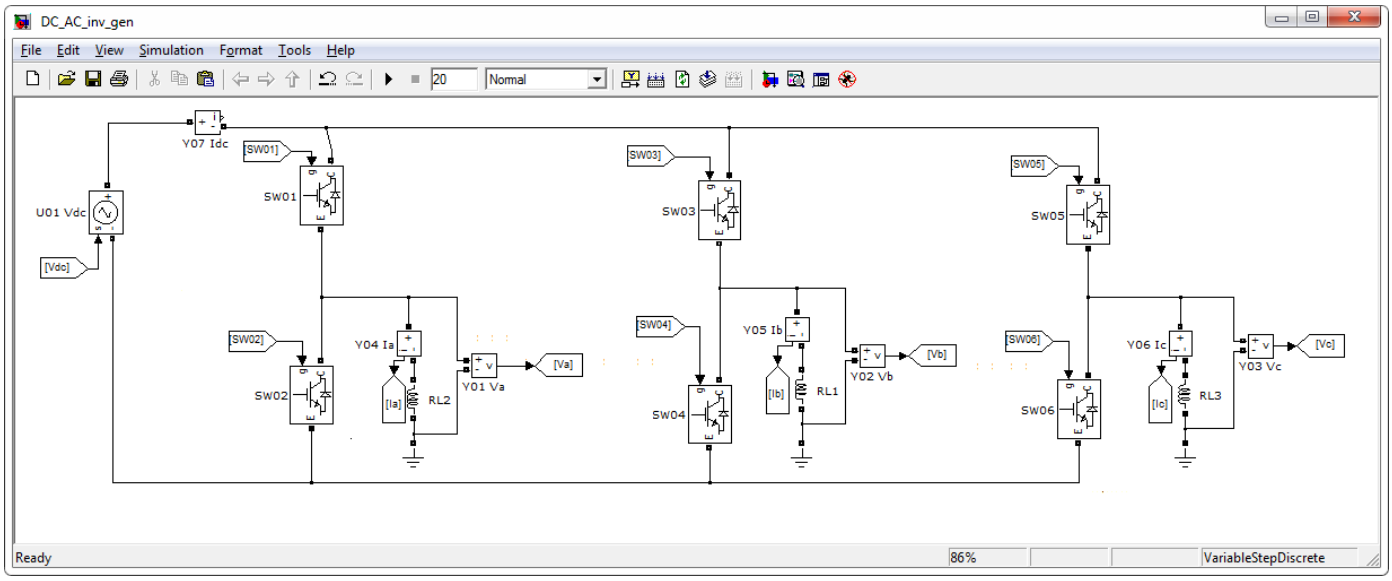


Figure 6: A three-phase inverter in SimPowerSystem

LCA SUPPORT IN SIMPOWERSYSTEMS™

LCA stands for Loss Compensation Algorithm. This feature removes the power losses that might occur when an IGBT/Diode switch is used (as a result of the Pejovic method). This technique works only for specific converter topologies. eHS Gen3 currently supports 2 types of LCA topologies: 2-level arm and 3-level NPC arm.

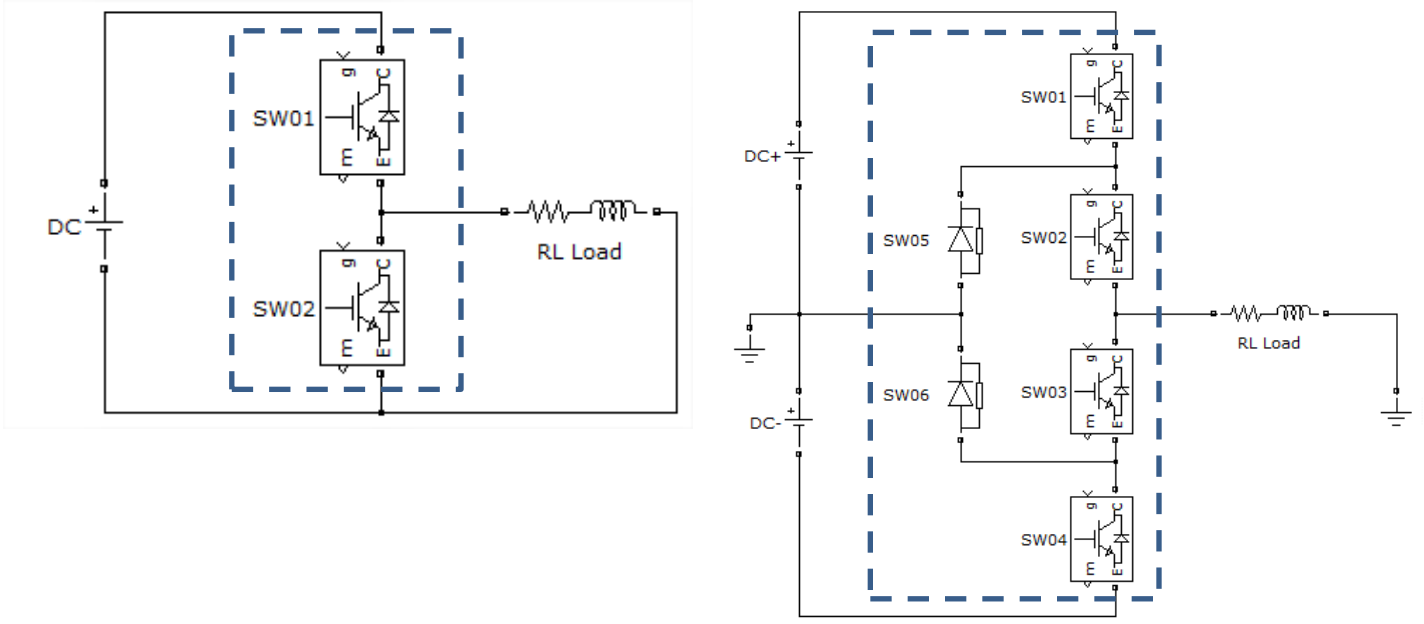


Figure 7: 2-levels converter's arm example and 3-levels converter's arm example

To use LCA in the SimPowerSystems™ environment, in the eHS circuit model, you must replace the 2-level arm and the 3-level NPC arm by the Universal Bridge and Three-Level Bridge blocks. Note that the forward voltages and Tf / Tt parameters will be discarded during the parameter extraction.

During generation of the eHS matrices, the Universal Bridge block will be detected as an LCA element and the losses compensation algorithm will be used instead of the classic switching logic.

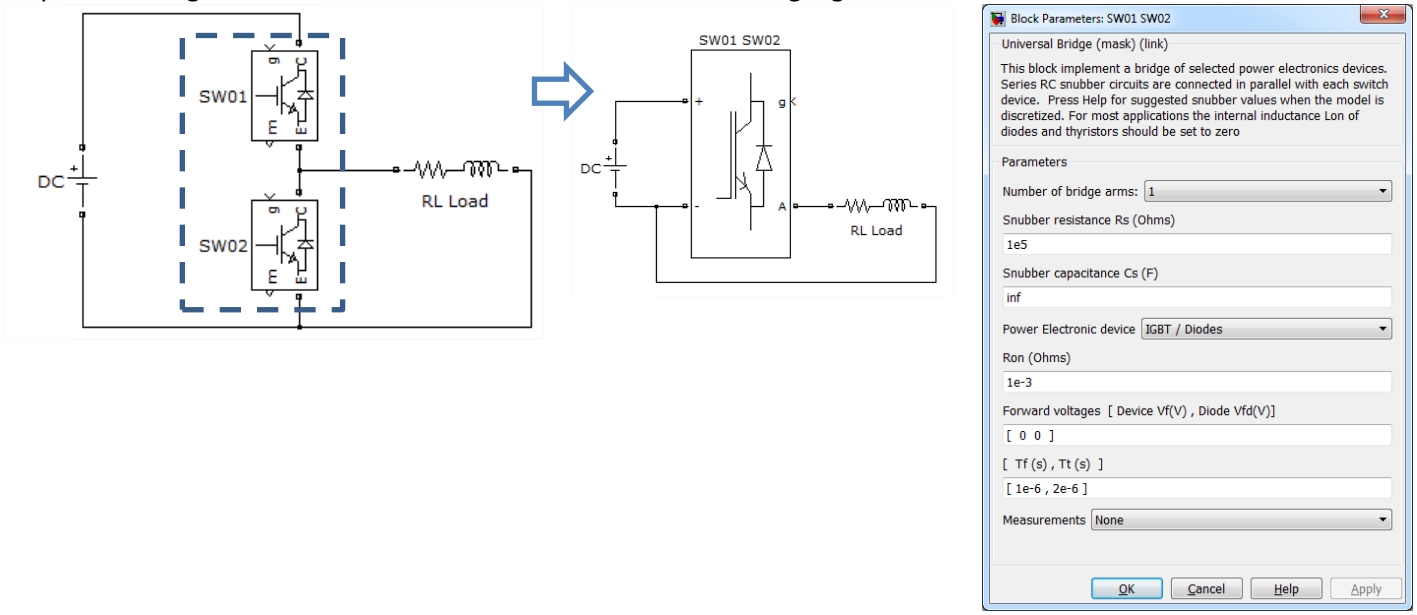


Figure 8: Use LCA on a 2-levels converter's arm and Universal bridge options

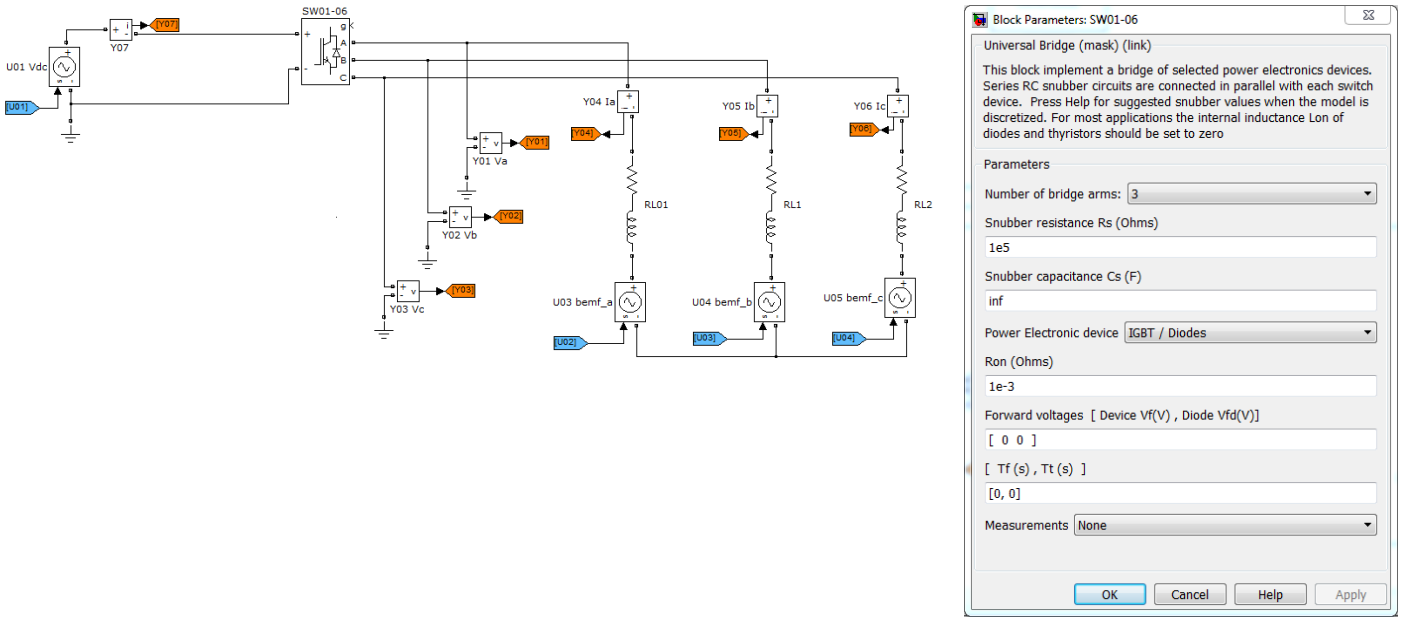


Figure 9: Using LCA component on a 2-level 3-phase inverter and Universal Bridge block parameters

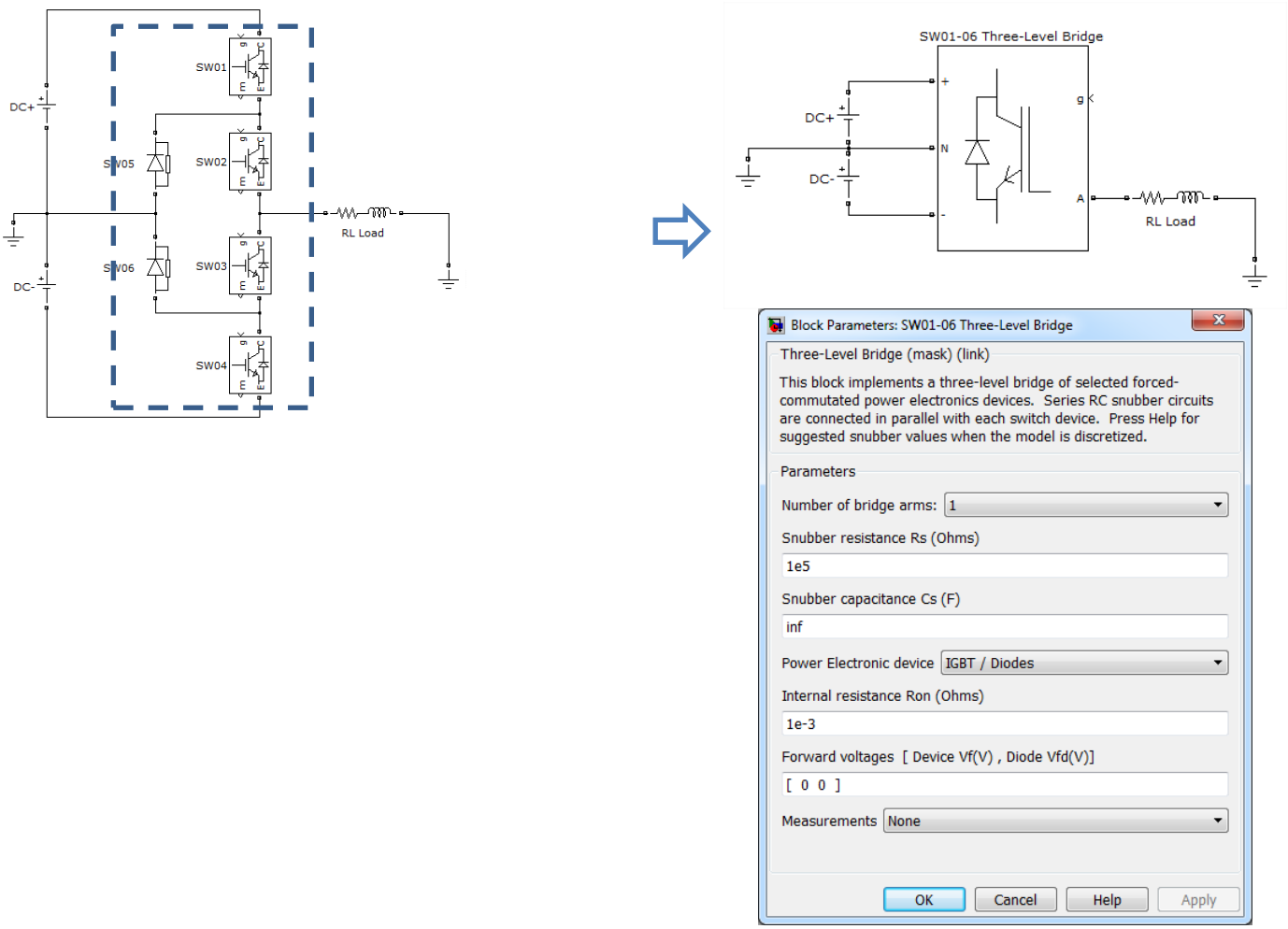


Figure 10: Use LCA on a 3-level NPC converter's arm

PLECS BLOCKSET SUPPORTED BLOCKS

Figure 11 shows the supported blocks from the PLECS Blockset library.

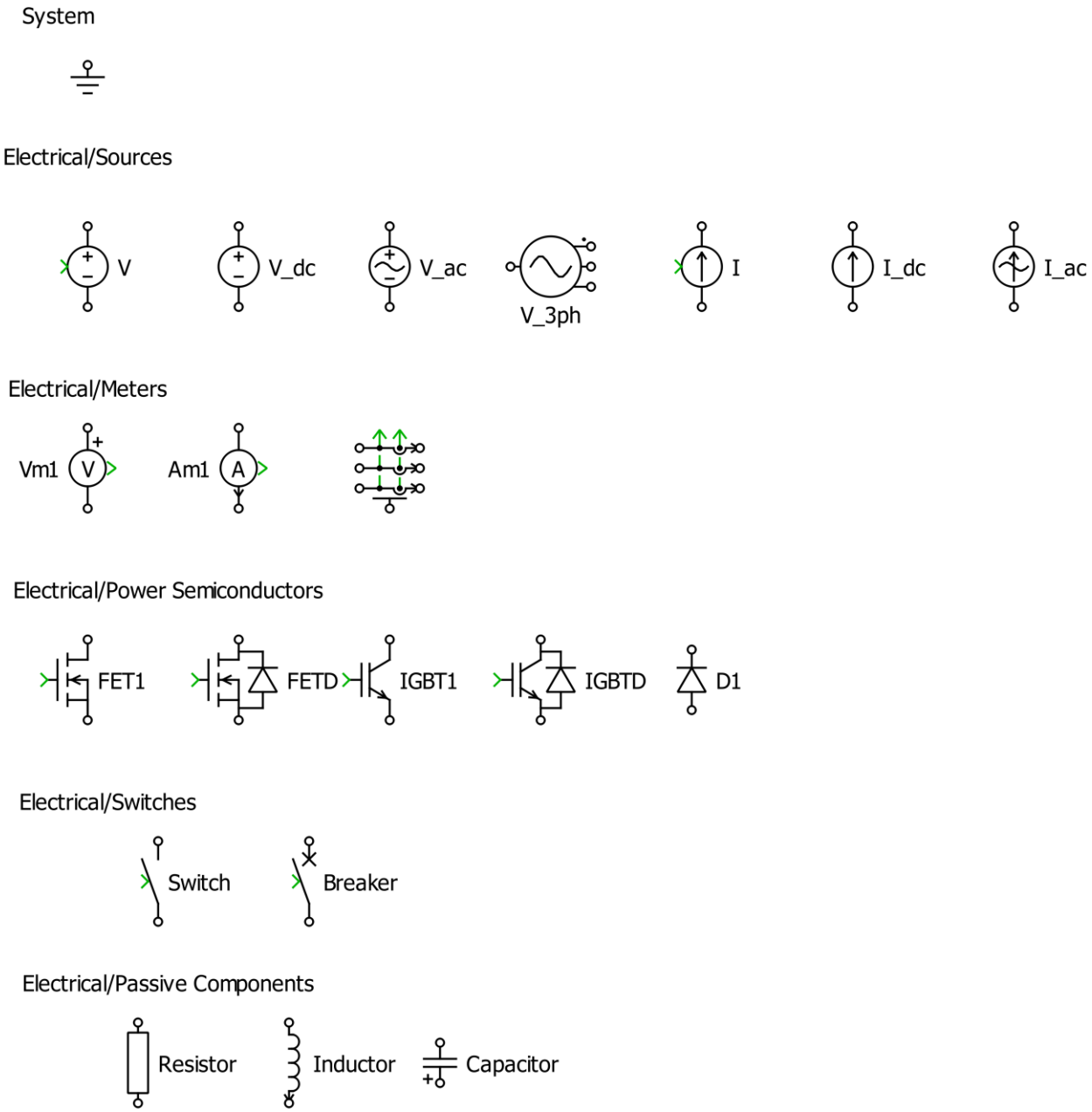


Figure 11: supported blocks from PLECS Blockset library

Figure 12 shows the additional elements from the PLECS Blockset which will be supported in the nearfuture.

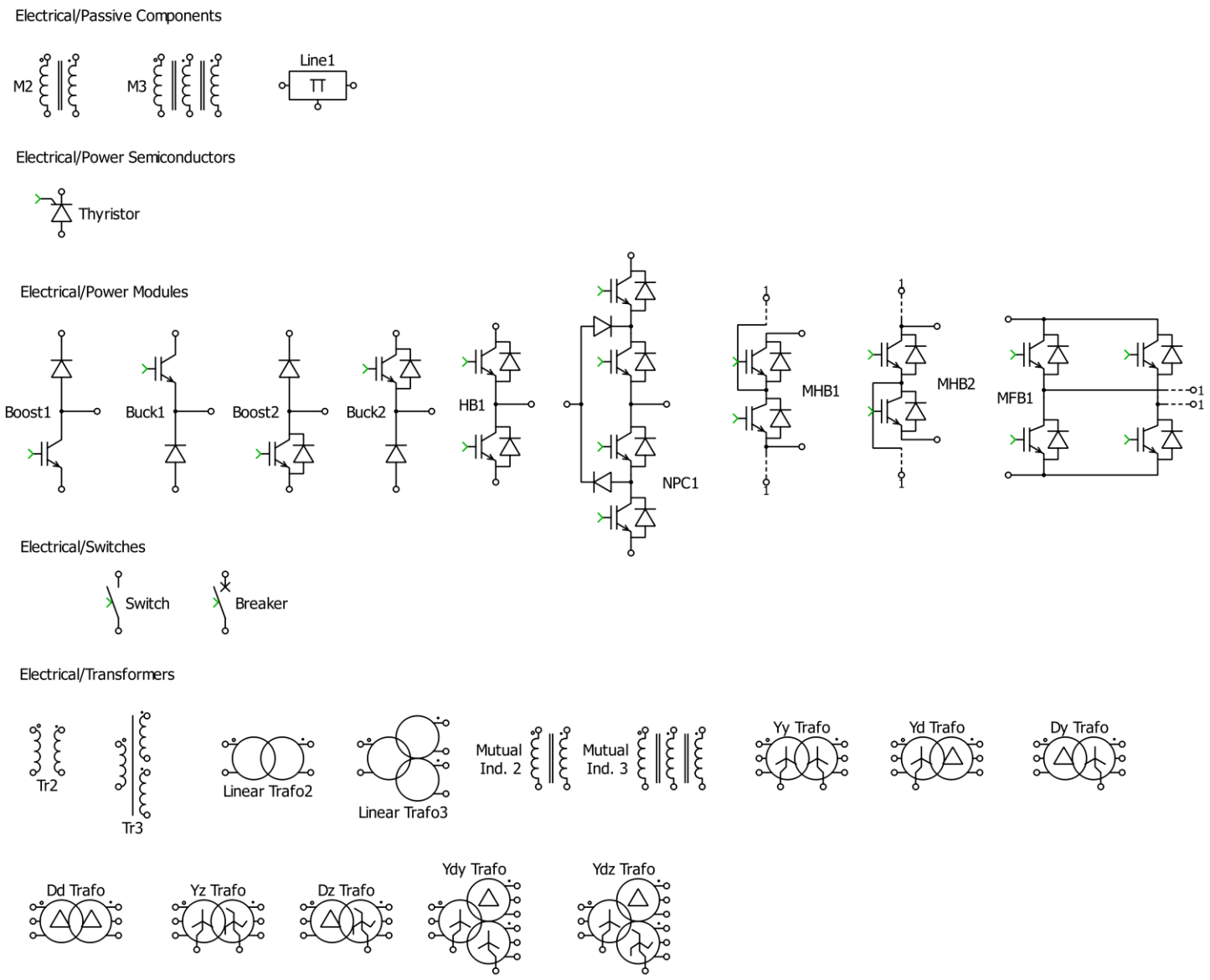


Figure 12: PLECS Blockset elements which will be supported in next release of the product

Figure 13 shows an example of a three-phase rectifier implemented with supported PLECS Blockset elements, with blocks named according to the naming convention described earlier in this chapter.

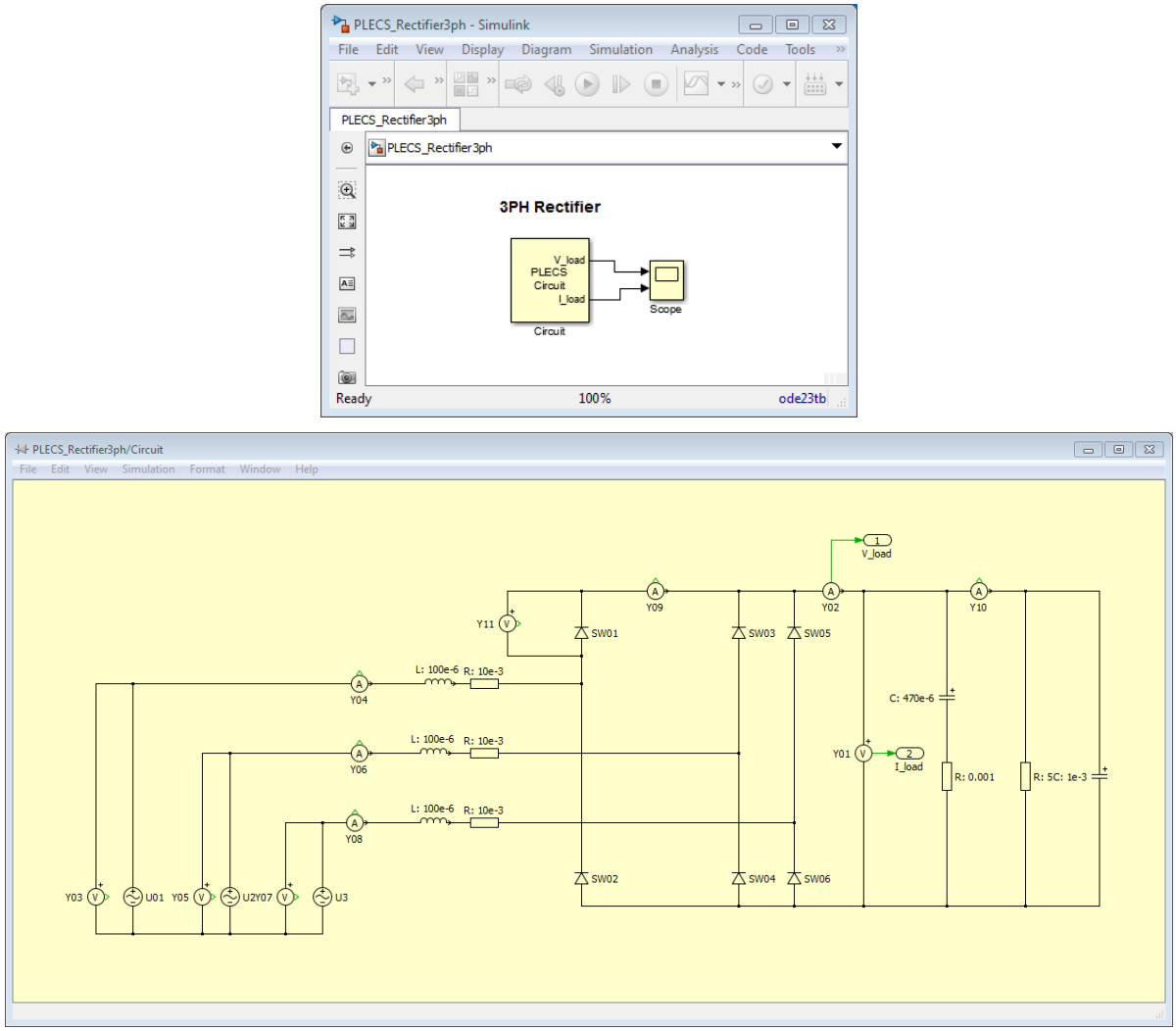


Figure 13: A three-phase rectifier implemented with supported elements and block named according to the naming rules in PLECS Blockset.

PSIM SUPPORTED BLOCKS

The circuit must be designed with blocks selected from a special subset of blocks from the PSIM libraries. The supported blocks are shown in the following table:

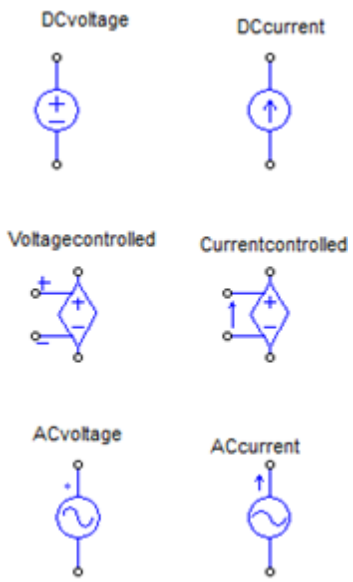
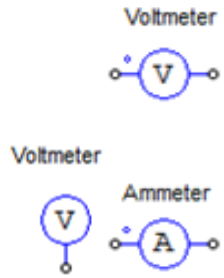
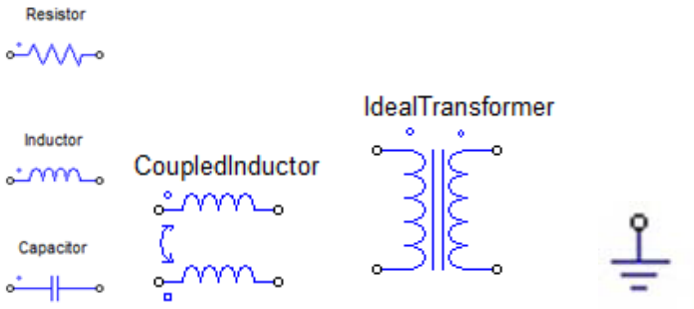
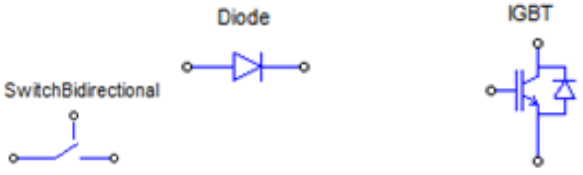
Name	Symbol
Sources eHS inputs (**U**)	
Voltage and current measurements point, eHS outputs (**Y**)	
Passive components, ideal transformer and ground	
Switches (**SW**)	

Figure 14 shows an example of a three-phase inverter implemented with supported Psim elements, with blocks named according to the naming convention described earlier in the chapter.

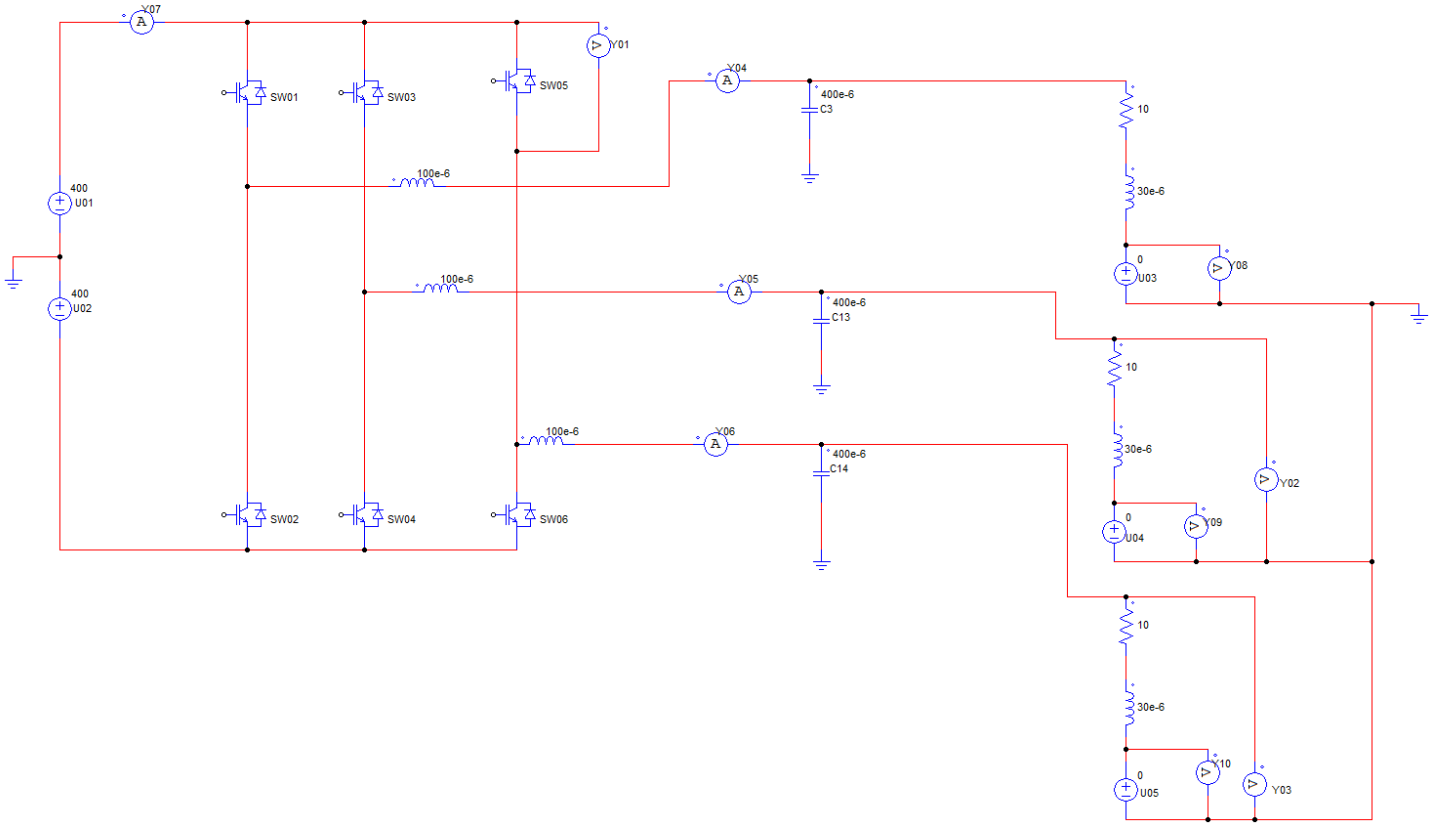


Figure 14: A three-phase inverter implemented with supported elements and block named according to the naming rules in PSIM.

NI MULTISIM SUPPORTED BLOCKS

The supported blocks for NI Multisim are available in the file “Exportable_OpalRT.ms13”. This file is located within the PSIM_Multisim_SPS_Models directory in the main LabVIEW project directory. Figure 15 shows this file. You can simply copy paste the blocks from this file to your model.

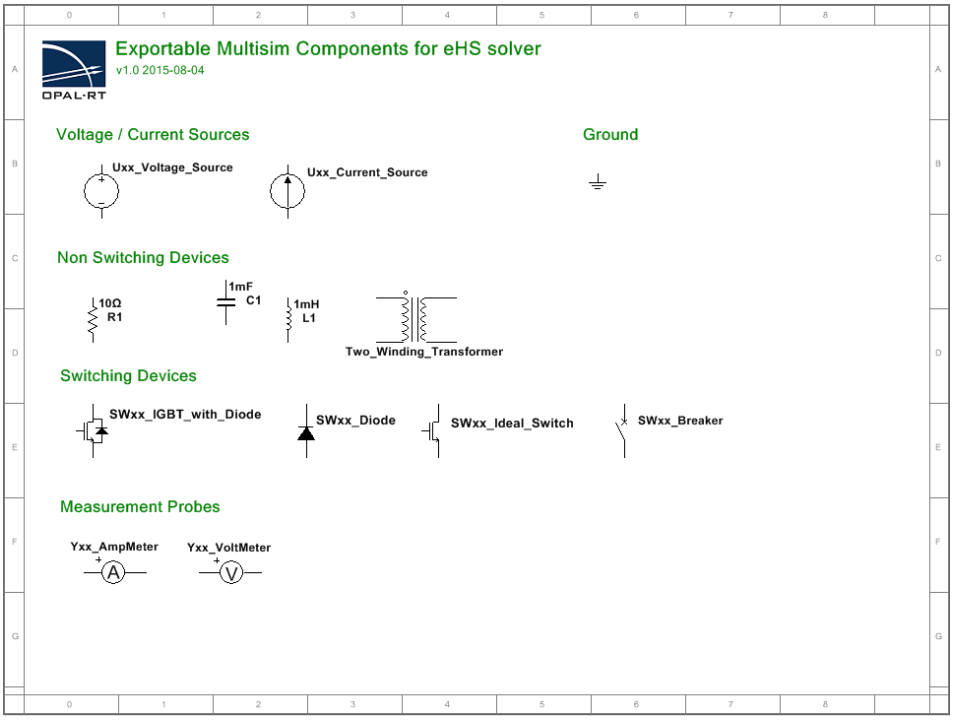


Figure 15: OPAL-RT liberay and supported elements in NI Multisim v13.

Figure 16 shows an example of a three-phase inverter implemented with supported NI Multisim (ver13) elements, with blocks named according to the naming convention described earlier in this chapter.
Note that for the NI Multisim workflow, you must enable the XML export and choose the proper path for the XML file. Refer to Appendix 3 for more details.

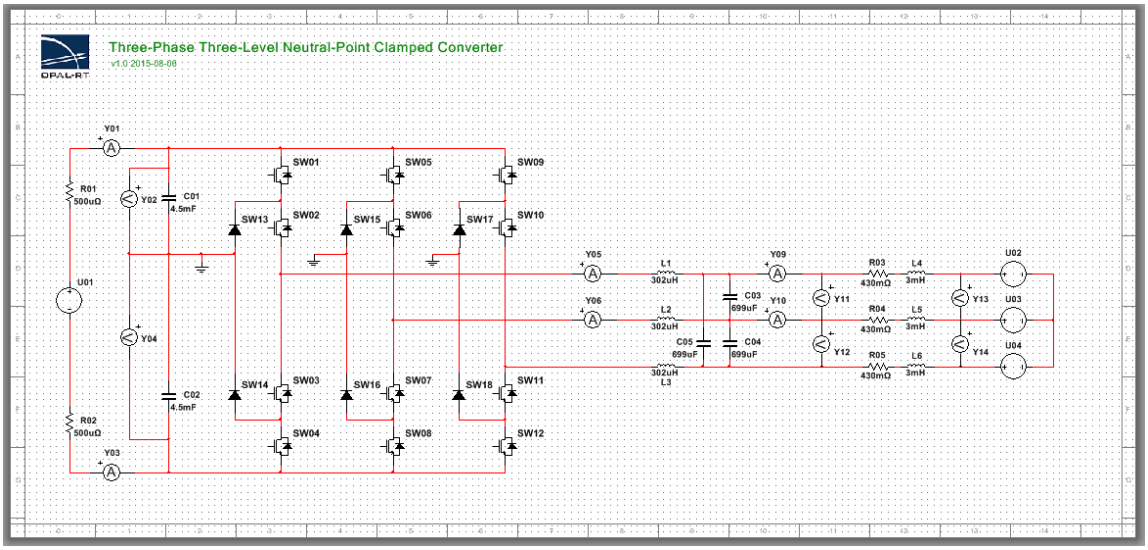


Figure 16: A three-phase inverter implemented with supported elements and block named according to the naming rules in NI Multisim.

SYSTEM REQUIREMENTS

In this document we assume that the user is familiar with LabVIEW programming. This section explains only the software and hardware needed to run the *Open eHS Real-Time Power Electronics Simulation Toolbox (NI PXIe-7976R)* toolbox solution with the NI PXIe platform.

This section details the software requirements for the system. Development computer (Windows 7 machine) and NI real-time target requirements are listed below. Please keep in mind that this is a default hardware configuration**. For software and hardware installation instructions, please refer to the *Open eHS Real-Time Power Electronics Simulation Toolbox (NI PXIe-7976R) – Getting Started Guide.*

HARDWARE

NI HARDWARE (REAL-TIME TARGET)

- PXI Express Chassis
    - NI PXIe-1082 (or another NI chassis)
- NI PXI Express Controller
    - NI PXIe-8135 RT (or another NI controller)
- FPGA Modules
    - NI PXIe-7976R for eHS operation & Digital Input
    - NI PXIe-7976R for Analog Output
- NI FlexRIO IO Adapter Modules
    - NI 6581B (48-Channel, DIO)
    - NI 5742 (32-Channel, 1 MS/s, 16-Bit Analog Output)
- 1300SMB Breakout Box for High-Speed Digital Devices
- 0.55 m Shielded Cable for High-Speed Digital Devices
- NI myRIO-1900 for external controller (**optional**)

**Note: Other FlexRIO I/O configurations are also supported by the package, but will require a redesign of the firmware and real-time section of the project. In addition, other real-time controllers and PXIe chassis can also be used.

SOFTWARE

DEVELOPMENT SYSTEM SOFTWARE (HOST COMPUTER)

NI Software	Version (minimum)
NI LabVIEW Full or Professional Development System (http://www.ni.com/download/labview-development-system-2016/6046/en/)	2016 (32-bit)
NI LabVIEW Real-Time Module (http://www.ni.com/download/labview-real-time-module-2016/6153/en/)	2016
NI LabVIEW FPGA Module (http://www.ni.com/download/labview-fpga-module-2016/6223/en/)	2016
NI FlexRIO (http://www.ni.com/download/ni-flexrio-16.0/6182/en/)	15.6
NI PXI Platform Services (http://www.ni.com/download/pxi-platform-services-16.0/6195/en/)	16.0
NI CompactRIO Waveform Library (http://www.ni.com/example/31206/en/)	4.1.0.9
NI LabVIEW for myRIO Toolkit (optional) (http://www.ni.com/download/labview-myrio-toolkit-2016/6203/en/)	2016

Circuit Editor Software (at least one of the following)	Version
MATLAB®/Simulink® with SimPowerSystems™ Toolbox	2011b
MATLAB®/Simulink® with PLECS Blockset	3.7
PSIM	9.3
NI Multisim	13

Application Software	Version (minimum)
Open eHS Real-Time Power Electronics Simulation Toolbox (NI PXIe-7976R)	2016.0.1.0

NOTE: If PSIM and\or NI Multisim will be used to edit circuits, the 32-bit version of the MATLAB® Runtime version 9.0 (R2015b) must be installed onto the development machine.

NOTE: It is recommended that users install MATLAB B2011B (32-bit) and\or MATLAB 2015B

NI SOFTWARE (REAL-TIME TARGET)

Before running the application software, the proper drivers must be installed onto the Real-Time target. Follow the steps below to install the necessary drivers. Skip to **Running This Application** if your target already has the proper software installed.

1. In NI-MAX, expand the **Remote Systems** tree and the NI Real-Time PXIe target.
2. Right-click on **Software** and select **Add/Remove Software**.

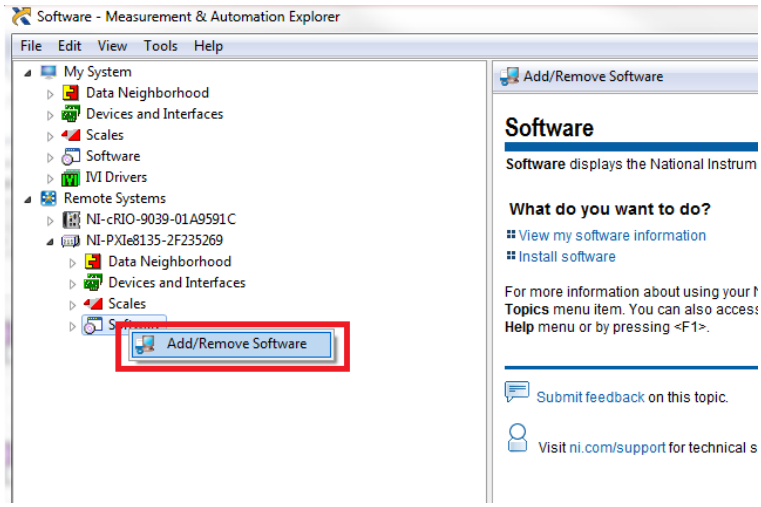


Figure 17: Adding software to the PXIe tatget in NI-MAX

3. Wait for the *LabVIEW Real-Time Software Wizard* to launch.

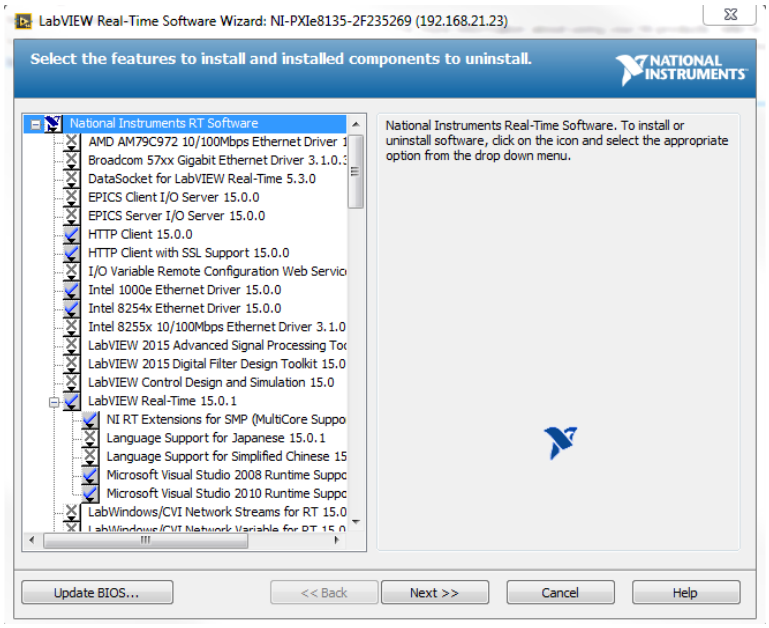


Figure 18: The LabVIEW real-time software wizard

4. In the LabVIEW Real-Time Software Wizard, select the recommended software set as shown in Figure 20 for the NI PXI. For every software component that must be installed, left click on the **X** or **checkmark** symbol and click on **Install this Feature** (see figure below).

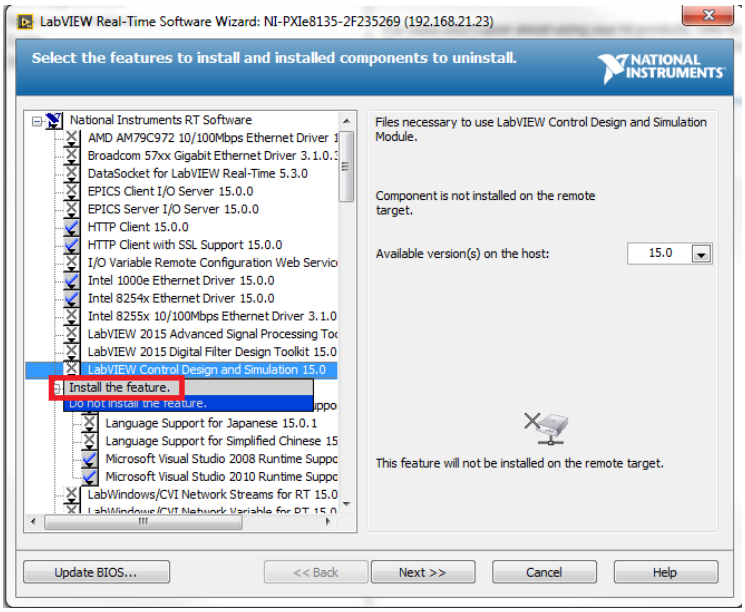


Figure 19: Choosing what software should be installed to the target

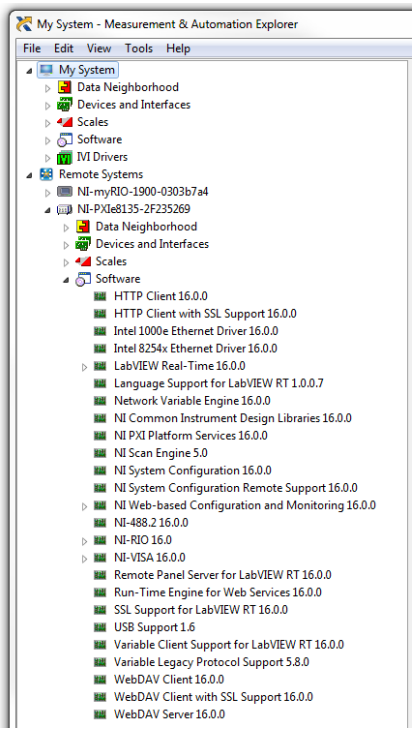


Figure 20: List of required software which must be installed to the PXIe controller

5. Click **Next**.
6. Click **Next** to view a summary of your selection.
7. Click **Next** to start installing the software. The NI PXIe target first boots into install mode and then begins the installation process. When the installation process is complete, the wizard reboots the NI PXIe target.
8. Click **Finish** to close the wizard.

INSTALLING AND OPENING THE PACKAGE

INSTALLING THE EHS PACKAGE

For detailed instructions on how to install the *Open eHS Real-Time Power Electronics Simulation Toolbox (NI PXIe-7976R)*, please refer to the *Open eHS Real-Time Power Electronics Simulation Toolbox (NI PXIe-7976R) - Getting Started Guide* located in the *Documentation* folder within the package (see figure below).

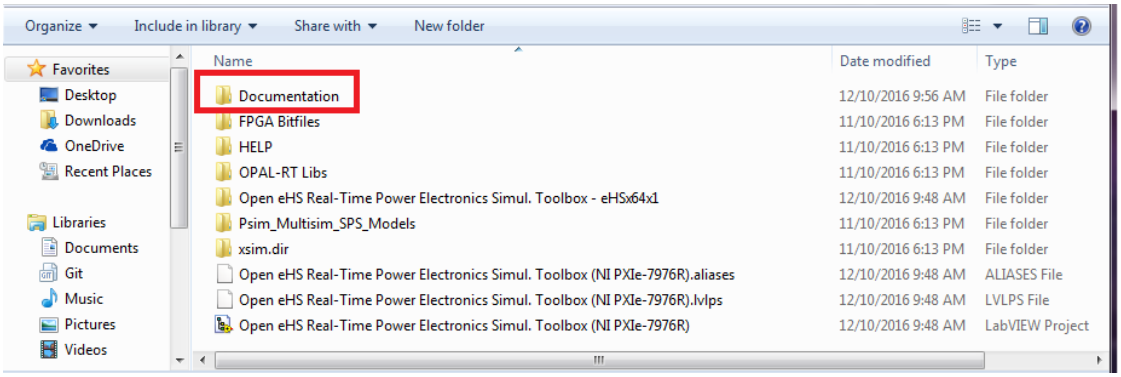


Figure 21: Location of documentation for the package

OPENING THE EHS TEMPLATE LABVIEW PROJECT

After installation, the LabVIEW project can be opened by simply double clicking on the **Show Examples** tab in *VI Package Manager* Please refer to the *Open eHS Real-Time Power Electronics Simulation Toolbox (NI PXIe-7976R) - Getting Started Guide* for installation instructions.

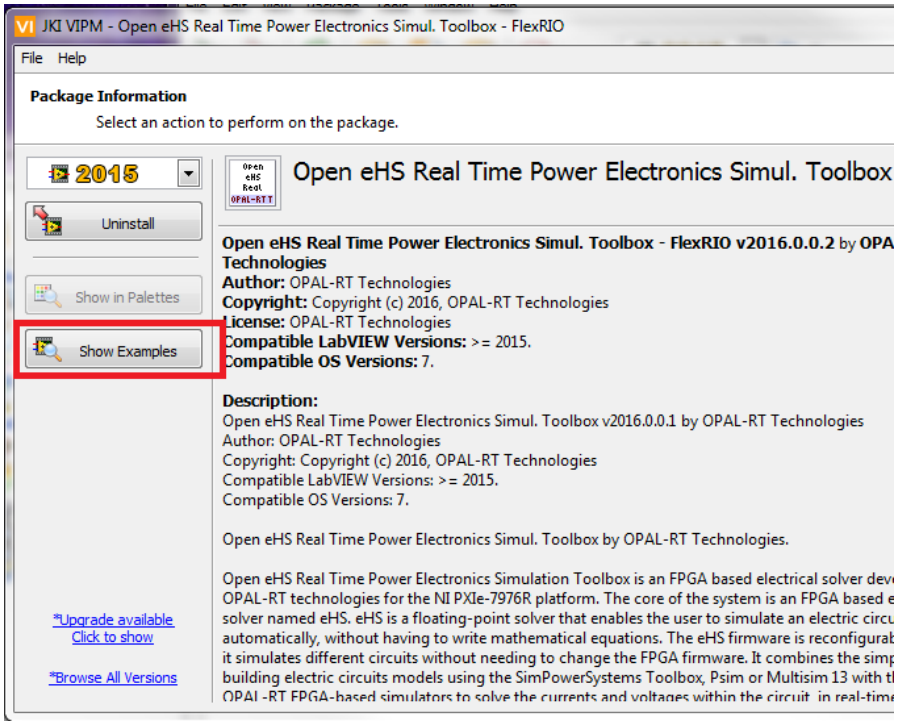


Figure 22: How to open the Open eHS package directory

Clicking ***Show Examples*** should open a Window with the contents depicted din the figure below. It is also possible to manually navigate to the package directory which is installed to the following path: *C:\Program Files (x86)\National*

Instruments\<LabVIEW Version>\examples\Open eHS(7976R) To open the project, simply double click on the **Open eHS Real-Time Power Electronics Simul Toolbox (NI PXIe-7679R).lvproj** item (see figure below).

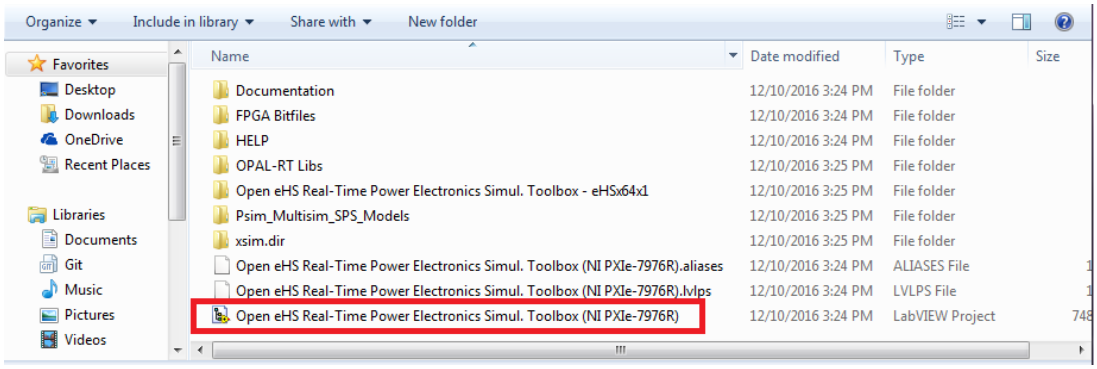


Figure 23: Opening the LabVIEW Project

INSTALLING THE MATLAB® COMPILER RUNTIME (OPTIONAL: PSIM AND MULTISIM ONLY)

Before generating the eHS netlist by running the *Generate & Load eHS Matrix.vi*, install the MATLAB® Compiler Runtime version 9.0 (R2015b). You can download the 32-bit version of the MATLAB® Runtime for R2015b from the MathWorks® website at:

http://www.mathworks.com/products/compiler/mcr/index.html

UPDATING THE NI TARGET'S IP ADDRESS

Right-click on the NI PXI target and select **Properties**. Update the *IP address / DNS Name* field to match the IP address of the target found in NI MAX before running the VIs.

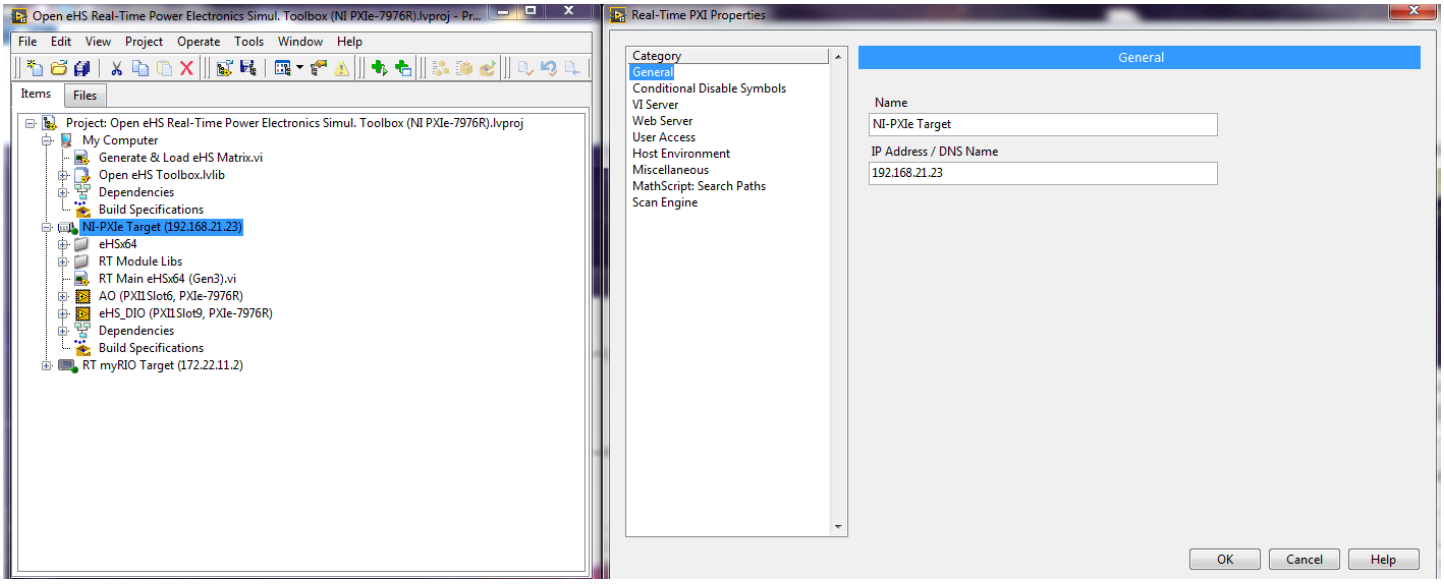


Figure 24: Updating the IP addresses of PXIe controller in the LabVIEW project

FILE AND LABVIEW PROJECT STRUCTURE

WINDOWS FILE STRUCTURE

Opening the *Open eHS Real Time Power Electronics Simul.* Toolbox (NI PXIe-7976R) example folder reveals the following file structure. The figure below details the specific folders that will be discussed in this section.

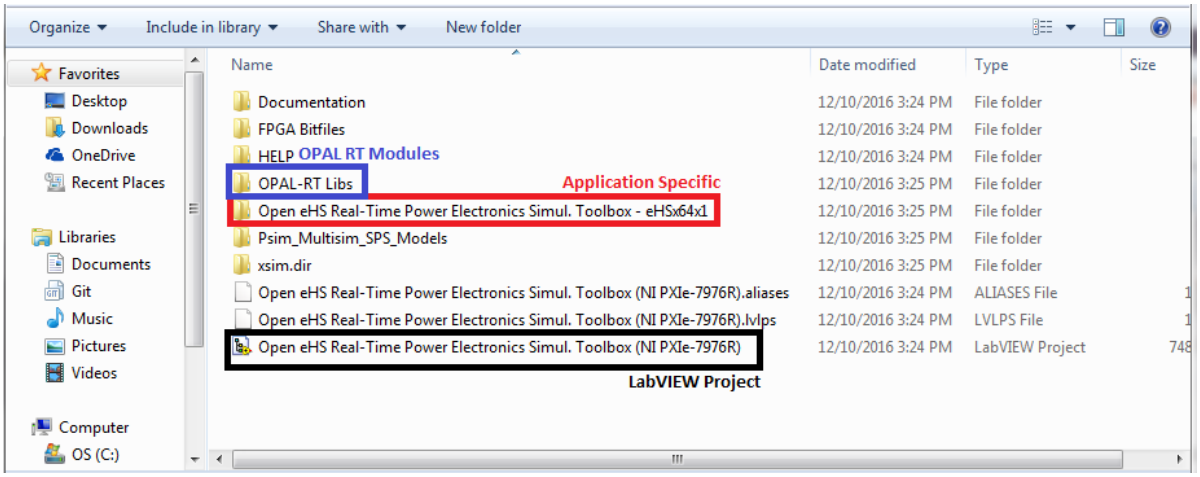


Figure 25: Project file hierarchy

- **LabVIEW Project:** Use this file to open the LabVIEW project (double-click to open).
- **Application Specific:** These folders contain the real-time source code that is specific to this package. The first folder (which ends with *eHSx64x1*) provides support for 1 eHS core.
- **OPAL-RT Modules:** This folder contains the source code for all the OPAL-RT modules used in this package. These include the eHS_DIO, AO, and more.

If you open the *Open eHS Real-Time Power Electronics Simul. Toolbox - eHSx64x1* folder, you will be presented with three items. The *Controls* and *SubVIs* folders contain the type defined controls and subVIs that are used by the toolbox's real-time drivers. The *RT Main eHSx64 (Gen3).vi* is the main VI (i.e. top level VI) for single eHS core simulations.

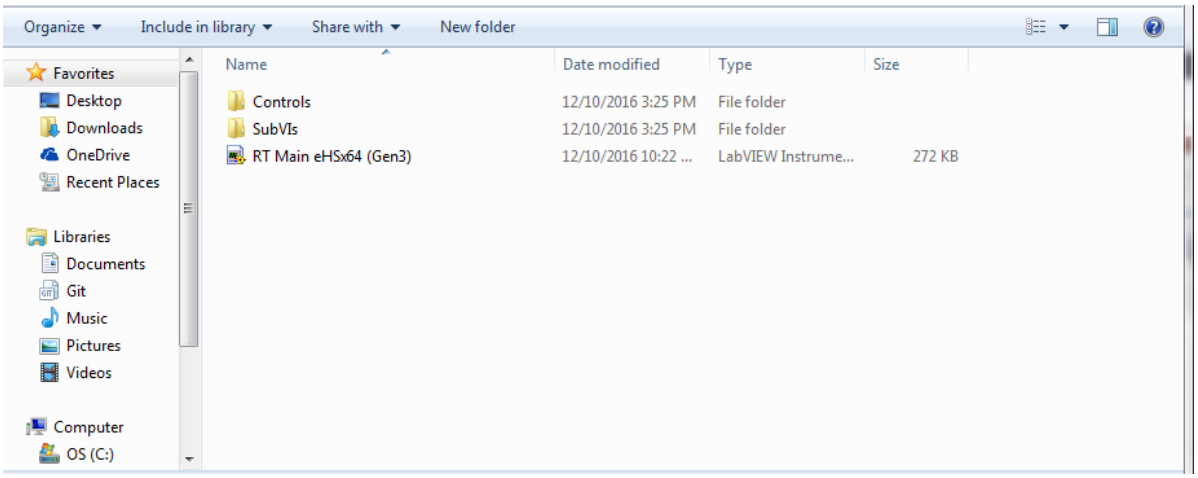


Figure 26: File structure of the application software

Back in the main project directory, if you open the *OPAL RT Libs* folder, you will be presented with several items. The folders represent the specific modules that are used by this package. The last item is a library that contains all the OPAL-RT licensed files.

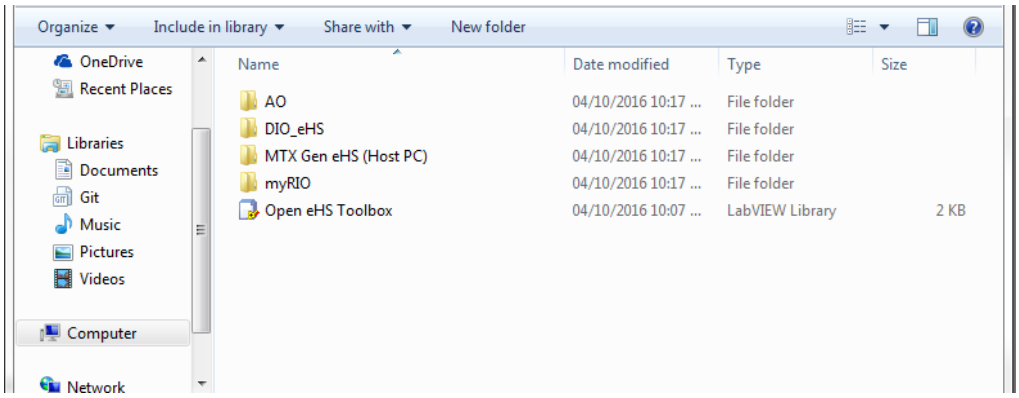


Figure 27: File structure of OPAL-RT modules

- **AO:** This folder contains the real-time and FPGA source code for the analog output portion of the package. It is designed to be used with the PXIe-7976R hardware (with a NI 6851B FAM) but can be modified for use on other targets as well.
- **DIO_eHS:** This folder contains the real-time and FPGA source code for the digital input portion of the package. It also contains the source code that allows the user to program and configure the eHS core.
- **MTX Gen eHS (Host PC):** This folder contains the source code that is used to generate the eHS core netlist. The code in this folder can only be executed on a development machine (Windows 7 OS).
- **myRIO:** This folder contains the source code that is used to program the NI myRIO. The NI myRIO and the contents of this folder are not required by this package. The NI myRIO was only used to perform test sequences and is optional.

The *AO*, *DIO_eHS*, and *myRIO* folders all abide by the same file structure. Each folder contains a folder for the real-time driver and FPFA source code. The source code itself is encapsulated into a real-time driver library and an FPGA library. This allows for easy integration within the LabVIEW project.

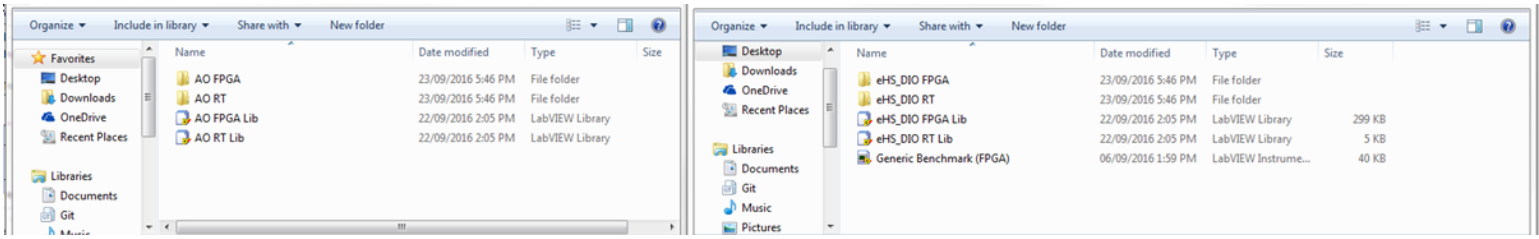


Figure 28: File structure of AO and eHS_DIO

LabVIEW Project Structure

The LabVIEW project is designed to mimic the Windows file structure. The figure below illustrates the Host, Real-Time, and FPGA components of the project. The VIs within each component will be discussed in more detail later.

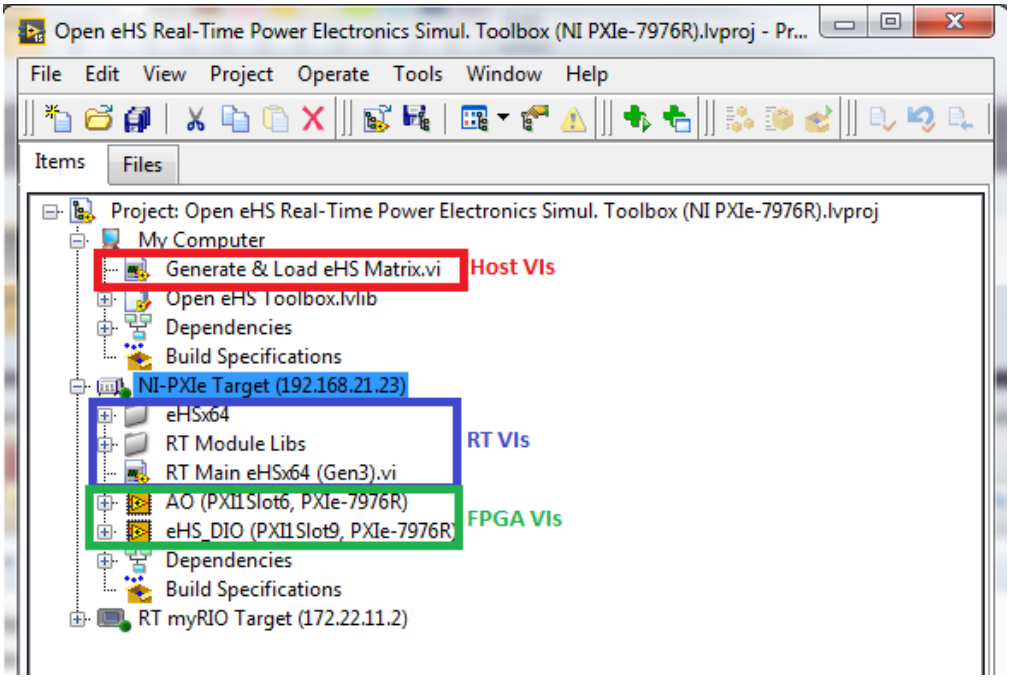


Figure 29: LabVIEW project structure

- The **Host VIs** contain the source code to generate the eHS core netlist(s) that is needed to configure the eHS core (s).
- The **eHSx64** virtual folder contains the real-time source code to support simulations with one eHS core .
- The **RT Module Libs** virtual folder contains the real-time drivers for the DIO_eHS and AO modules
- The **RT Main eHSx64 (Gen3).vi** is the main VI (i.e. top level VI) for single eHS core simulations.
- The **FPGA VIs** section illustrates the different FPGA targets that were added to the project. Each target contains the source code that it requires. Depending on the package being used, different FPGA targets (and configurations can be used.

Expanding the *RT Module Libs* virtual folder will reveal the LabVIEW libraries that contain the real-time drivers for the different modules used by the package. These libraries represent the exact files that were previously discussed in figures 27 & 28.

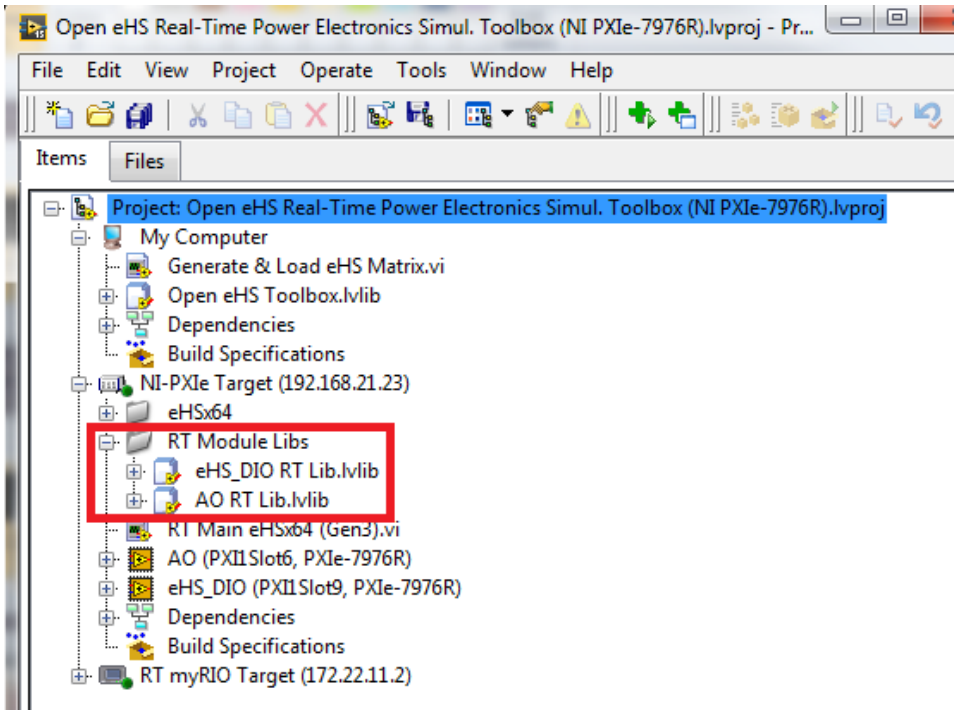


Figure 30: RT modules within LabVIEW project

Expanding any one of the FPGA items in the project will reveal a LabVIEW libraries that contains the FPGA source code for the DIO_eHS and AO FPGAs. These libraries also represent the exact files that were previously discussed in figure 27&28 above.

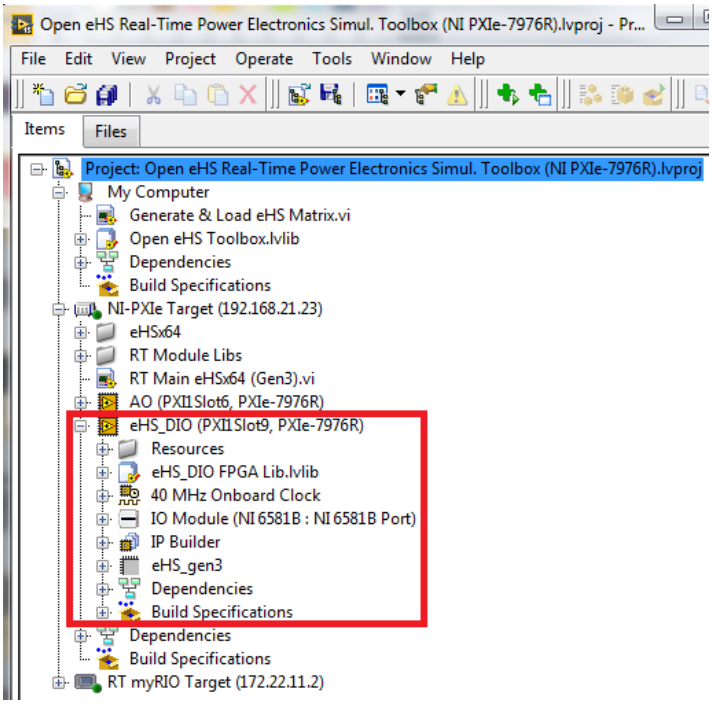


Figure 31: FPGA modules within LabVIEW project

THE EHS GEN3 FPGA (EHS_DIO) FIRMWARE AND RT DRIVER

The provided project files support the use of one eHS core.

ONE EHS CORE

The figure below shows the block diagram of the template FPGA firmware provided with the package. To design your own FPGA firmware, it is recommended to use the provided FPGA firmware as a starting point to accelerate development. Before starting the tutorials, open and explore the *eHS_DIO Firmware.vi* located under the eHS_DIO FPGA target within the *eHS_DIO FPGA Lib.lvlib* library (Please see Appendix 4 if you notice that the *eHS_DIO Firmware.vi* is broken). It includes the following features:

- eHS Gen3 IP core
- 3 FPGA-based sine wave generators
- eHS input mapping
- 48 channels of digital inputs via NI6581B
- SPWM generator for 3-phase 2-level inverter (6 SPWM signals)
- 2 channels of PWM generator (2 signals per channel)
- Mapping of internal PWM inputs
- Mapping of eHS outputs
- Scale and bias of eHS outputs
- 32 channels of analog outputs using NI 5742
- DAQ FIFO to monitor the eHS outputs on the host PC.

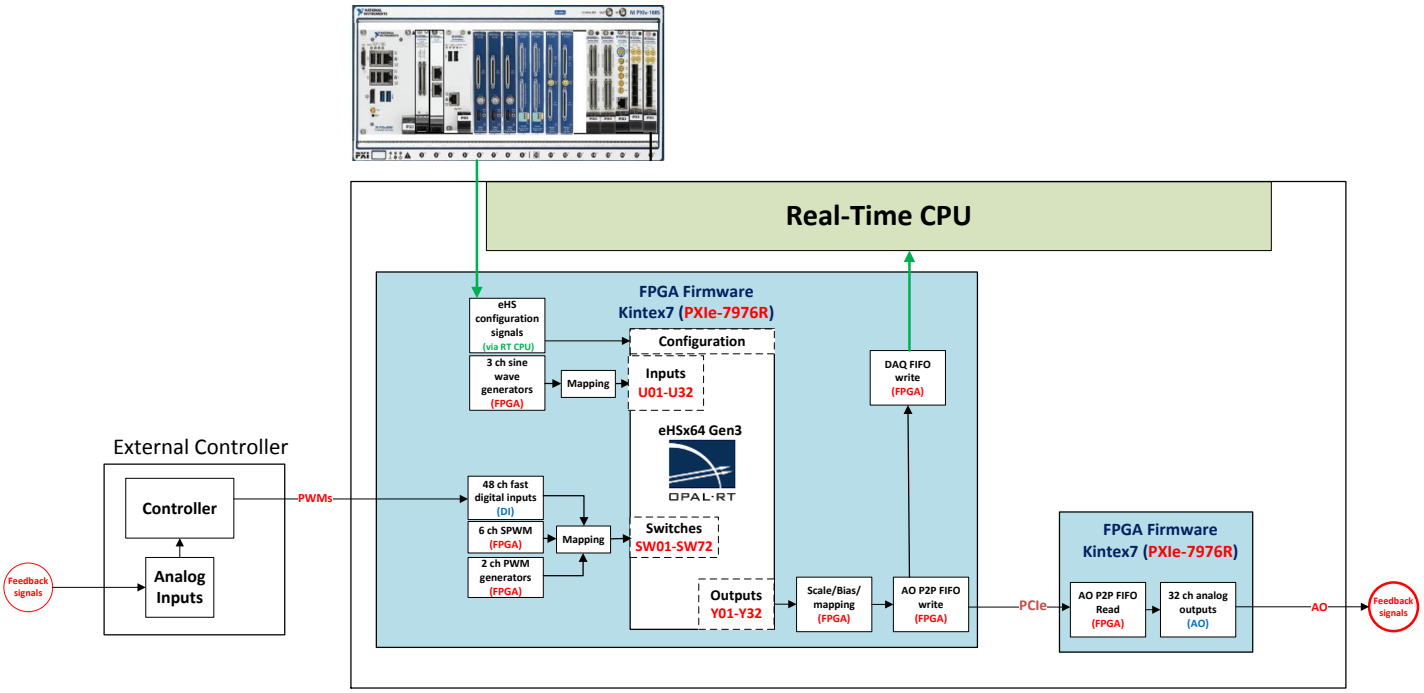


Figure 32: eHS Gen3 FPGA Firmware (1 eHS Core)

The figure below shows the block diagram of the real-time process (*RT Main eHSx64 (Gen3).vi*) that initializes, configures, and monitors the FPGA firmware of all FPGAs that are used in this configuration.

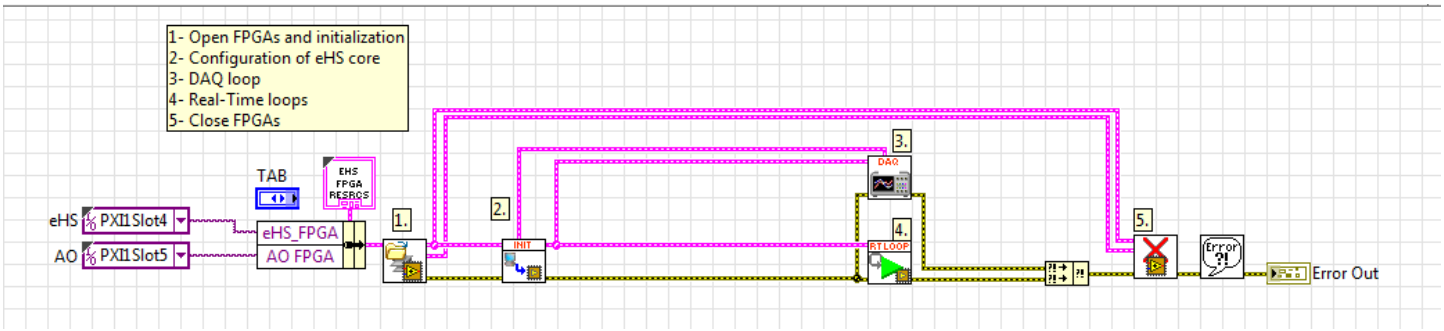


Figure 33: Block diagram of real time process (1 eHS Core)

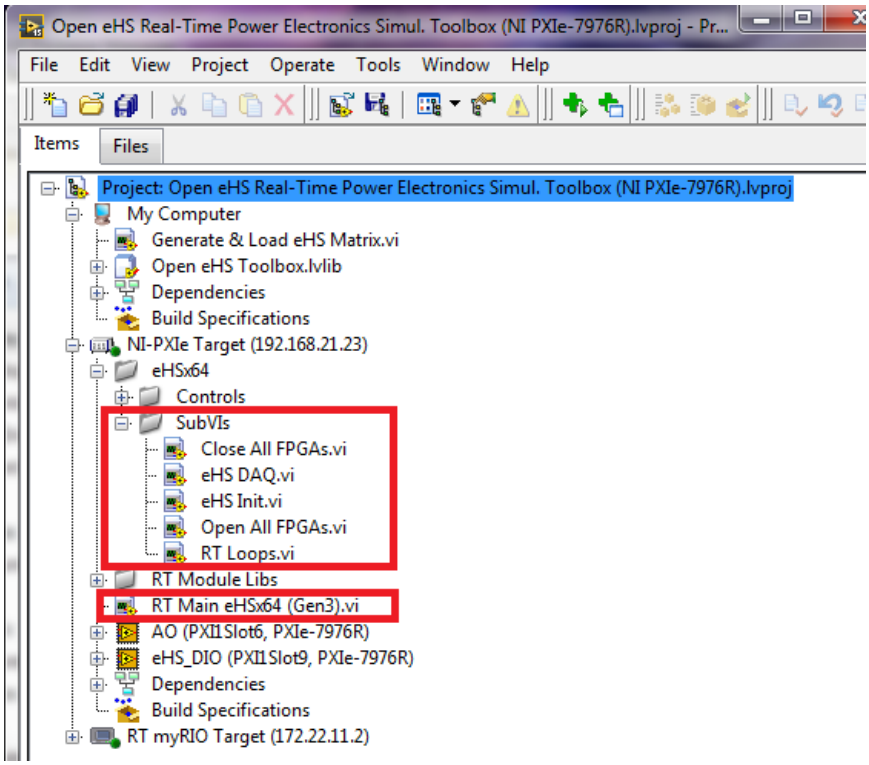


Figure 34: Location of application level VIs for 1 eHS core simulations

During the real time simulation, using the *eHS DAQ*.vi front panel (see figure above to find this VI in the LabVIEW project), one can monitor the outputs of the eHS core (Y01 to Y32). Also, using the *RT loops.vi* front panel, one can update the parameters of the FPGA firmware as seen in the figures below.

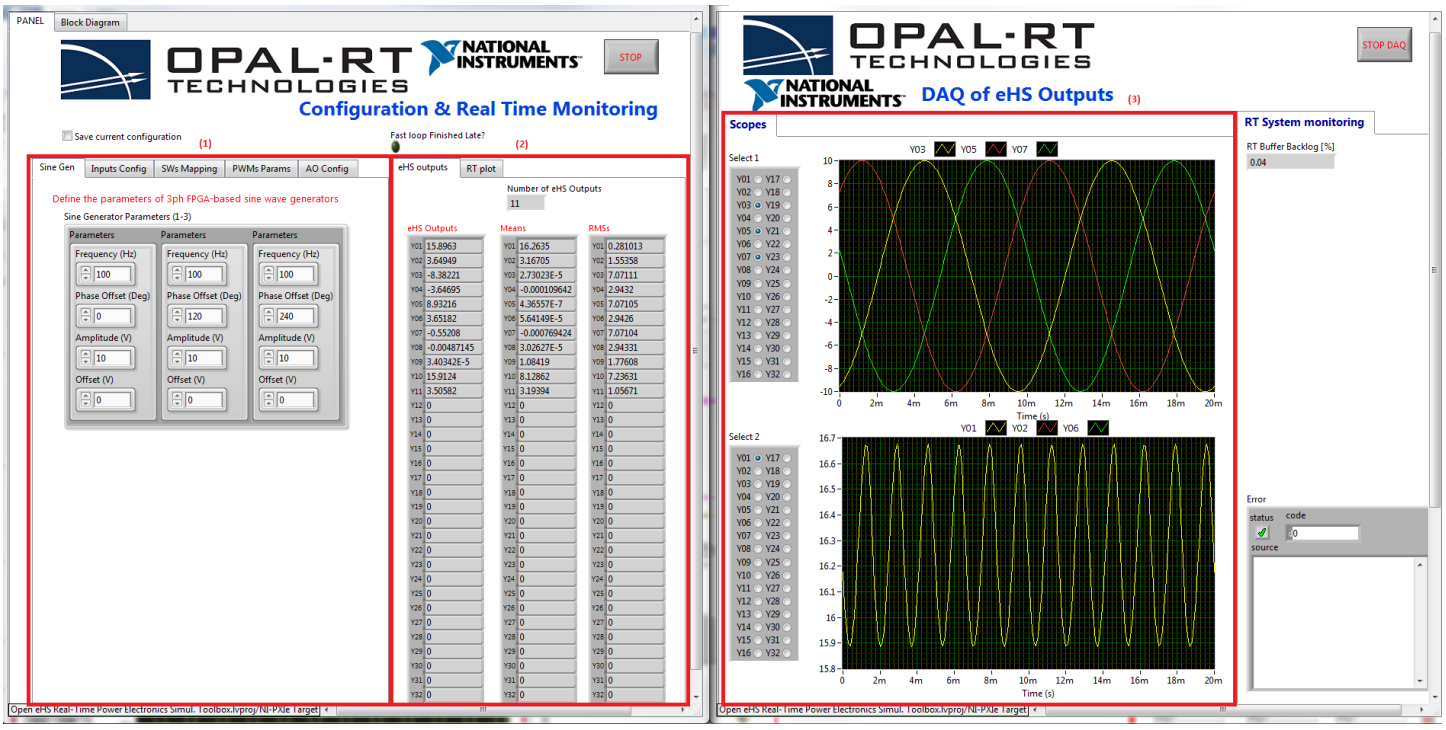
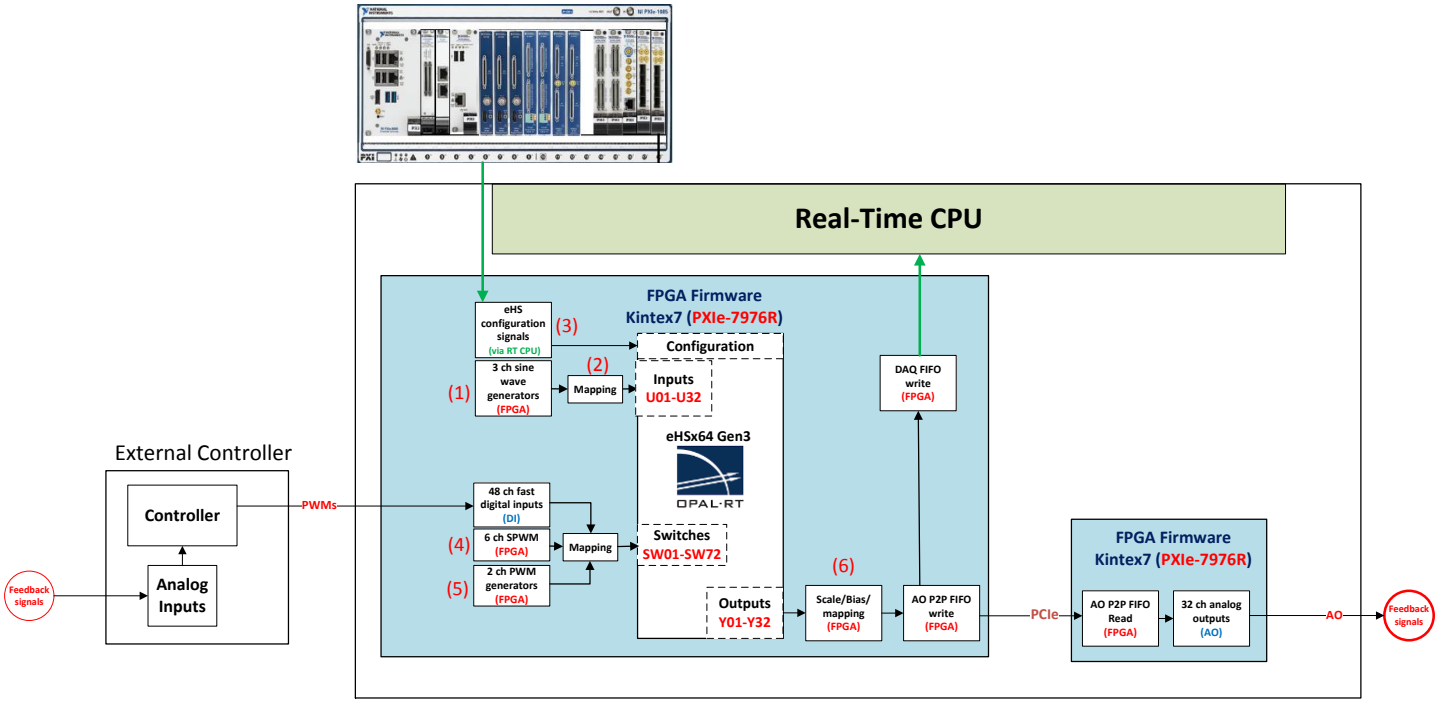


Figure 35: Panel of real time loops and DAQ loop: (1)- Updating FPGA Firmware parameters (2)- Monitoring real time, RMS and mean values of eHS outputs (3)- Monitoring the eHS outputs in DAQ scopes

The figure below explains in detail the user interface connectivity between the FPGA firmware parameters and the related tabs in the real-time front panel. Notice that there are essentially three different types of tabs: internal signal generation tabs, eHS input mapping, and eHS output mapping\scaling. When a model is being simulated on the target, the user will have to use these panels to correctly route all signals to the eHS core.



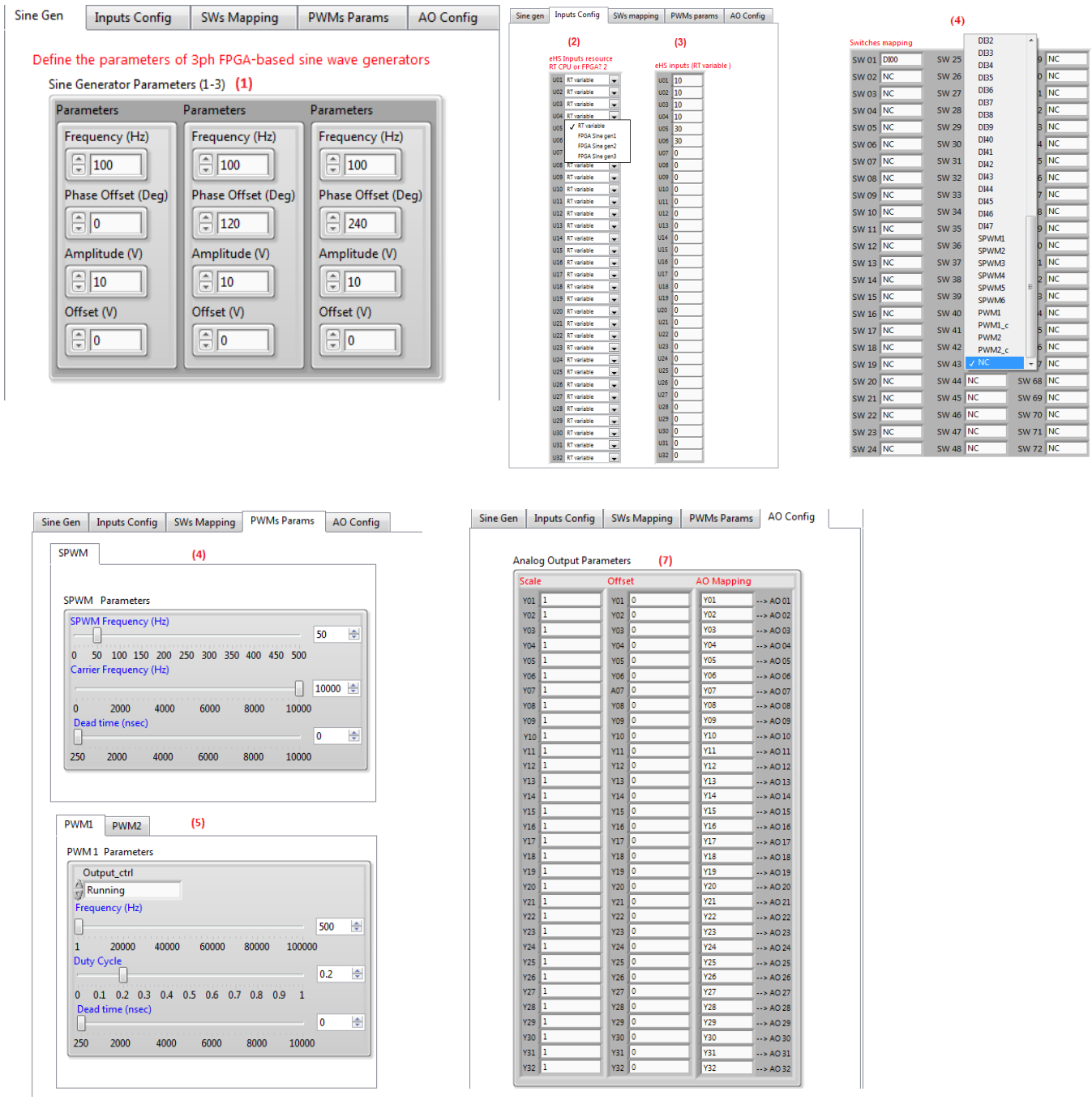


Figure 36: Connectivity between FPGA firmware parameters and related tabs in RT panel

TUTORIAL 1: THREE-PHASE RECTIFIER EXAMPLE

This tutorial explains how to generate eHS Gen3 matrices, configure the eHS core, and run the real-time simulation for a three-phase rectifier on the NI PXIe platform.

Step 1: Before starting the real-time simulation, navigate to the *PSIM_Multisim_SPS_Models* directory and use one of supported circuit editors (MATLAB®/Simulink® with the SimPowerSystems™ Toolbox, the PLECS Blockset, PSIM, or NI Multisim v13) to open and explore one of the following models:

- Rectifier3ph_Psim.psimsch
- Rectifier3ph_SPS.mdl
- Rectifier3ph_PLECS.mdl
- Rectifier3ph.ms13.

In this configuration, *U01, U02* and *U03* are the eHS input sources, and *Y01* to *Y11* are the measurement points (eHS outputs). This circuit also has six diode switches, *SW01* to *SW06*. (See figure below**Error! Reference source not found.**).

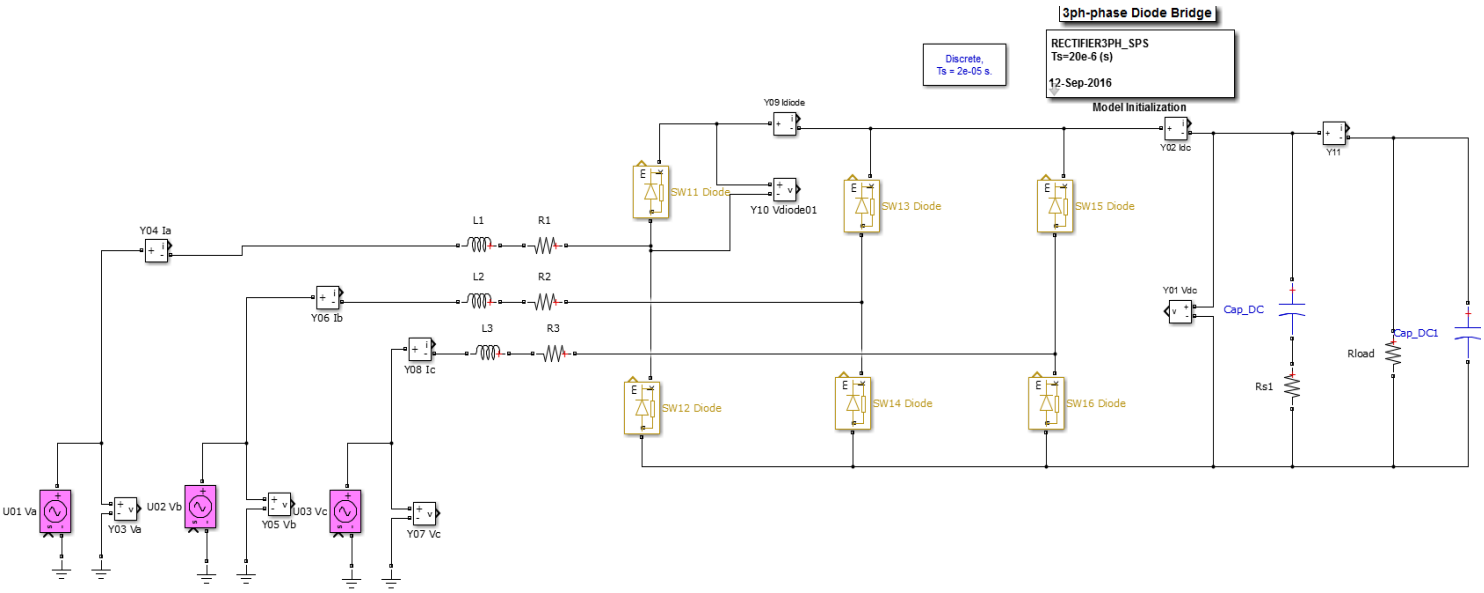


Figure 37: SPS circuit for the three-phase rectifier example

Step 2: Open the *Open eHS Real-Time Power Electronics Simulation Toolbox (NI PXIe-7976R)* LabVIEW project and the *Generate & Load eHS Matrix.vi* (See figure below).

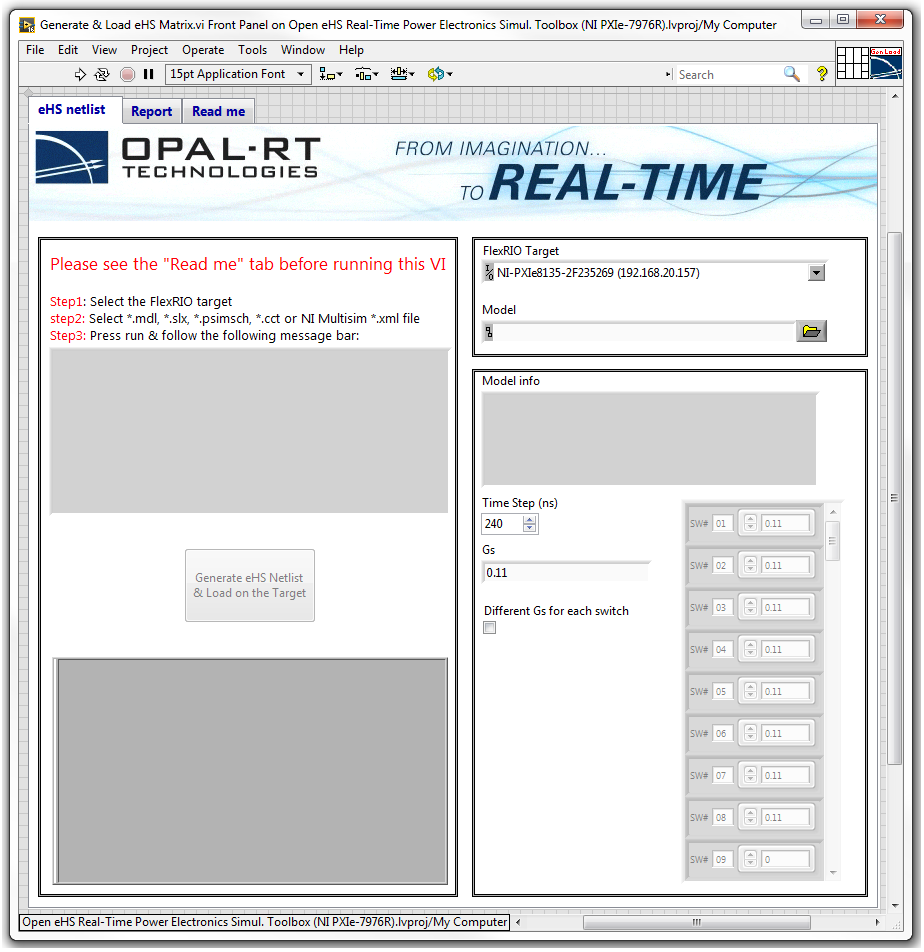


Figure 38: Front panel of *Generate & Load eHS Matrix.vi*

Step 3: On the *Generate & Load eHS Matrix.vi* front panel, select the PXI target from the drop-down menu. Then select either *Rectifier3ph_Psim.psimsch, Rectifier3ph_SPS.mdl, Rectifier3ph_PLECS.mdl, or Rectifier3ph.xml* from the *Psim_Multisim_SPS_Models* directory for the *eHS1 Model* file path. To load a second model on to the PXIe hardware for use on a second eHS core, the user could choose to select a model for the *eHS2 Model* input (this functionality might not be supported by your version of the package).

Note that to generate the eHS matrix, the users must have:

- a licensed PSIM software installed on their PC to use the *.psimsch model;
- a licensed MATLAB®/Simulink® with SimPowerSystems™ Toolbox installed on their PC to use SimPowerSystems (*.mdl or *.slx) models;
- a licensed MATLAB®/Simulink® with PLECS Blockset Toolbox installed on their PC to use PLECS (*.mdl or *.slx) models;
- a licensed NI Multisim 13 capable of generating a netlist in an XML format (refer to Appendix 2 for more detail).

Run the VI and follow the instructions detailed on the front panel (Step 1 to Step 4) to generate the eHS matrix and load it onto the NI PXIe target. Note that the eHS solver sample time is accessed through the *Time Step (ns)* parameter. This value depends on the complexity of the circuit and typically ranges from 150 nanoseconds to 2.5 microsecond. The Gs

parameter is the conductance of the switches of the circuit. One can modify the time step to be equal or larger than the minimum time step. Consult the message panel and make sure that this VI operation was successful.

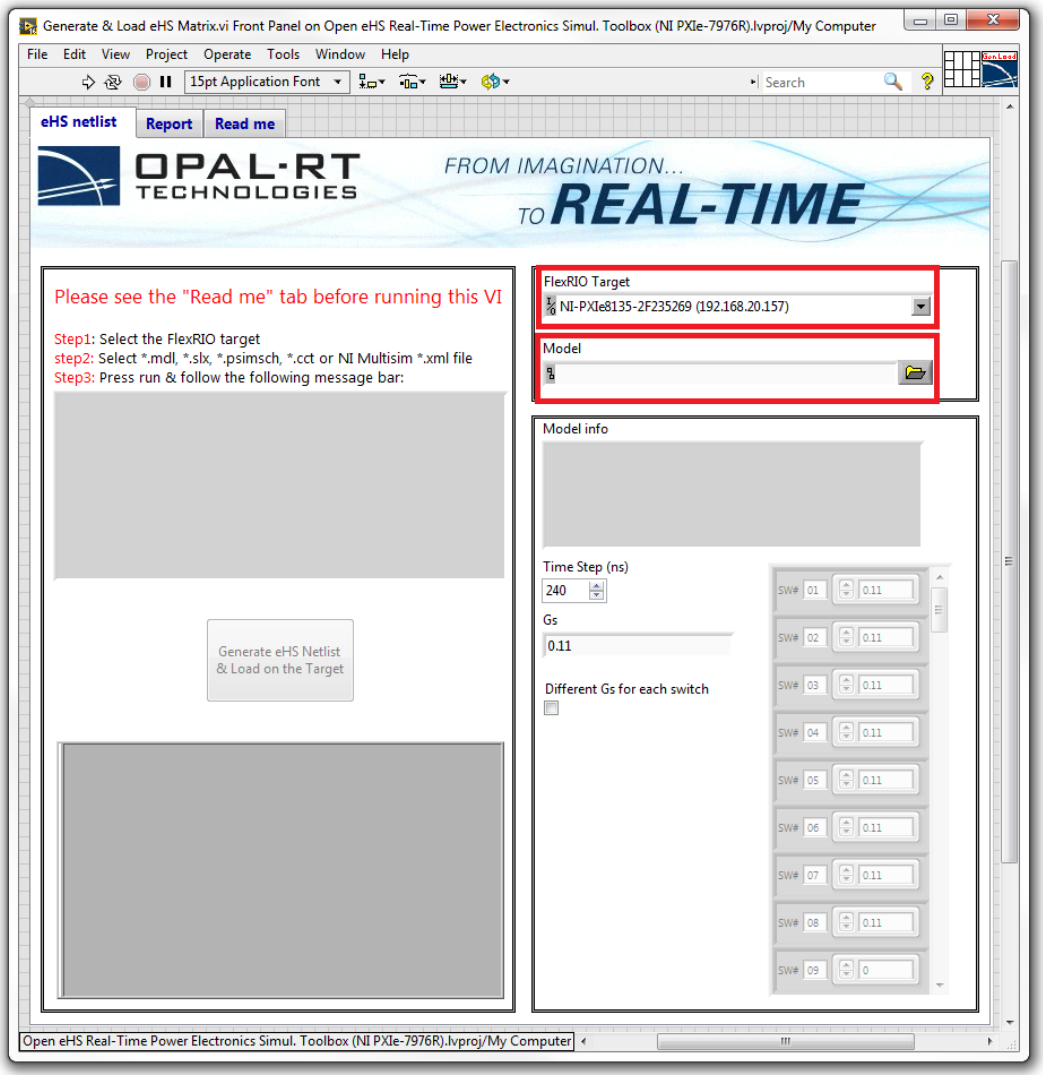


Figure 39: Generating eHS matrix with proper time step

After successful execution of this VI, consult the *eHS1 Report* tab to see detailed information about the eHS1 Gen3 Solver netlist (See figure below).

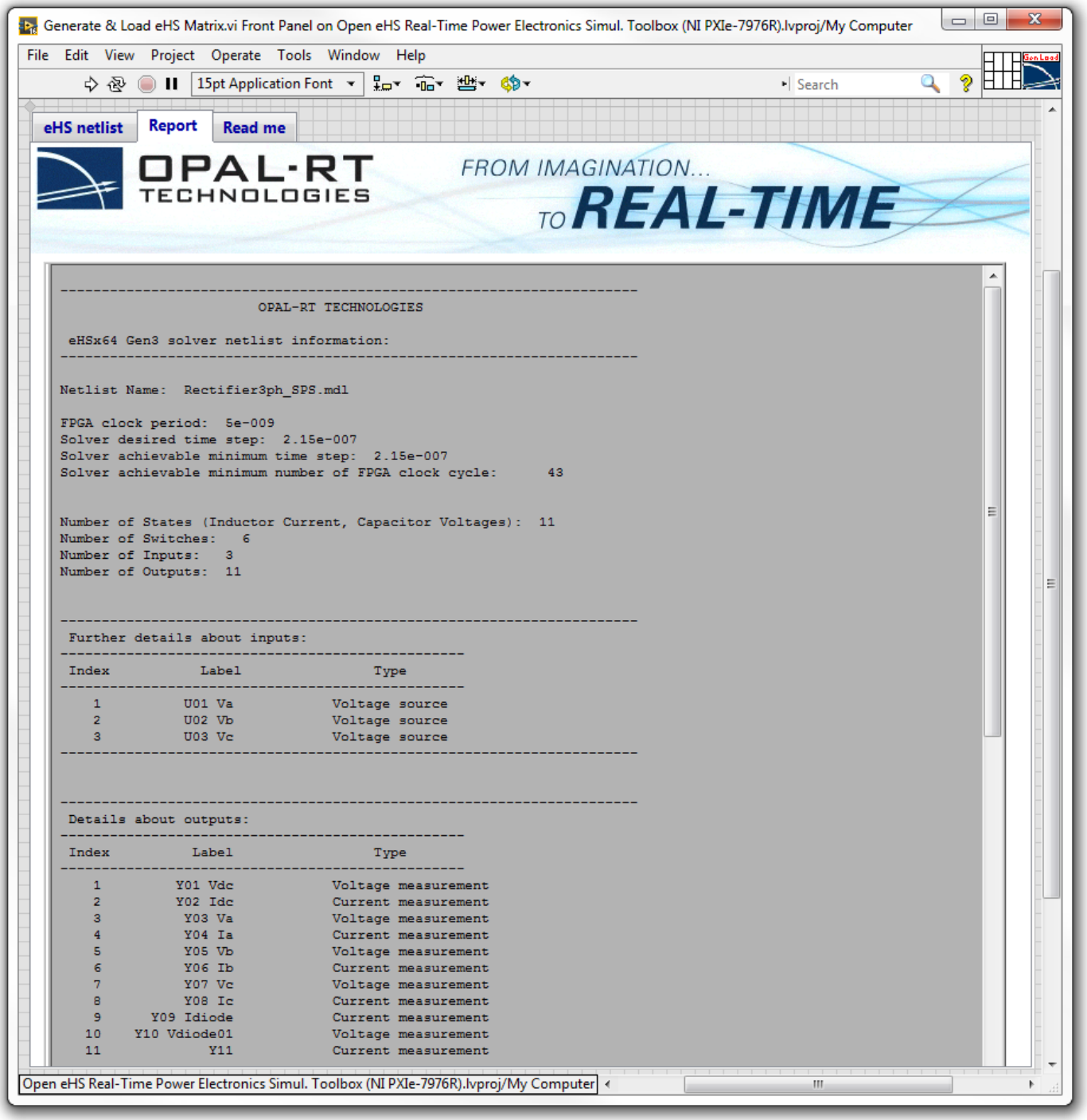


Figure 40: Report tab shows detailed information about the model

Step 4: Open the *RT Main eHSx64 (Gen3).vi* and press **Run** on the front panel. By running this VI, the *eHS DAQ.vi* and *RT Loops.vi* front panels will appear on the screen.

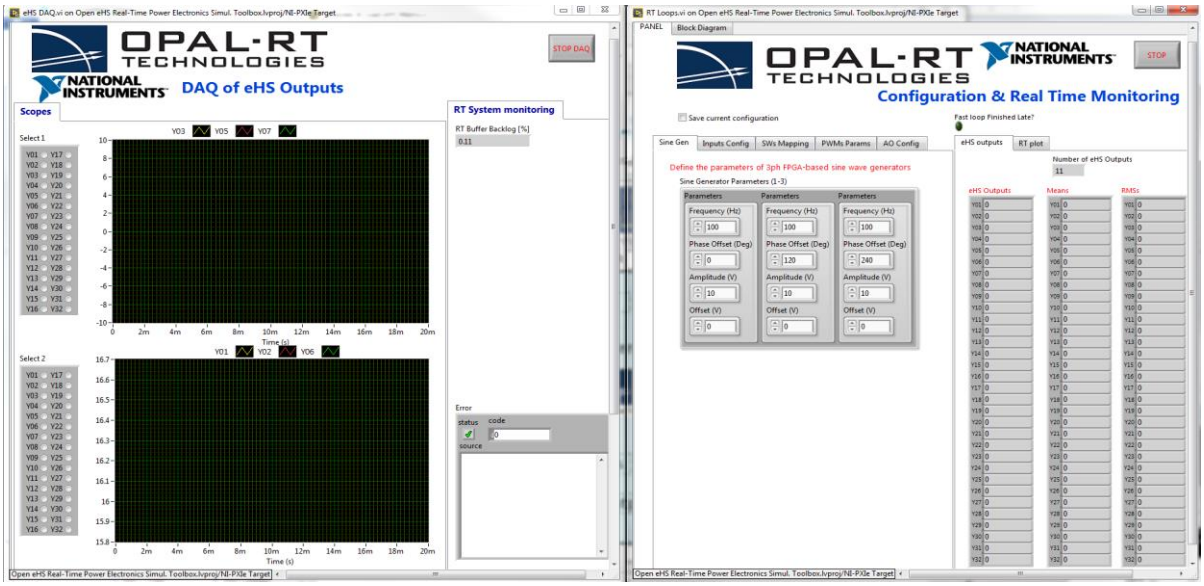


Figure 41: eHS DAQ & RT Loops VIs after running the RT process VI.

From the *Inputs Config* tab, select the eHS Input Resource U01 to be **FPGA Sine gen1**, U02 to be **FPGA Sine gen2**, and U03 to be **FPGA Sine gen3**. (See figure below)

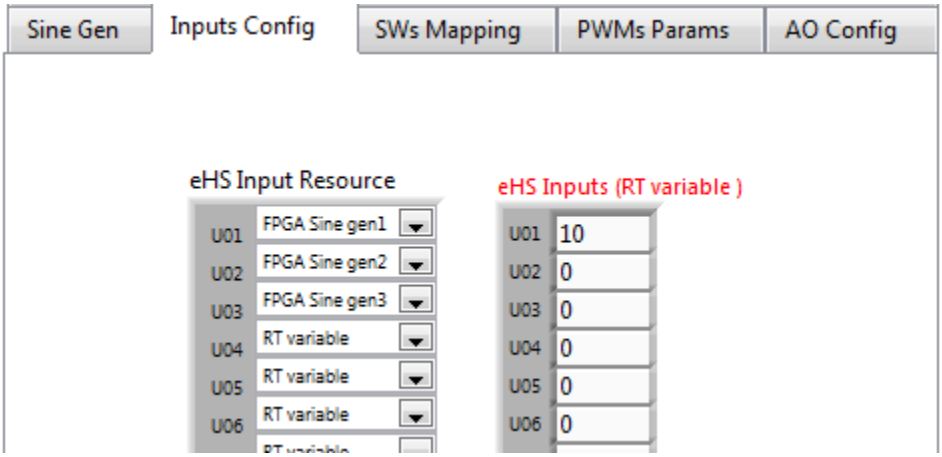


Figure 42: Mapping U01 to FPGA Sine gen1, U02 to FPGA Sine gen2 and U03 to FPGA Sine gen3

In the *eHS DAQ*.vi front panel for the upper Waveform Chart, select **Y03**, **Y05**, and **Y07** and for the lower Chart select **Y09** and **Y10**. Refer to the schematics to see which point you are monitoring on the *eHS DAQ.vi* front panel.

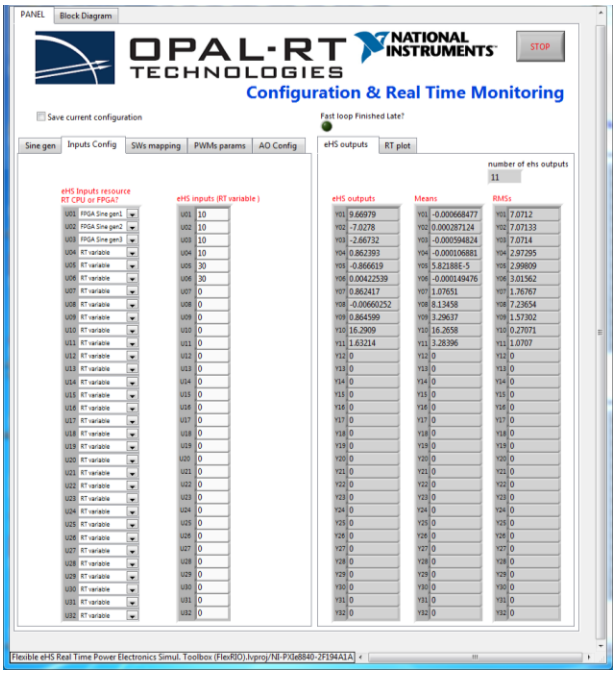


Figure 43: RT Loops and eHS DAQ front panels after mapping eHS inputs to the signal generators

Step 5: In the **Sine gen** tab, change the frequency and amplitude of the sine wave generators which are connected to U01, U02 and U03 and observe the effect on the Waveform Graphs.

Step 6: You can route each channel of the eHS outputs (Y01 to Y32) to each channel of the analog outputs (AO0 to AO32). Select the **AO Config** tab, click on the mapping drop-down menu for AO1 and choose **Y9**. Now you have connected AO1 to Y9. You can monitor the analog outputs using an external oscilloscope. Note that the NI 5742 features 32 channels of 16-bit analog output for 5 V full-scale range. In other words, you must tune the scale and offset parameters to scale the eHS outputs in the range of 0 to 2^{16}-1 (65535) which corresponds to 0 to 5 volt analog output values. Monitoring the negative part of the output signals is done by setting the *Offset* parameter.

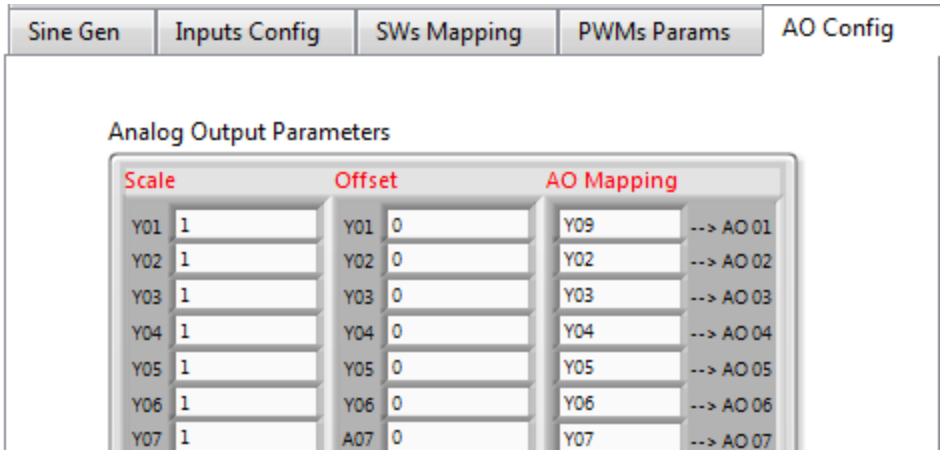


Figure 44: Mapping of eHS outputs to analog outputs and tuning scale and offset parameters

Step 7: Press the **Stop** button on the *RT Loops.vi* front panel. Navigate to the *RT Loops.vi* block diagram by pressing Ctrl+E on the keyboard. This block diagram involves six main subVIs: (see figure below)

1. Open FPGA and initialization
2. Configuration of eHS core
3. Data acquisition (DAQ) loop
4. Real-time loops
5. Close FPGAs

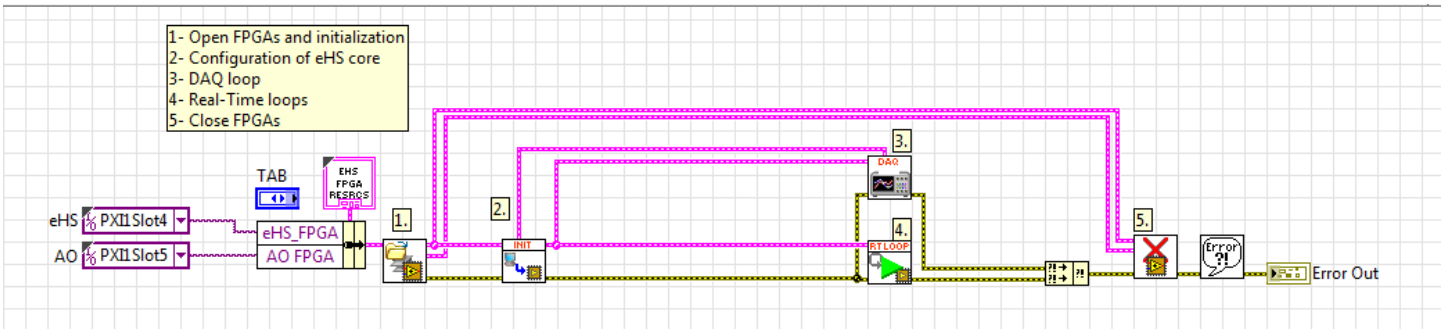


Figure 45: LabVIEW block diagram of *RT Process.vi*

Step 8: Double-click on the *eHS Init.vi* to open it and modify the legends of the *eHS DAQ.vi* graphs. In the *Channel Name* column, change *Y01* to *sine ph1*, *Y02* to *sine ph2*, and *Y03* to *sine ph3*. Save and close the VI, then run the *RT Main eHSx64 (Gen3).vi*. Now, instead of seeing *Y01, Y02***,** and *Y03*, you can use the appropriate names for your measurement points: *sine ph1*, *sine ph2*, and *sine ph3*. (See figures below).

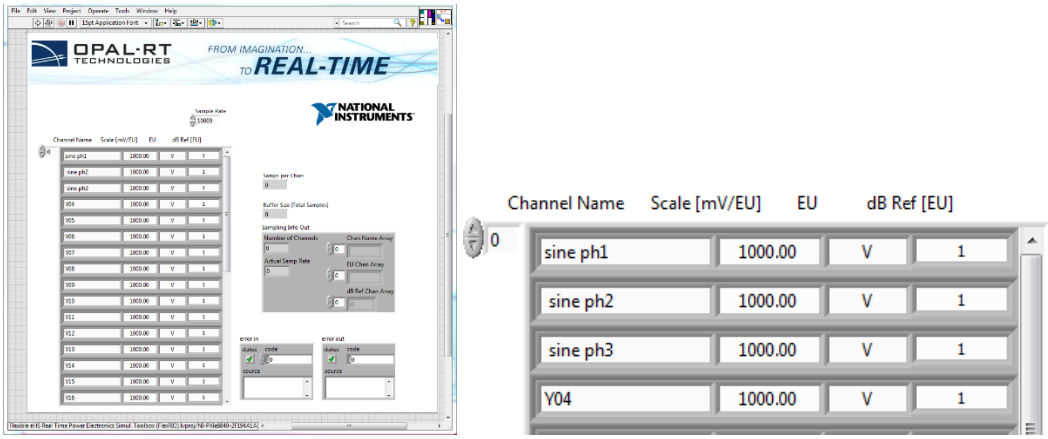


Figure 46: Modifying the name and legend of the eHS outputs for the DAQ graphs

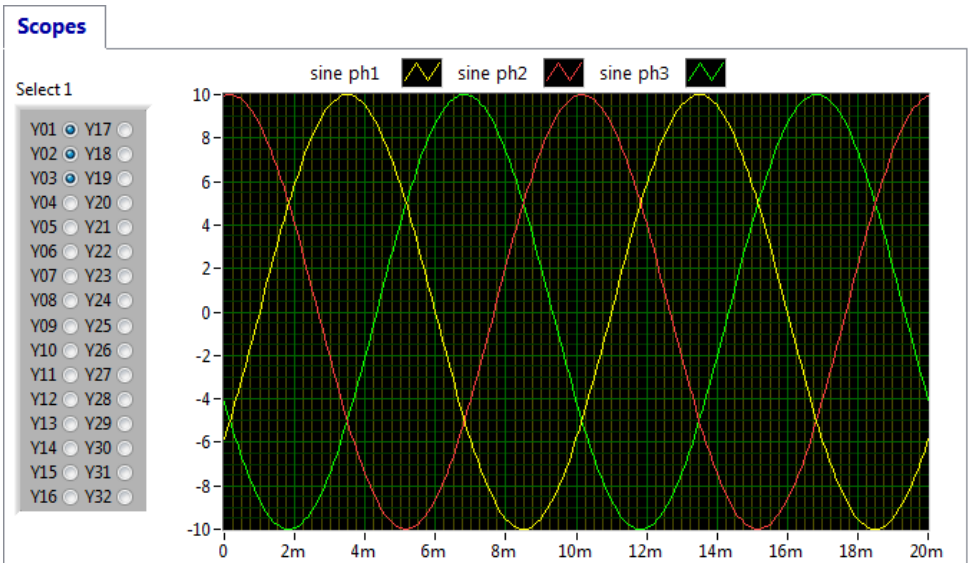


Figure 47: Proper legend for eHS DAQ graphs

Step 9: Press the **Stop** button on the front panel of the *RT Loops.vi*. Navigate to the *RT Loops.vi* block diagram by pressing Ctrl+E. There are two individual parallel real-time loops. The upper loop is for configuration and mapping. The lower loop is used to communicate (read/write) with the eHS (see figure below).

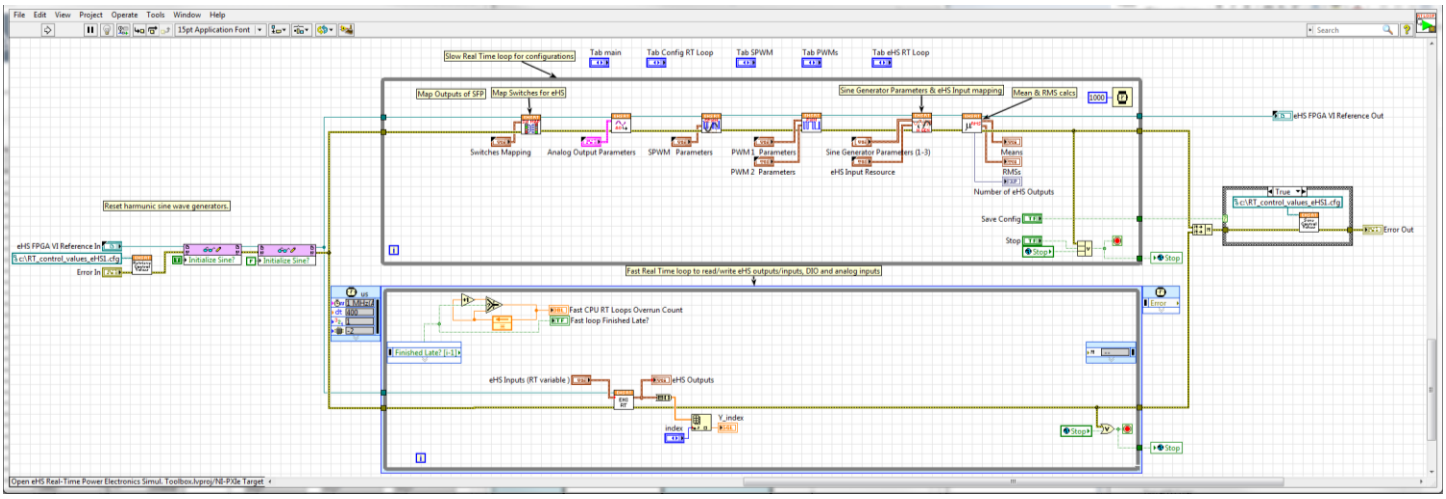


Figure 48: LabVIEW block diagram of the *RT Loops VI*

Step 10: In the block diagram, press Ctrl+H to open the *Context Help* window and then hover the mouse over each subVI to see a short description. You can also refer to Appendix 1 for a description of all subVIs in the OPAL-RT libraries (See figure below).

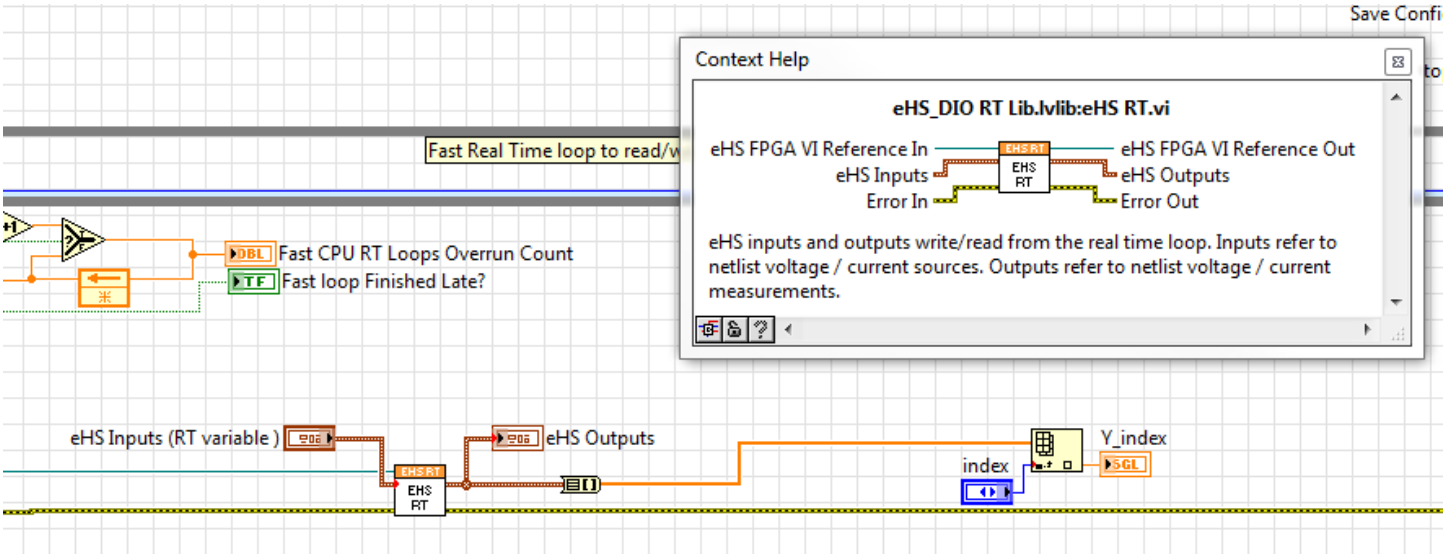


Figure 49: *Context Help* description for the *eHS RT VI*

TUTORIAL 2: HOW TO USE AN EXTERNAL CONTROLLER

This tutorial explains how to run the real-time simulation of a boost converter with an internal PWM and an external PWM signal. It also explains how to change the load and configuration of a boost converter during its real-time simulation. Finally, it explains how to use an external controller in a closed-loop control configuration.

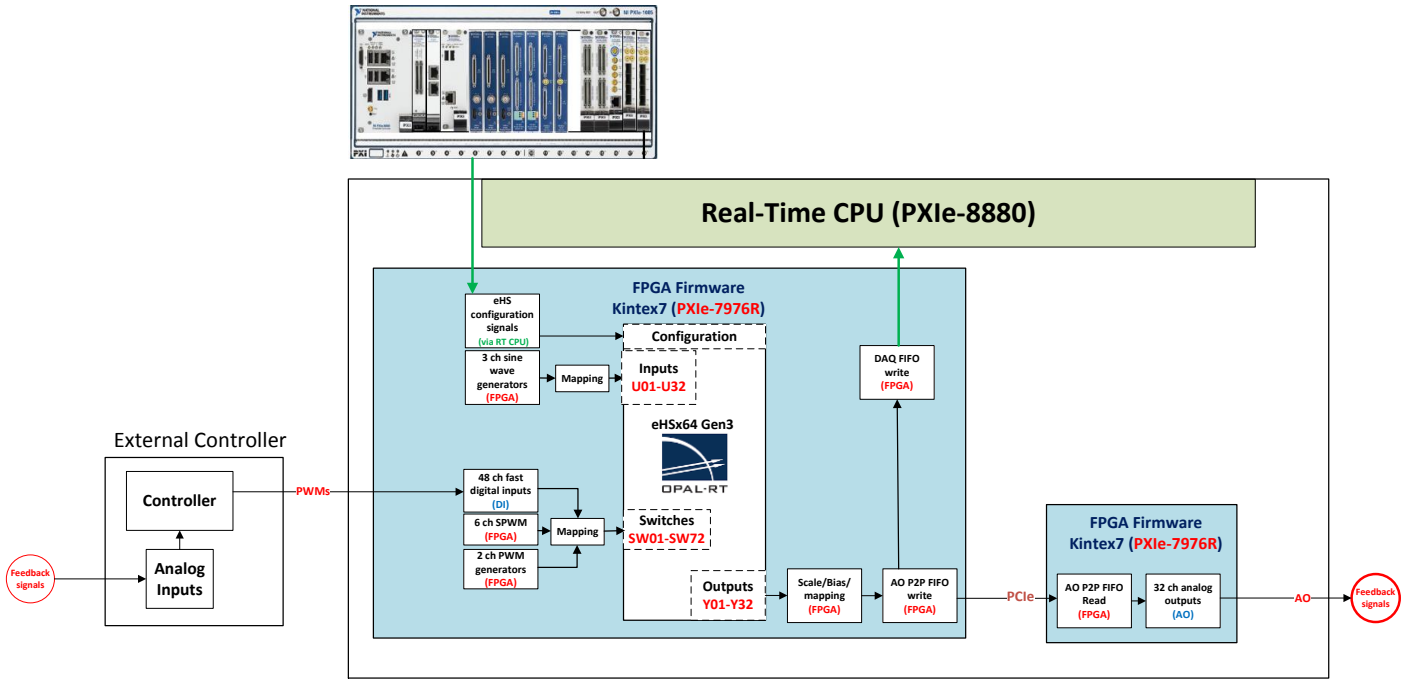


Figure 50: Hardware in loop (HIL) and external controller for Boost tutorial

Step 1: Navigate to the *PSIM_Multisim_SPS_Models* directory and use one of supported circuit editors (MATLAB®/Simulink® with SimPowerSystems™ Toolbox, PLECS Blockset, PSIM or NI Multisim v13) to open and explore one of the following models:

- Boost_Psim.psimsch
- Boost_SPS.mdl
- Boost_PLECS.mdl
- Boost.ms13.

In this configuration, **U01** is the input voltage source of the eHS and **Y01** to **Y04** are the measurement points (outputs of the eHS). This circuit also contains one controlled switch **SW01** and one diode switch **SW02**. In this tutorial, you will learn how to drive the gate of **SW01** from internal and external PWM resource (see figure below).

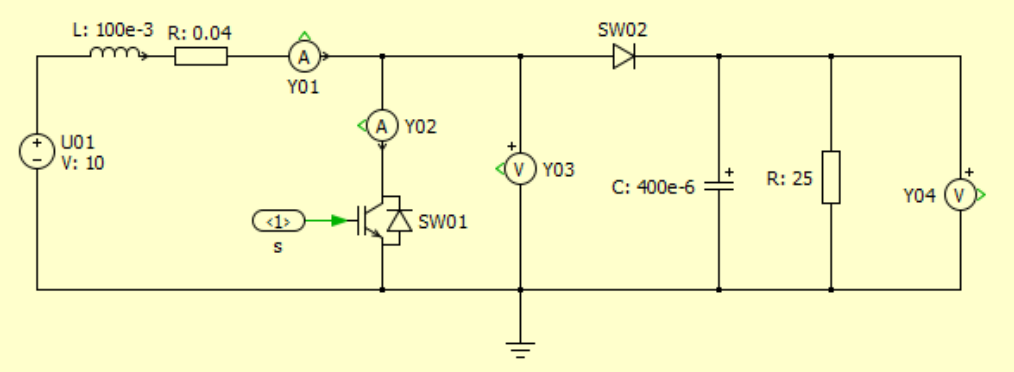


Figure 51: Boost converter for real-time simulation used in tutorial 2

Step 2: On the *Generate & Load eHS Matrix.vi* front panel, select the NI PXI target. Then, select either *Boost_Psim.psimsch, Boost_SPS.mdl, Boost_PLECS.mdl, or Boost.xml* from the *Psim_Multisim_SPS_Models* directory for the *eHS1 Model* file path. To load a second model on to the PXIe hardware for use on a second eHS core, the user could choose to select a model for the *eHS2 Model* input.

Note that to generate the eHS matrix, the user must have one of the following:

- a licensed PSIM software installed on their PC to use the *.psimsch model;
- a licensed MATLAB®/Simulink® with SimPowerSystems™ Toolbox installed on their PC to use SimPowerSystems (*.mdl or *.slx) models;
- a licensed MATLAB®/Simulink® with PLECS Blockset Toolbox installed on their PC to use PLECS (*.mdl or *.slx) models;
- a licensed NI Multisim 13 capable of generating a netlist in an XML format (refer to Apendix 2 for more detail).

Run the VI and follow the instructions detailed on the front panel (Step 1 to Step 4) to generate the eHS matrix and to load it on the PXI target. Note that the eHS solver sample time is accessed through the *Time Step* parameter. This value depends on the complexity of the circuit, and typically ranges from 100 nanoseconds to one microsecond. The Gs parameter is the conductance of the switches of the circuit. One can modify the Time Step to be equal to or more than the minimum time step. Consult the message panel and make sure that this VI operation was successful.

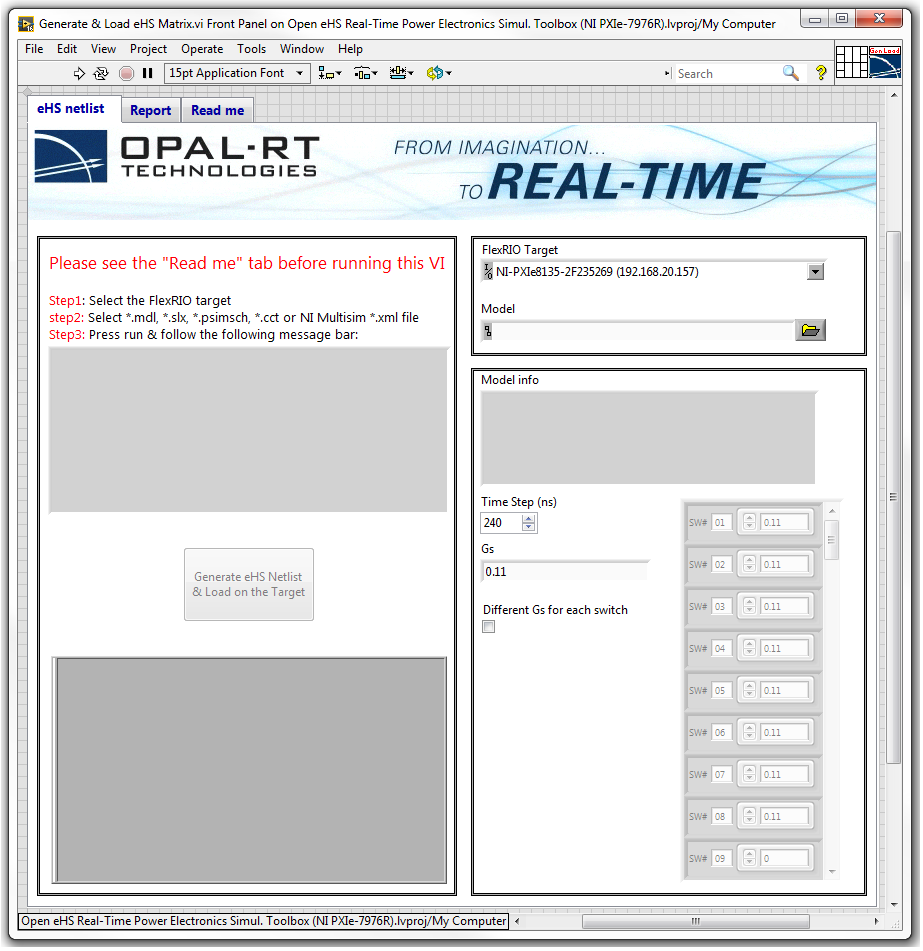


Figure 52: Generating eHS matrix and loading in the FlexRIO target for Boost tutorial

Step 3: Open the *RT Main eHSx64 (Gen3).vi* and press **Run** on the front panel. By running this VI, the *eHS DAQ.vi* and *RT Loops.vi* front panels will appear on the screen.

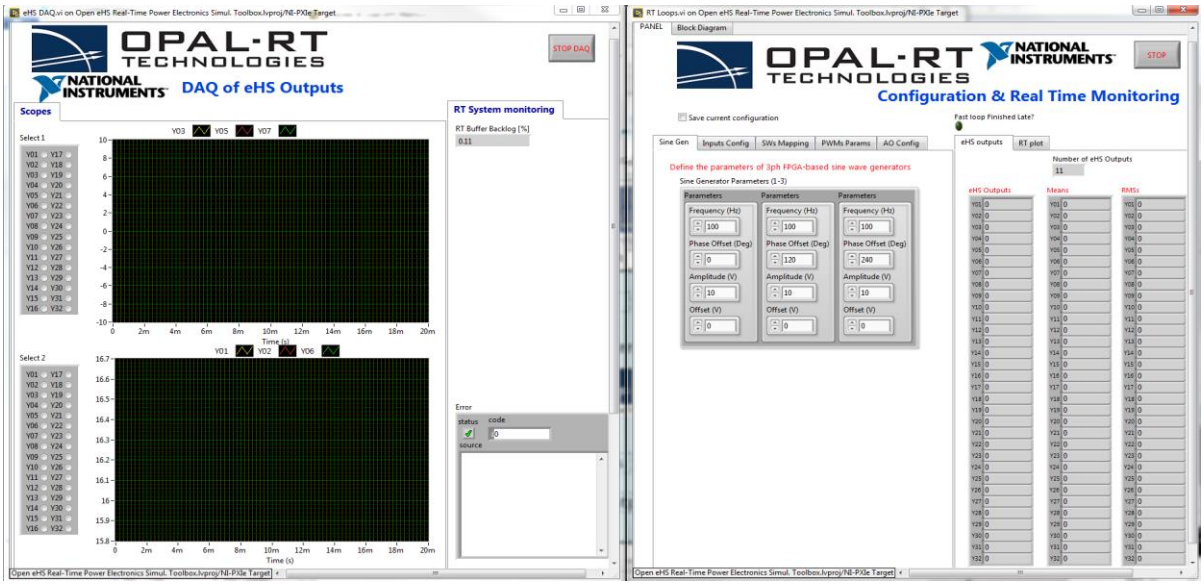


Figure 53: RT loops and DAQ panel after running the RT process VI.

In the **Inputs Config** tab, select **RT Variable** for *U01* and then modify eHS input *U01* to be **100** volts (See figure belowFigure 54).

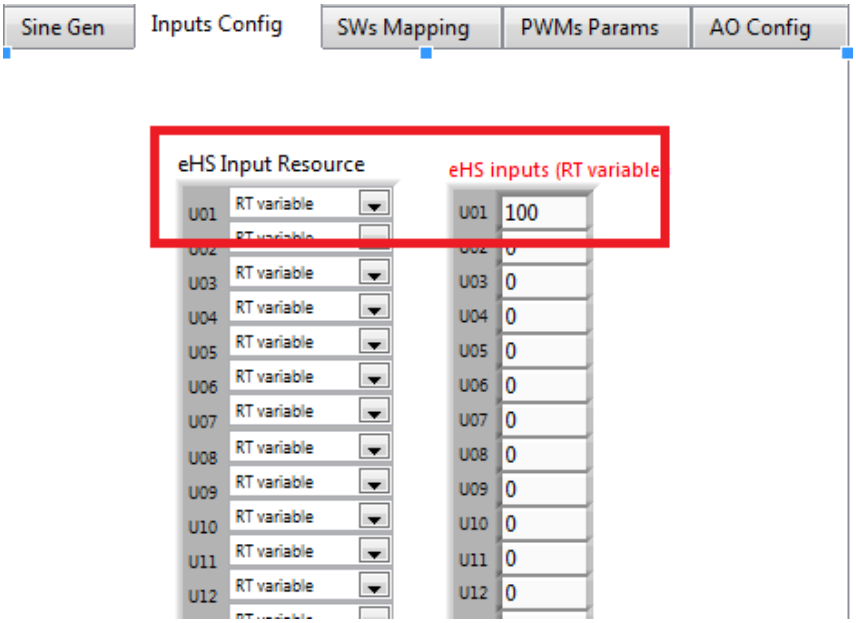


Figure 54: eHS inputs mapping for Boost convertor.

Step 4: To route an internal PWM signal to SW01, click on the **SWs Mapping** tab and select **PWM1** for *SW01* as seen in the figure below. This should route the interval PWM signal to *SW01* (See figure below).

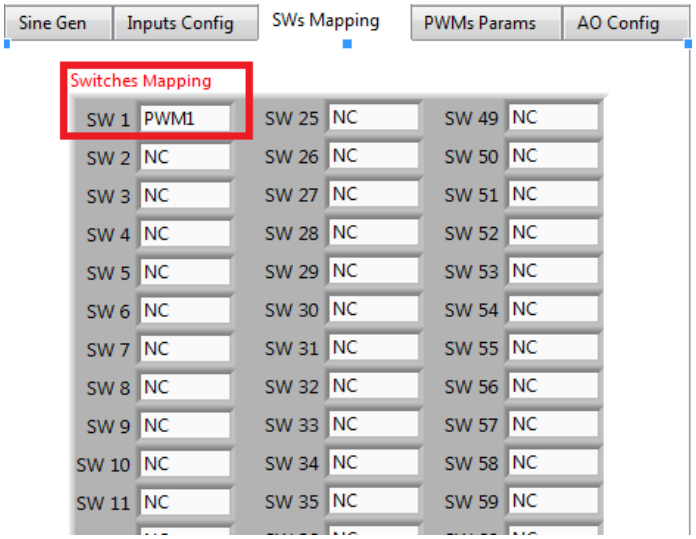


Figure 55: Routing PWM1 to SW01

From the *PWMs Params* tab, change the *Frequency* and *Duty Cycle* of the internal PWM signal and see the effect on the *eHS DAQ.vi* front panel for Y01 to Y04. Consult the schematic of the Boost converter to see which voltage or current measurements Y01 to Y04 refer to. Set the *PWM1* parameters as detailed in the figure below.

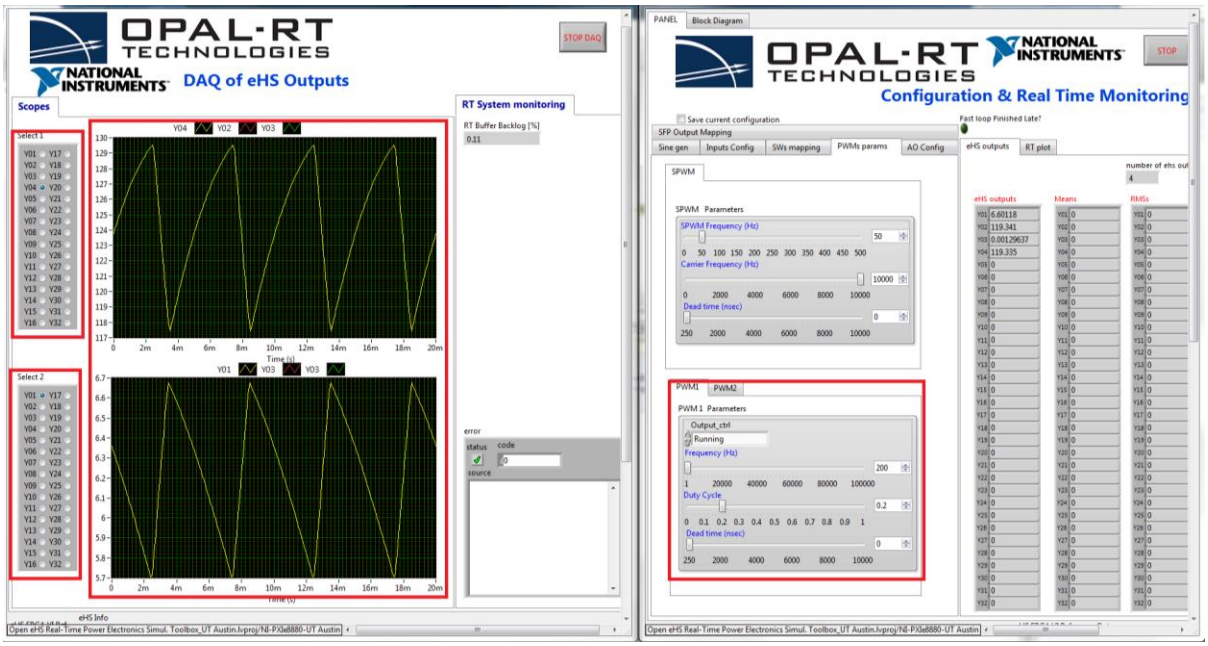


Figure 56: Updating PWM1 parameters and monitoring the voltage and current measurements of Boost convertor

Step 5: External PWM for Boost convertor:

Physical wiring required for this step:

- *Connect Port B/PWM0 on the myRIO to the DDCB P0.0 port of the PXIe-6581B FlexRIO Adapter Module (Slot 4 in PXIe chassis).*
- *Ensure that the myRIO and FlexRIO Adapter Module are sharing a common ground. By default, the eHS firmware is configured to accept 3.3VTTL logic on the DDCB connector which is the voltage level supported by the myRIO.*

On the front panel of the *RT Loops.vi*, click on the **SWs Mapping** tab and select **DIO24** for *SW01* as seen in the figure below. This should route the external PWM signal connected to *DIO24 (*on the PXIe-6581B FlexRIO Adapter Module) to *SW01*.

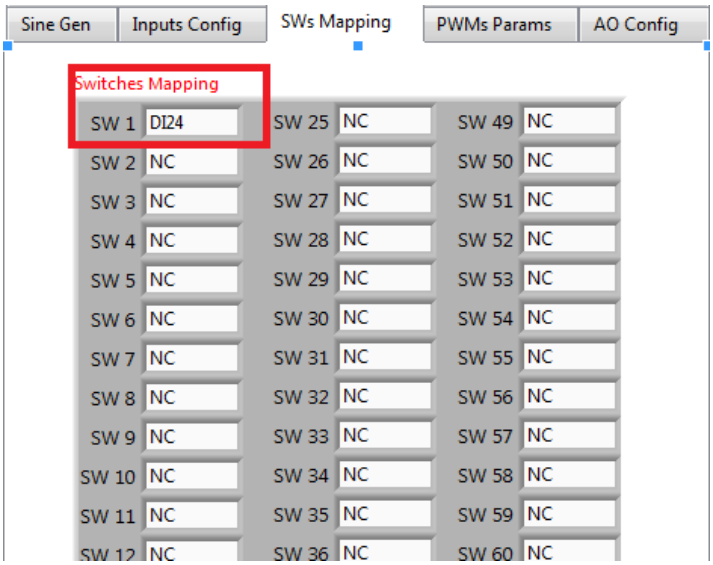


Figure 57: Switch mapping

Step 6: Open *myRIO_DIO_sw.vi* from the LabVIEW project (see figure below). Run the *myRIO_DIO_sw.vi* by clicking on the **Run** arrow.

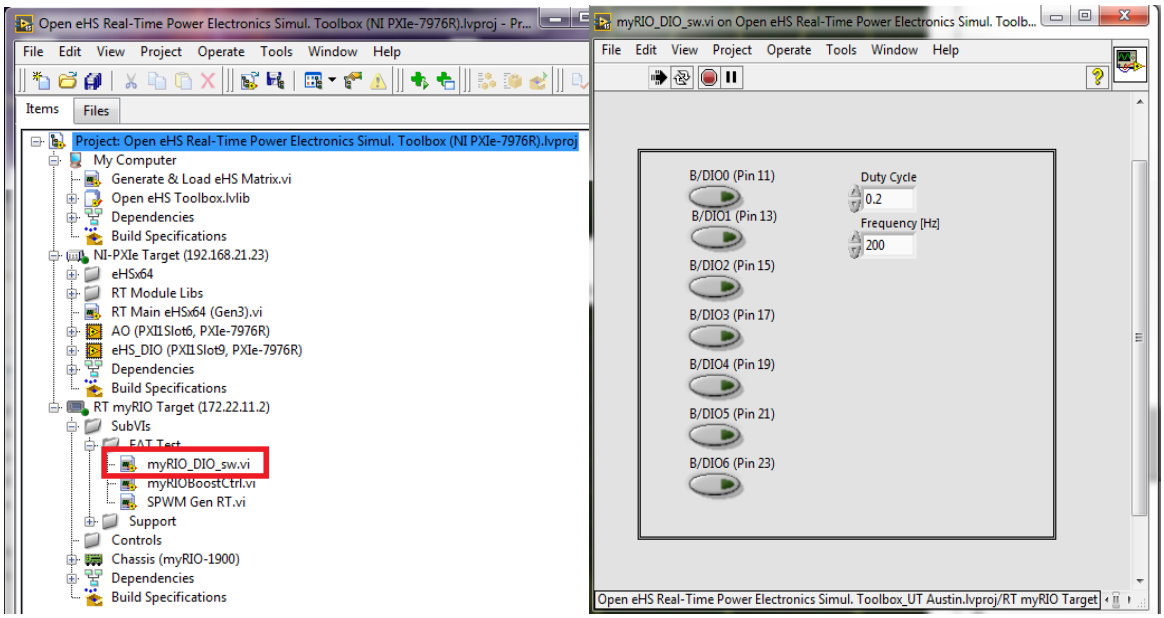


Figure 58: Front panel of myRIO_DIO_sw.vi

Step 7: On the *eHS DAQ.vi* front panel, select **Y04** and **Y01** on the scopes.

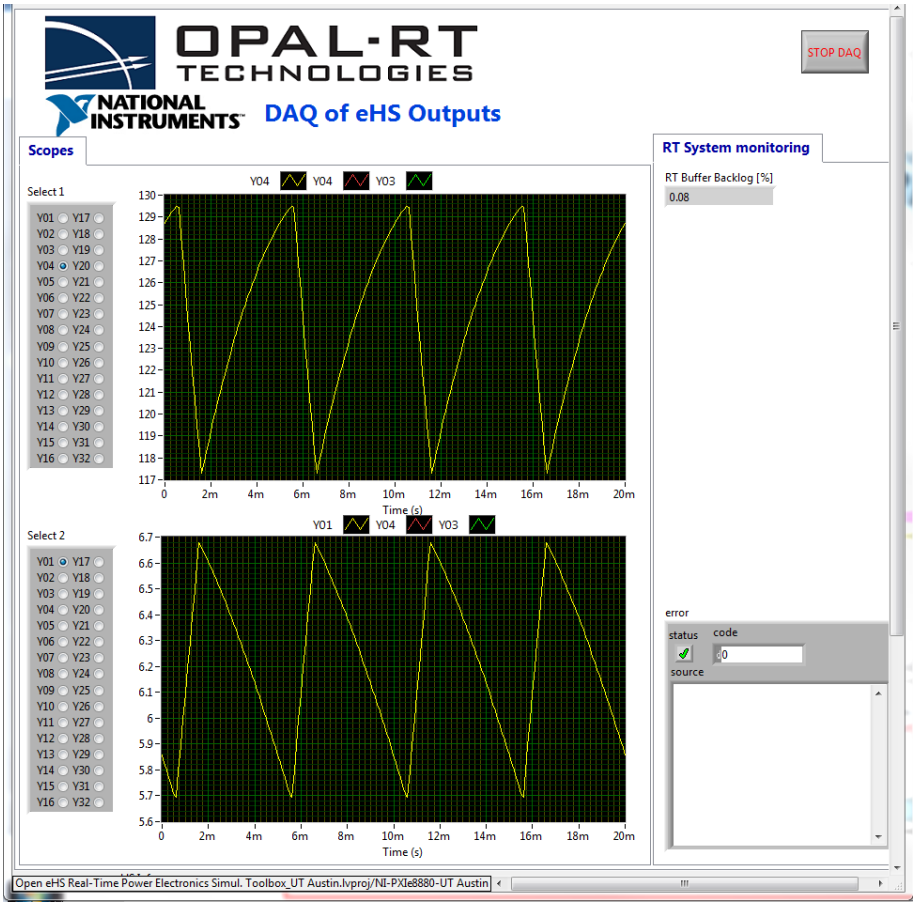


Figure 59: Block diagram of *myRIO_DIO_sw.vi*

Step 8: On the *myRIO_DIO_sw.vi* front panel, you can modify the frequency and duty cycle of the PWM generator. At the same time, you can observe the operating voltage point moving on the *eHS DAQ.vi* front panel.

Stop the simulations.

Step9: Controlling the Boost converter in a closed-loop system:

Physical wiring required for this step:

- *Connect Port B/PWM0 on the myRIO to the DDCB P0.0 port of the PXIe-6581B FlexRIO Adapter Module (Slot 8 in PXIe chassis).*
- *Ensure that the myRIO and FlexRIO Adapter Module are sharing a common ground. The eHS firmware is configured to accept 3.3VTTL logic on the DDCB connector which is the voltage level supported by the myRIO.*
- *Connect the Analog Input of myRIO B/AI0 to the analog output AO01 of FlexRIO PXIe-5742R FlexRIO Adapter Module (Slot 6 in PXIe chassis).*

Open and run *myRIOBoostCtrl.vi* located under the myRIO target in the LabVIEW project. Press the **Help** button to see the diagram of the closed-loop configuration (See figure below). Press the **Back** button to return to the parameters panel

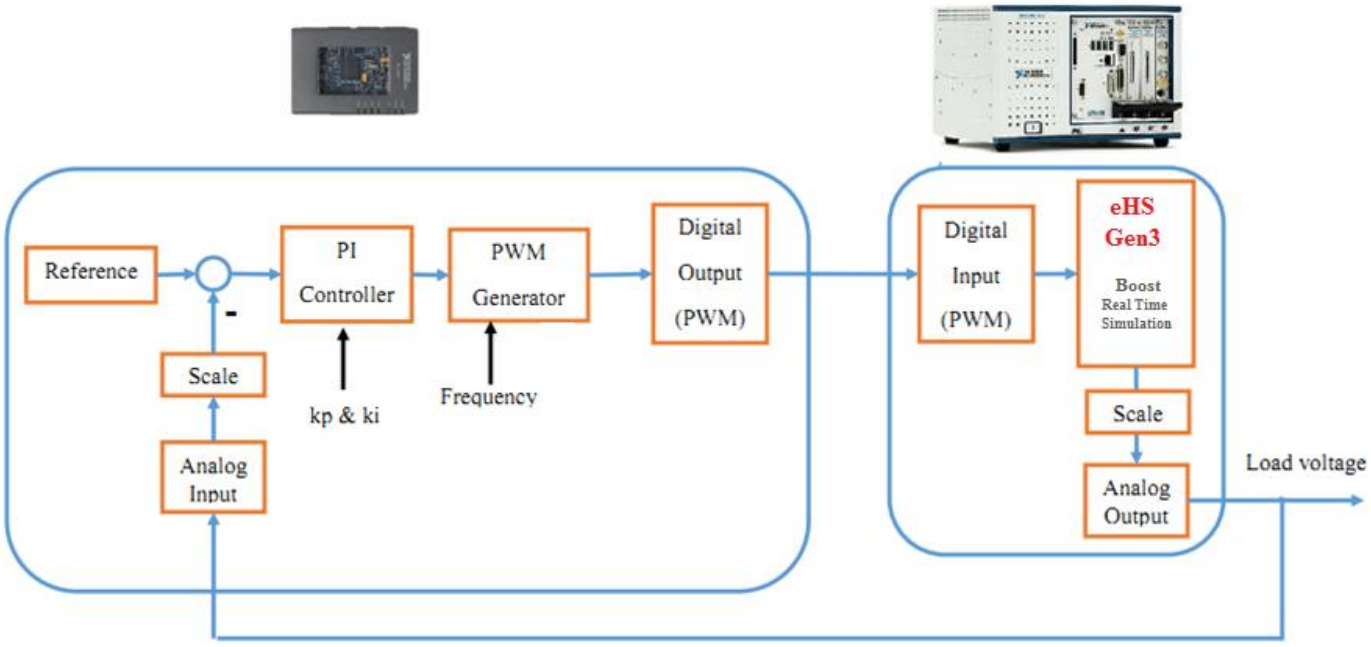


Figure 60: HIL block diagram of Boost convertor

In this panel, the AI Scaling and AI Bias parameters are used to scale and add bias to the Analog input for the myRIO. Adjust the scale and bias parameters in both the *RT Loops.vi* front panel and the *myRIOBoostCtrl.vi* front panel.

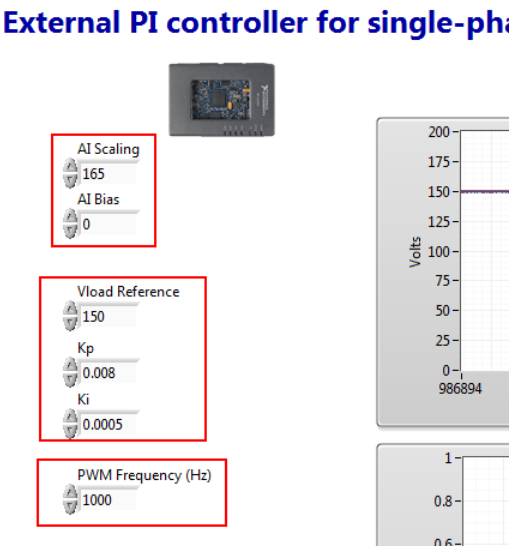


Figure 61: Parameters configuration in myRIO side

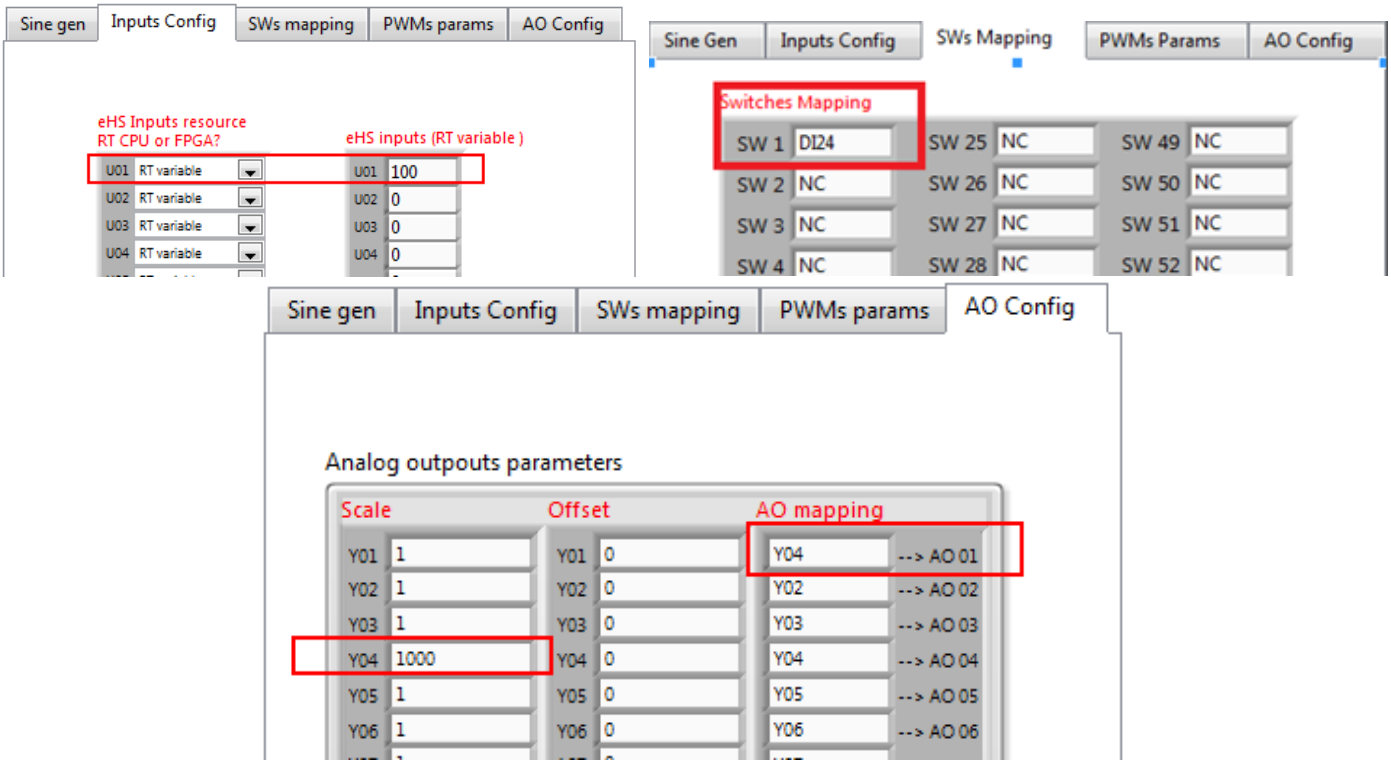


Figure 62: Parameters configuration in FlexRIO side

Step 10: Change the *PWM frequency* on the *myRIOBoostCtrl.vi* to **100** and see the effect on both the *myRIOBoostCtrl.vi* front panel and the *eHS DAQ.vi* front panel. To see how the external controller tracks the input voltage reference, change the **Vload Reference** parameter and observe the effect on the *myRIOBoostCtrl.vi* front panel. See how duty cycle is in tune with the PI controller to track the reference voltage. Modify the **Kp** and **Ki** parameters of the PI controller. The closed-loop controller might be unstable when the **Kp** parameter is increased.

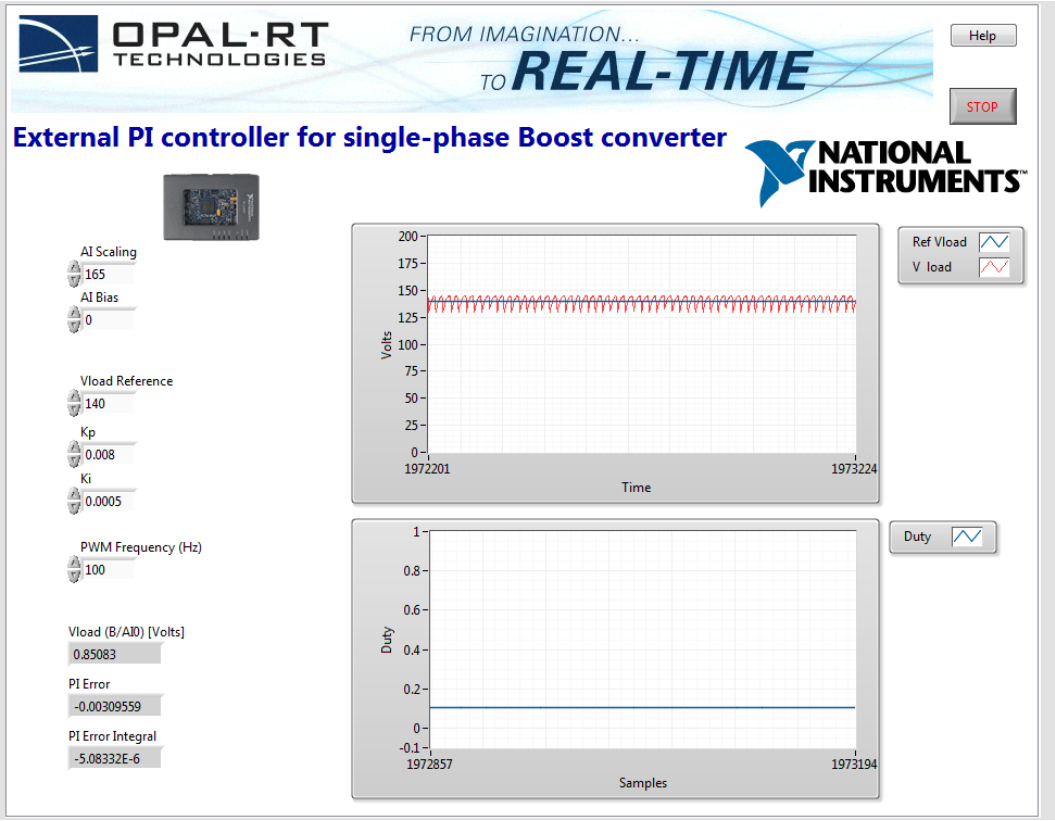


Figure 63: External PI controller for Boost converter using myRIO

TUTORIAL 3: REAL TIME SIMULATION OF 3-PHASE INVERTER

This tutorial will explains how to run the real-time simulation of a 3-phase inverter with an internal SPWM signal and an external SPWM signal. It will also explain how to use an external controller in a closed-loop control configuration.

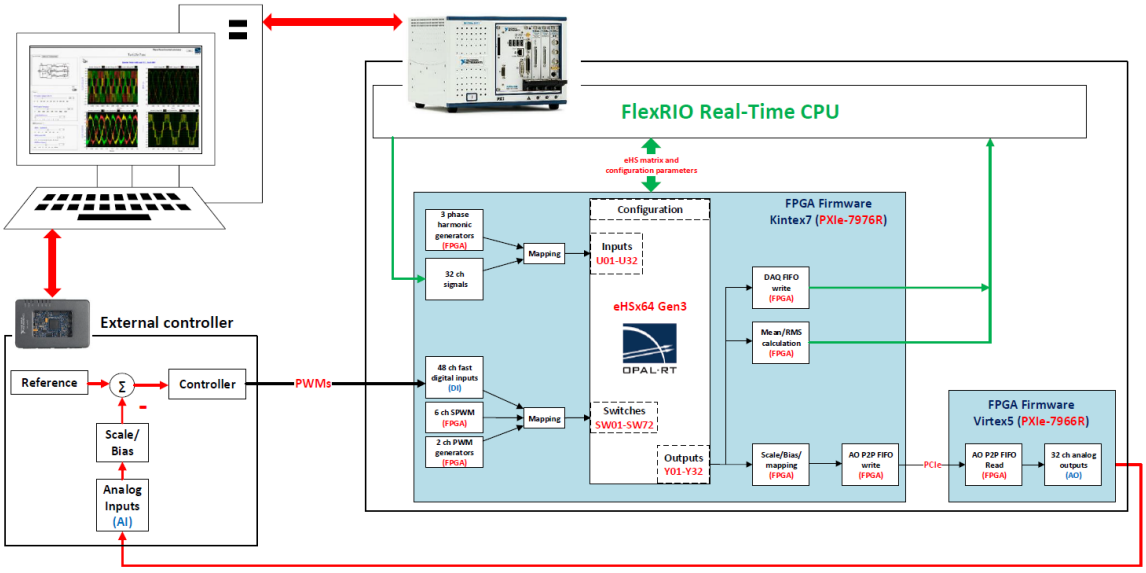


Figure 64: Hardware in loop (HIL) and external controller for Boost tutorial

Step 1: Navigate to the *PSIM_Multisim_SPS_Models* directory and use one of the supported circuit editors (MATLAB®/Simulink® with SimPowerSystems™ Toolbox, PLECS Blockset, PSIM or NI Multisim v13) to open and explore one of the following models:

- Inverter3ph _Psim.psimsch
- Inverter3ph _SPS.mdl
- Inverter3ph _PLECS.mdl
- Inverter3ph.ms13.

In this configuration, **U01** and **U02** are the input voltage sources of the eHS and **Y01** to **Y07** are the measurement points (outputs of eHS). This circuit also contains six controlled switches labelled **SW01** to **SW06**. In this tutorial, you will learn how to drive the gate of **SW01-SW06** from internal and external SPWM resources (see Figure 65).

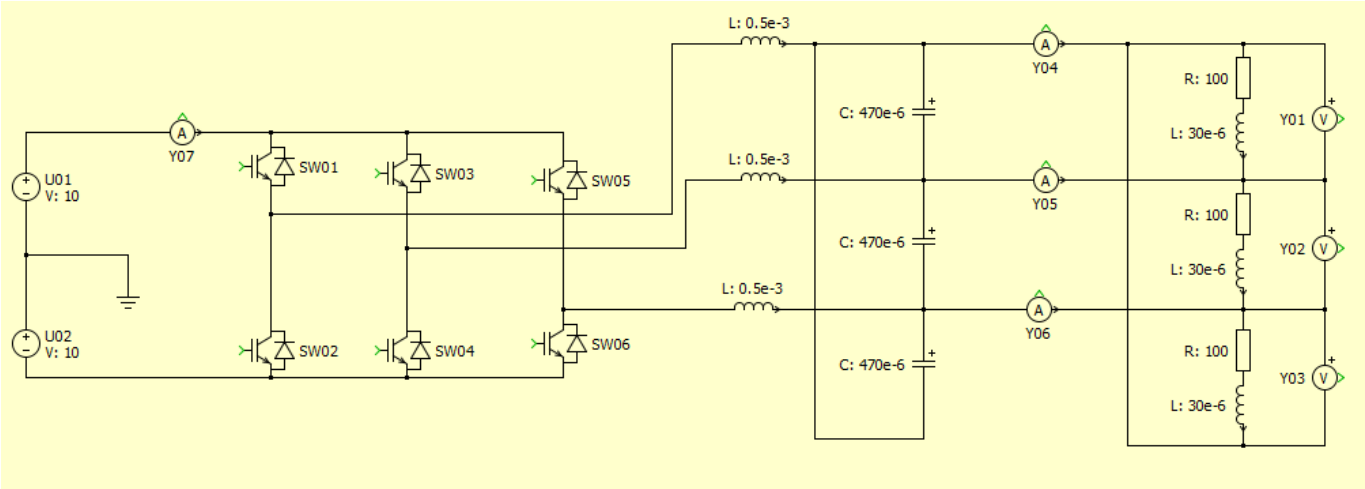


Figure 65: Real-time simulation of 3 phase inverter used in tutorial 3

Step 2: On the *Generate & Load eHS Matrix.vi* front panel, select the NI PXI target. Then, select either *Inverter3ph_Psim.psimsch, Inverter3ph_SPS.mdl, Inverter3ph_PLECS.mdl*, or *Inverter3ph.xml* from the *Psim_Multisim_SPS_Models* for the *eHS1 Model* file path. To load a second model on to the PXIe hardware for use on a second eHS core, the user could choose to select a model for the *eHS2 Model* input.

Note that to generate the eHS matrix, the user must have one of the following:

- a licensed PSIM software installed on their PC to use the *.psimsch model;
- a licensed MATLAB®/Simulink® with SimPowerSystems™ Toolbox installed on their PC to use SimPowerSystems (*.mdl or *.slx) models;
- a licensed MATLAB®/Simulink® with PLECS Blockset Toolbox installed on their PC to use PLECS (*.mdl or *.slx) models;
- a licensed NI Multisim 13 capable of generating a netlist in an XML format (refer to Apendix 2 for more detail).

Run the VI and follow the instructions detailed in the front panel (Step 1 to Step 4) to generate the eHS matrix and load it onto the NI PXI target. Note that the eHS solver sample time is accessed through the Time Step parameter. This value depends on the complexity of the circuit, and typically ranges from 150 nanoseconds to 2.5 microsecond. The Gs parameter is the conductance of the switches of the circuit. One can modify the Time Step to be equal to or larger than the minimum time step. Consult the message panel and make sure that this VI operation was successful.

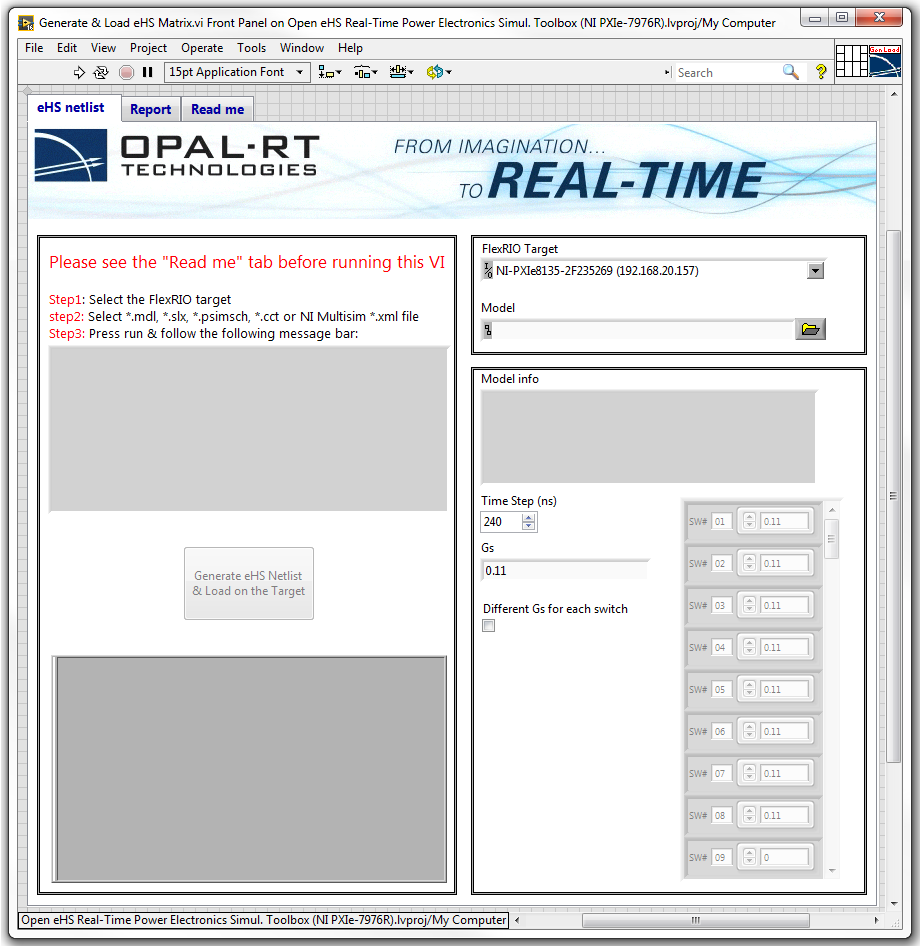


Figure 66: Generating eHS matrix and loading in the FlexRIO target for Inverter tutorial

Step 4: Open the *RT Main eHSx64 (Gen3).vi* and press **Run** on the front panel. By running this VI, the *eHS DAQ.vi* and *RT Loops.vi* front panels will appear on the screen.

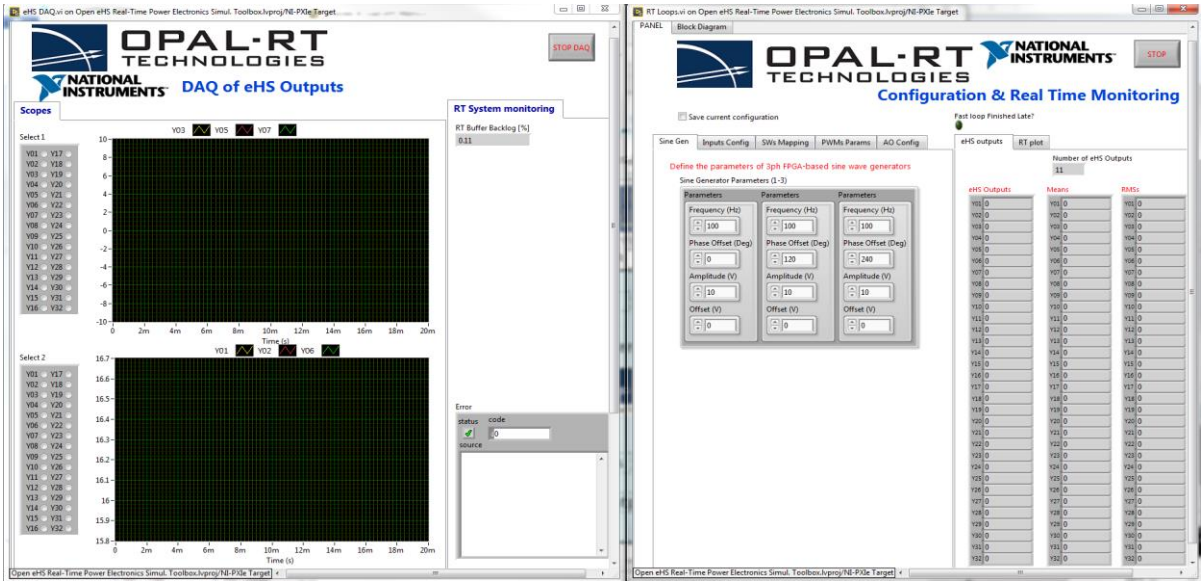


Figure 67: RT loops and DAQ panel after running the RT process VI.

Step 5: From the **Inputs Config** tab select **RT variable** for **U01** and **U02** and then modify **eHS inputs U01** and **U02** to 100 volt. From the **SWs Mapping** tab, route the internal SPWM signals to SW01-SW06 (See Figure 68).

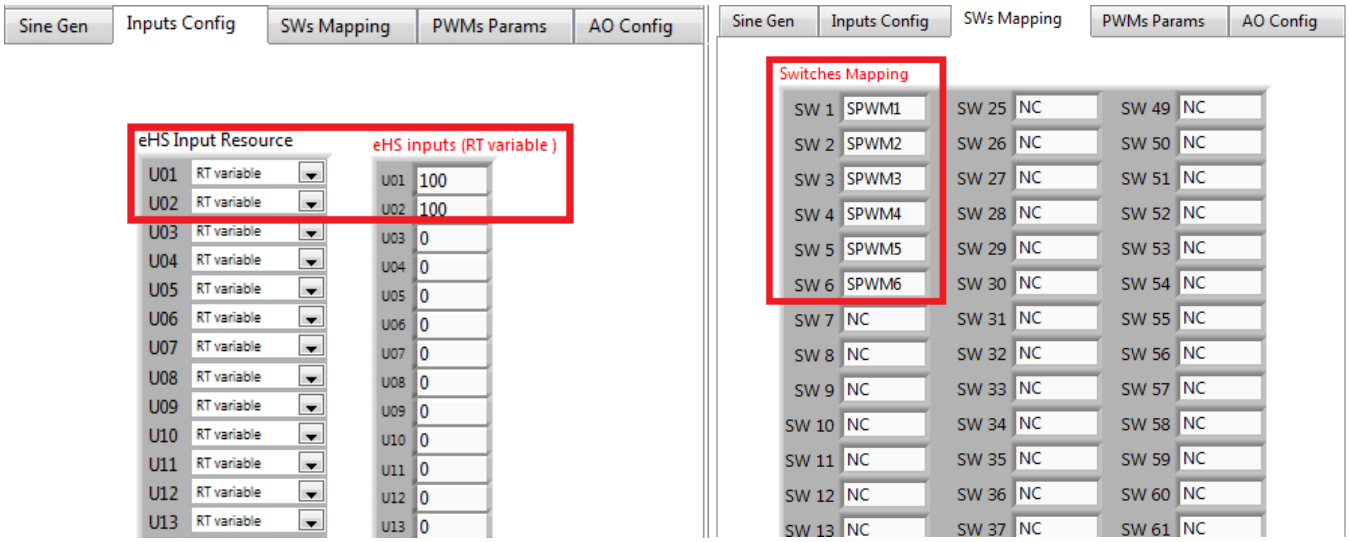


Figure 68: Inputs resources for 3 phase Inverter.

Step 6: From the **PWMs Params** tab change the **SPWM Reference** and **Carrier Frequency** and monitor the load voltages (**Y01**, **Y02** and **Y03**), load currents (**Y04**, **Y05**, **Y06**) and source current (**Y07**) on the *eHS DAQ.vi* front panel.

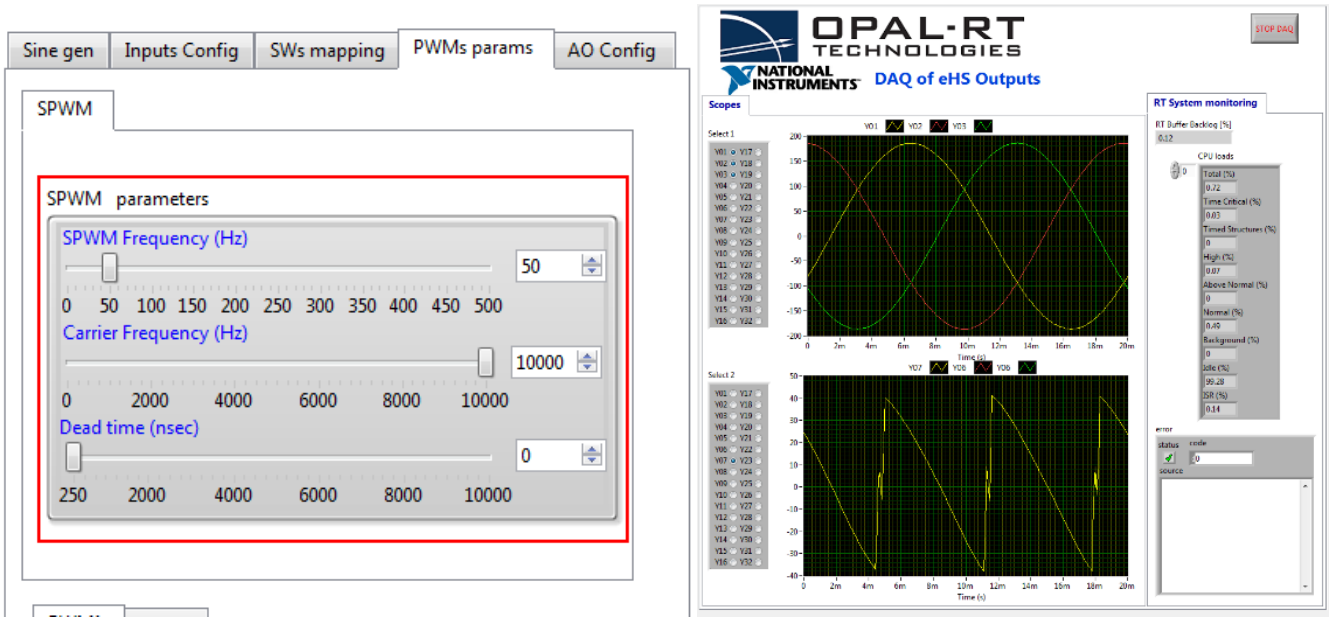


Figure 69: Adjusting carrier and reference frequency of internal SPWM generator.

Step7: HIL external controller for 3-phase inverter:

Physical wiring required for this step:

- *Connect the B/AI0, B/AI1 and B/AI2 Analog Inputs of the myRIO to the AO01, AO02 and AO03 analog outputs of the FlexRIO (NI-5742 in slot 2)*
- *Connect the myRIO ports B/DIO0 to B/DIO5 to the digital inputs of the PXIe-6581B FlexRIO Adapter Module DDCB P0.18 to DDCB P0.23 (slot 4 in PXIe chassis).*

From the LabVIEW project tree, open and run *RT_Inverter_Hysteresis_ctrl.vi* located under the myRIO target.
Press the **Help** button to see the diagram of the closed-loop configuration (see Figure 70). Press the **Back** button to return to the parameters panel (See Figure 71).

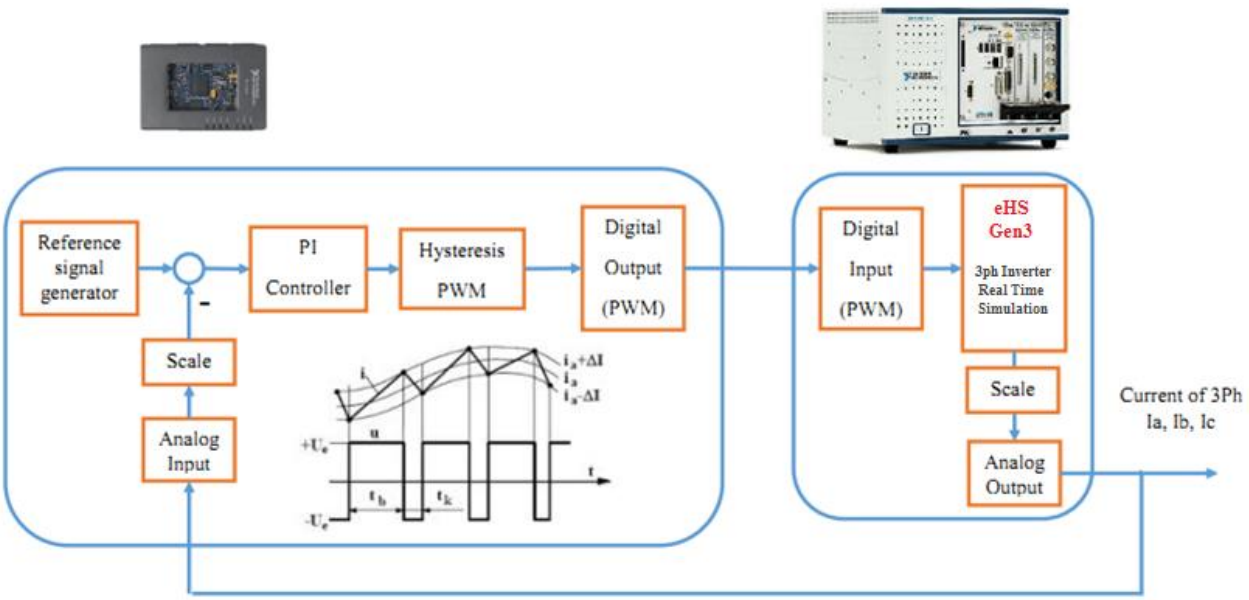


Figure 70: HIL block diagram of 3 phase Inverter

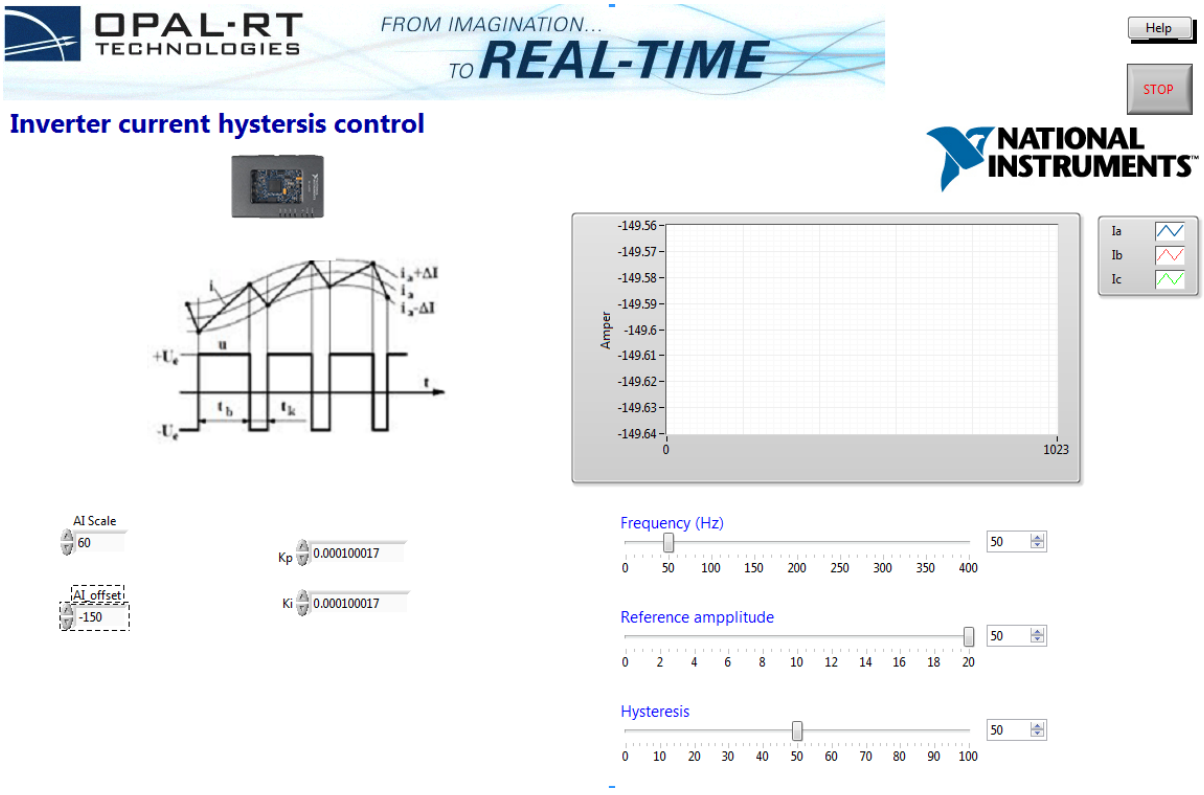


Figure 71: Front panel of RT_Inverter_Hysteresis_ctrl.vi

Step 5: From the **Inputs Config** tab select **RT variable** for *U01* and *U02* and then modify eHS inputs *U01* and *U02* to be **100** volts. From the **SWs Mapping** tab, route the external SPWM signals to SW01-SW06 (See Figure 72).

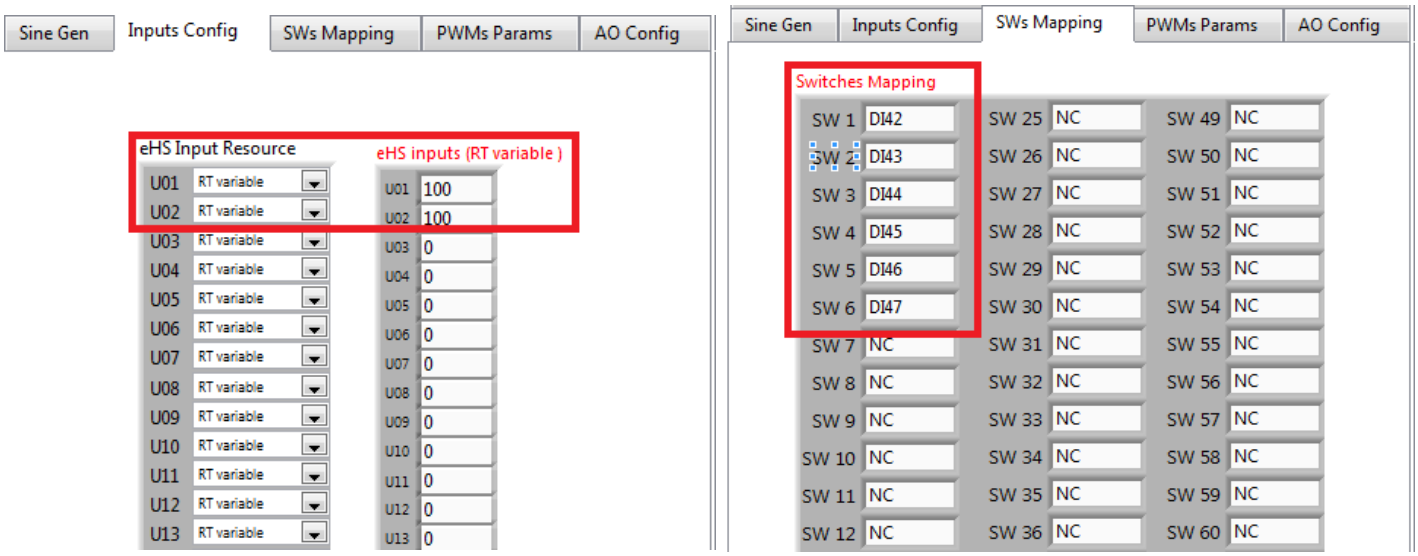


Figure 72: Input resources and switches mapping for 3 phase Inverter.

Step 6: From the **AO Config** tab, modify scale and offset of Y04, Y05 and Y06 (load currents of inverter). Additionally, route Y04, Y05 and Y06 to AO01, AO02 and AO03, respectively (See Figure 73).

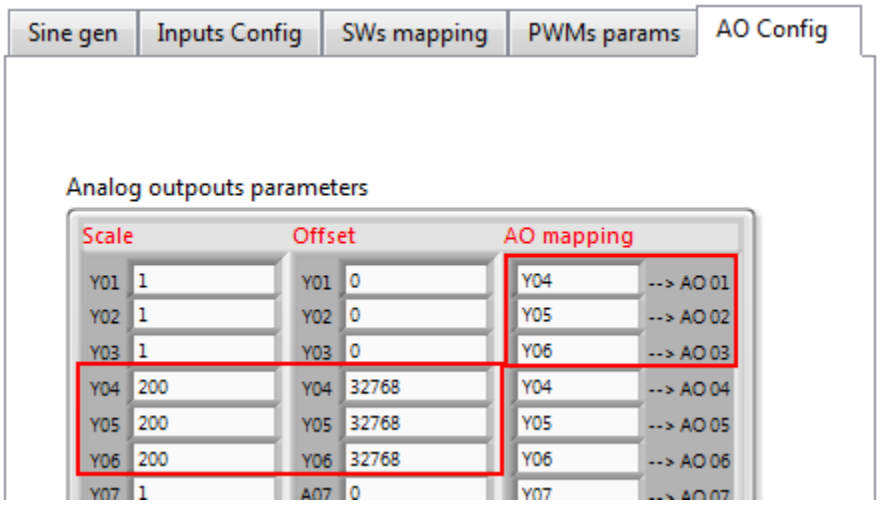


Figure 73: Scale, offset and mapping of load currents of Inverter (Y04, Y05 and Y06)

Step 7: On the front panel of *RT_Inverter_Hysteresis_ctrl.vi*, modify the scale and offset of the analog inputs to have the values shown in Figure 74. Also, modify the external controller parameters, PI gains, reference frequency, reference amplitude, and hysteresis bound and see the effect on the eHS DAQ and myRIO front panels.

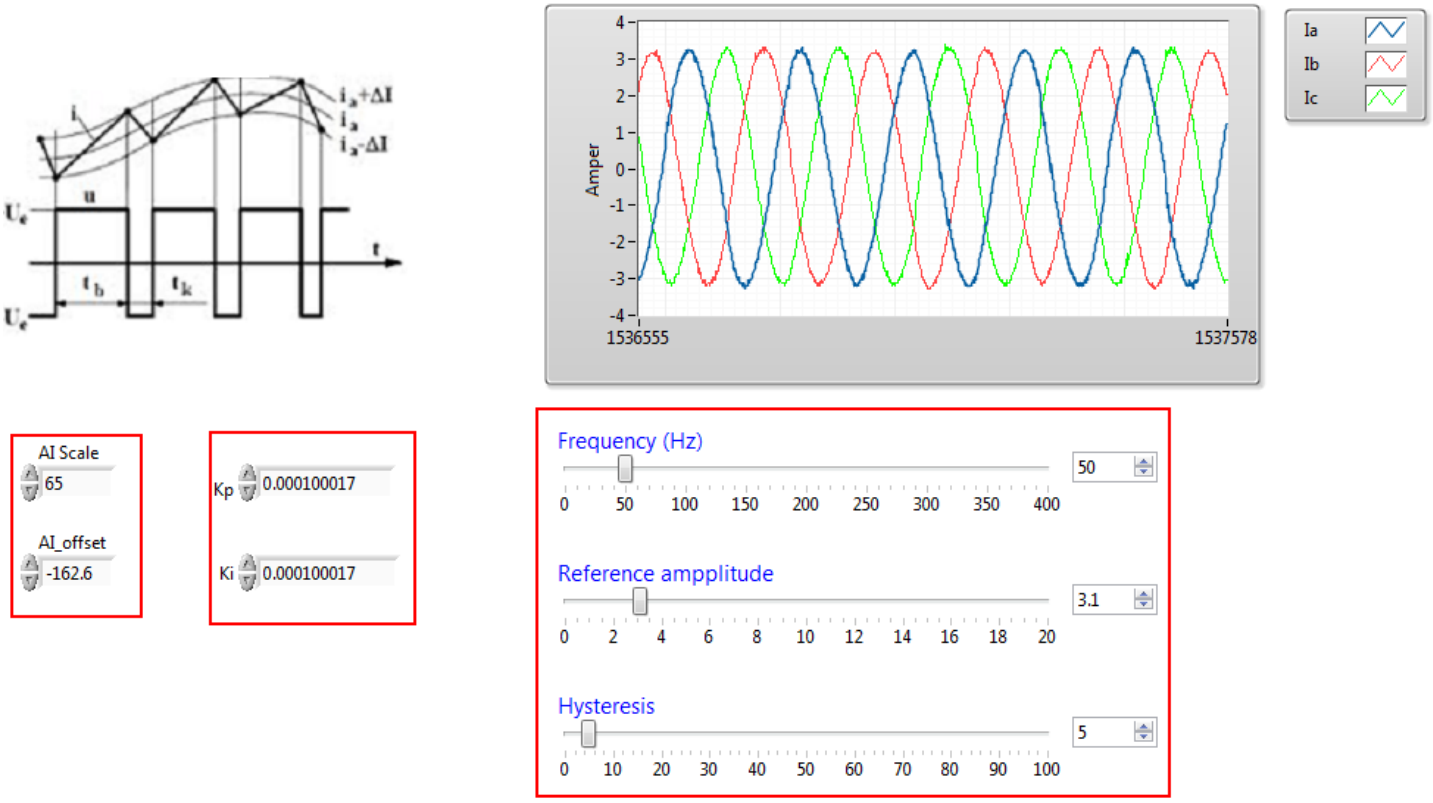


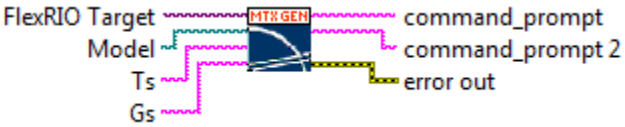
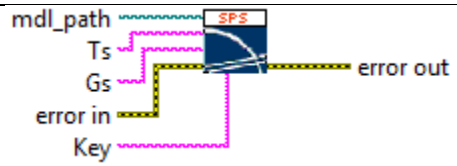
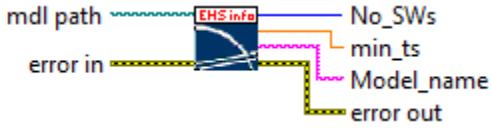
Figure 74: Scale, offset in myRIO side and modifying controller parameters and see the effect

APPENDIX 1: OPAL-RT LIBRARIES AND RELATED SUBVIS

LabVIEW API Index

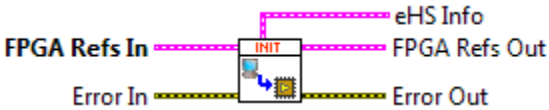
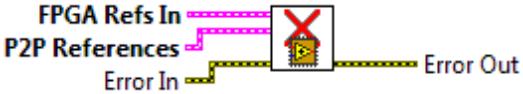
Host VIs

Host eHS (Gen3) Library

LabVIEW symbol	Name of VI	VI Short Help Description
	Generate & Load eHS Matrix	Main function to load model information, to modify time step and Gs, to generate eHS netlist matrix, and to transfer it to the FlexRIO target.
	Generate eHS Matrix.vi	Generates eHS matrix and loads it in the target.
	Matlab_ Generate eHS Matrix.vi	Generates eHS matrix using Matlab. Access only from gen_eHS_mtx.vi.
	Extract Model Information.vi	Extract model information such as minimum required time step, number of switches etc.
	Matlab_Extract Model Information.vi	Extract model information such as minimum required time step, number of switches etc. using Matlab. Access only from mdl_info.vi.

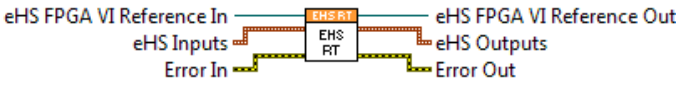
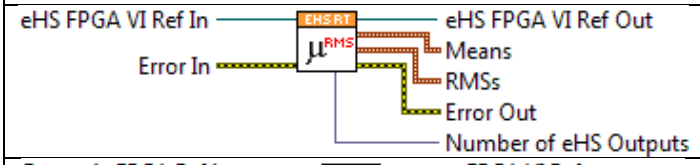
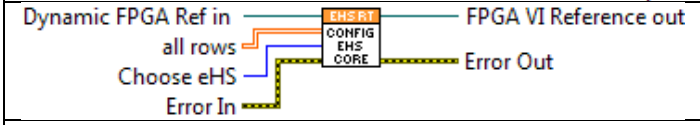
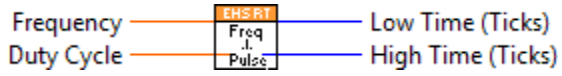
REAL-TIME LIBRARIES

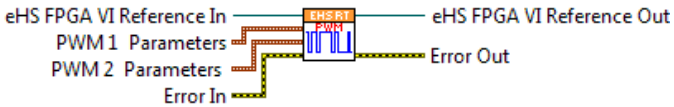
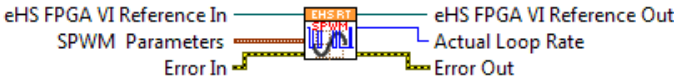
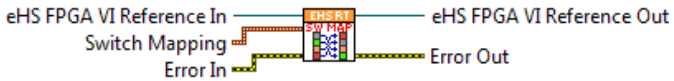
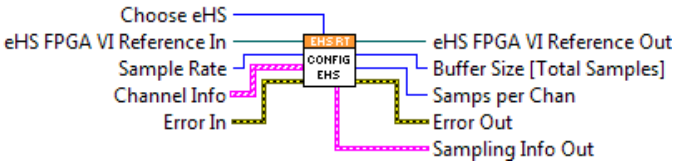
EHSx64

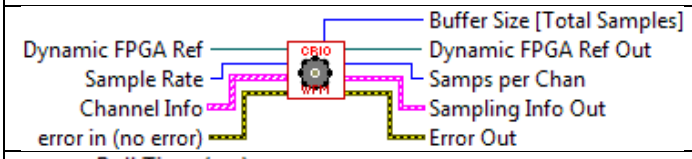
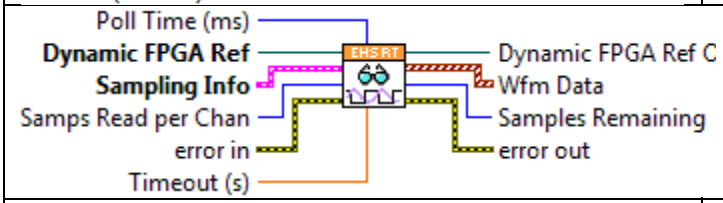
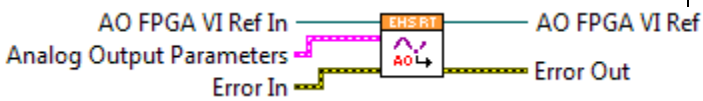
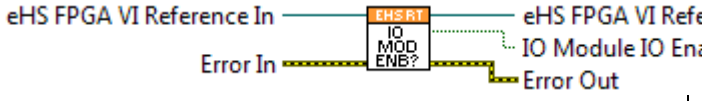
LabVIEW Symbol	Name of VI or Control	VI Short Help Description
	RT Main eHSx64 (Gen3).vi	Main real-time function including FPGA initialization, eHS configuration, data acquisition loop and real-time loops.
	Open All FPGAs.vi	Open and initialize FPGAs, FIFOs and P2P streams.
	eHS Init.vi	eHS initialization and required FPGA configurations.
	eHS DAQ.vi	Monitor the output of the eHS.
	RT Loops.vi	Include real-time loops: Fast RT loop to communicate with eHS, DIO and AI. Slow RT loop to do configurations and mapping.
	Close All FPGAs.vi	Close FPGA and FIFO references and stop the firmware.

	Stop Signal (Global).vi	Global signal that can be used to stop different processes in an application.
	P2P References.ctl	References to all the P2P streams being used in the system.
	FPGA Resources.ctl	References to all the FPGA resources (hardware) being used by the RT driver.
	FPGA References.ctl	Reference to all the FPGAs\firmware that are used by the RT driver.

EHS_DIO RT LIB

LabVIEW Symbol	Name of VI or Control	VI Short Help Description
	eHS RT.vi	eHS inputs and outputs write/read from the real time loop. Inputs refer to netlist voltage/current sources. Outputs refer to netlist voltage/current measurements.
	eHS Mean_RMS.vi	Reads RMS and mean values of eHS outputs.
	eHSgen3_config.vi	Configure eHS with netlist parameters.
	Frequency To Pulse.vi	Change the frequency and duty cycle to

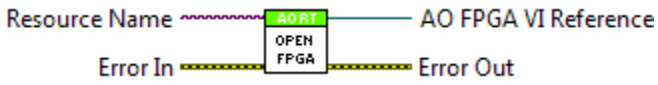
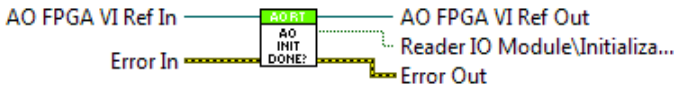
		related on/off clocks in FPGA (40Mhz).
	Sine Wave Generators.vi	Configures the FPGA based sine wave parameters.
	Loading Opal-RT.vi	FPGA and RT loading function.
	Open eHS FPGA.vi	Open a reference to the eHS_DIO FPGA\firmware.
	PWM Parameters.vi	Configures FPGA based PWMs generator parameters.
	Retrieve Control Values.vi	This VI allows the user to load previous RT Loop configuration settings.
	Save Control Values.vi	This VI allows the user to save their current RT Loop configuration settings.
	Scale Sinewave Prop32.vi	Scale sine wave properties.
	Select Channel.vi	Select which outputs of eHS core are going to be monited.
	Sine Generation Scaling.vi	Prepare parameters of embedded FPGA sinewave generator.
	SPWM Parameters.vi	Configures FPGA based SPWM generator parameters.
	Switches Mapping.vi	Configures the routing between the eHS switches gates and the digital inputs of the chassis, PWMs and SPWM resource.
	Configure Individual eHS.vi	Use this VI to configure the eHS firmware. This VI can be used to configure independent eHS core by simply

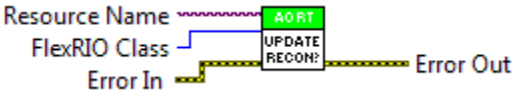
		providing a different FPGA reference.
	cRIO Configure WFM.vi	Configuration of cRIO waveform library.
	AcqRead(Wfm).vi	Reads waveform data from the DMA FIFO.
	AO Parameters.vi	Configures the mapping of the eHS outputs to the analog outputs of the chassis, as well as their gain and offset calibration parameters.
	Check IO Module Enable.vi	Use this VI to verify if the eHS_DIO firmware has finished initializing.
	Choose eHS.ctl	Define which eHS core is being targeted (eHS 1 or eHS 2).
	eHS Analog Output Parameters.ctl	Analog output parameters.
	eHS Config FPGA Ref.ctl	Reference to eHS_DIO FPGA\firmware for specific configuration settings.
	eHS FPGA VI Ref.ctl	Reference to eHS_DIO FPGA\firmware.

	eHS Info.ctl	eHS information.
	eHS Input Source.ctl	eHS input source.
	eHS Input Sources.ctl	eHS input sources.
	eHS Inputs - RT Variable.ctl	eHS inputs provided in the form of RT variables.
	eHS Outputs to AO.ctl	eHS outputs to analog outputs.
	eHS Outputs.ctl	eHS outputs.
	eHS Params FPGA VI Ref.ctl	Reference to specific the eHS_DIO FPGA\firmware parameters.

	eHS PWM Parameters.ctl	eHS PWM parameters.
	eHS Sine Generation Parameters.ctl	eHS Sine generation parameters.
	eHS SPWM Parameters.ctl	eHS SPWM mapping.
	eHS Switch Mapping.ctl	eHS switch mapping.

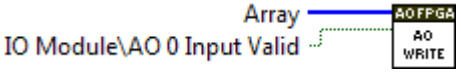
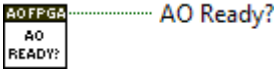
AO RT LIB

LabVIEW Symbol	Name of VI or Control	VI Short Help Description
	Open AO FPGA.vi	Open a reference to the AO FPGA\firmware.
	AO Initialization Done.vi	Use this VI to ensure that the AO firmware has initialized correctly.

	AO FPGA VI Ref.ctl	Reference to the AO firmware.
	Update Reconfiguration (797xR).vi	This VI is used to enable support for FPGA reconfiguration on FlexRIO PXIe-797xR targets. If a 797XR target is used, set the *FlexRIO Class* input to **PXIe-797xR.** Otherwise, set it to **Other**.

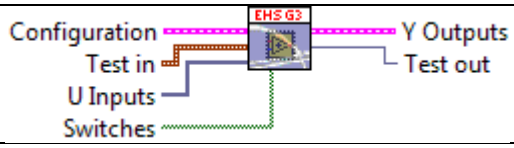
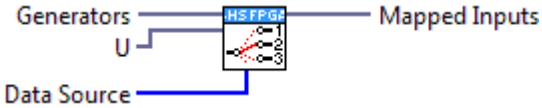
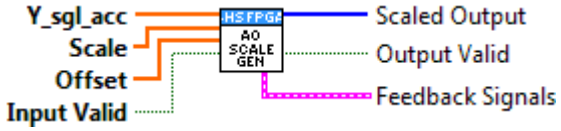
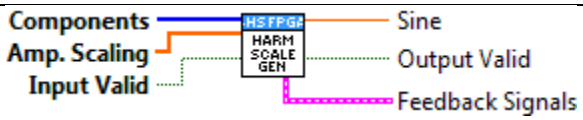
FPGA LIBRARIES

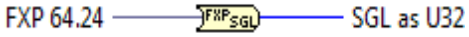
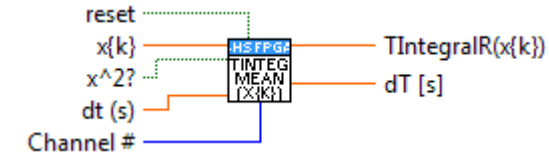
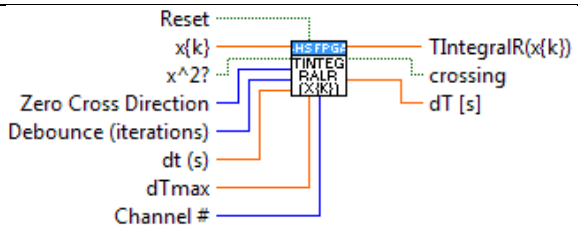
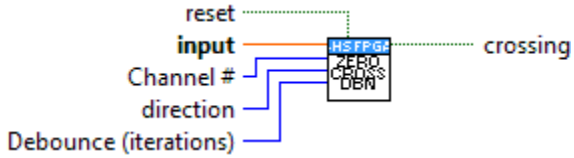
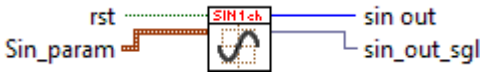
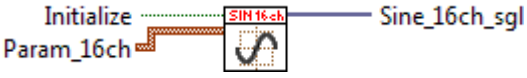
AO FPGA LIB

LabVIEW Symbol	Name of VI or Control	VI Short Help Description
	AO module.vi	Use the VI as a main template to support AO task on FlexRIO targets.
	AO Write.vi	Use this VI to write all data to all ports on the hardware.
	AO Ready.vi	Use the VI to check whether the AO module is ready to output new data.

	AO Variables.ctl	Analog output data.
	AO State.ctl	Analog output states used for state machine.

EHS_DIO FPGA LIB

LabVIEW symbol	Name of VI or Control	VI Short help description
	FlexRIO_eHS_gen3_firmware.vi	Main VI for the eHS_DIO firmware.
	eHS_gen3_core.vi	eHS gen3 IP core for FPGA firmware.
	Selection Mapper.vi	This VI is used to route data coming from RT Variables and\or Internal Signal Generators
	Scaling Gen.vi	This VI is used to scale signals such that they can be output by analog output hardware whose AO range is too small to support the signals true amplitude.
	Harmonic Scaling Gen.vi	This VI is used to scale harmonic signals for output.
	U32 to SGL.vi	Type cast data from U32 to single floating point.

	SGL to U32 (Gen).vi	Type cast data from single floating point to U32.
	Boolean to FXP-32-32.vi	Use this VI to convert a Boolean value to a fixed point value.
	FXP to SGL (EncodeU32).vi	Use this VI to convert a fixed point value to a single.
	FXP32x32 to SGL32x32.vi	Use this VI to convert an array of fixed point values to an array of singles (array size 32).
	Harmonic Sine Generator.vi	This VI generates sinus waveforms with up to 9th order harmonic.
	TIntegralR Multichannel with External Reset.vi	Multichannel integrator with external reset
	TIntegralR Multichannel.vi	Multichannel intergrator
	Zero Crossing with Debounce Float Multichannel.vi	Detects zero crossings of an input signal. The data type you wire to input determines the output data type.
	1 Channel Sine Wave Generator.vi	One channel FPGA-based sine wave generator.
	16 Channel Sine Generator.vi	16 channel FPGA-based sine wave generator.
	AO Offset.ctl	AO offset settings.
	AO Scaling.ctl	AO scaling settings.
	Mean.ctl	Mean values.

	Solver Status Booleans.ctl	Control for solver status.
	Solver U.ctl	U inputs to the eHS core.
	Solver Y.ctl	Y outputs of the eHS core.
	FPGA Globals.vi	FPGA global variable that is used to transfer data between parallel FPGA processes. This data does not need to be transferred to the host.
	Output Channel Settings.ctl	Output channel mapper settings.
	Output Mappings.ctl	Output mapping.
	PWM Channel Map.ctl	PWM channel mapping.
	PWM Mappings.ctl	PWM mapping.
	PWM Origin.ctl	PWM origin.
	72 Gate to eHS.ctl	Control used by the "72 Gate to eHS" register.
	Generators.ctl	Control used by the "Generator Reg" register.
	PWMs.ctl	Control used by the "PWMs Reg" register.
	PWMs48chDIO.ctl	Control used by the "" register.
	Scaled Output.ctl	Control used by the "Scaled Output" register.
	SPWM.ctl	Control used by the "SPWM State" register.
	Y_FXP.ctl	Control used by the "Y_FXP" register.
	Generator Settings.ctl	Sine wave settings for internal generator.

	Generators Array.ctl	One of the parameters used by the Generator Settings control.

APPENDIX 2: CHANGING THE FPGA BITFILE FROM THE RT DRIVER

When developing FPGA firmware, it will be necessary to load the new bitfiles onto the targets of your choosing. The process of updating the bitfiles that will be used by your targets is fairly simple and has been standardized.

The package that is provided supports (out of the box) loading bitfiles onto a PXIe-7976R (NI 6851B FAM) used for the DIO and eHS firmware and a PXIe-7976R (NI 5742) used for the AO firmware. The figure below shows the libraries\modules that are part of the real-time driver. These libraries contain the code that will allow you to control\monitor the firmware in each FPGA.

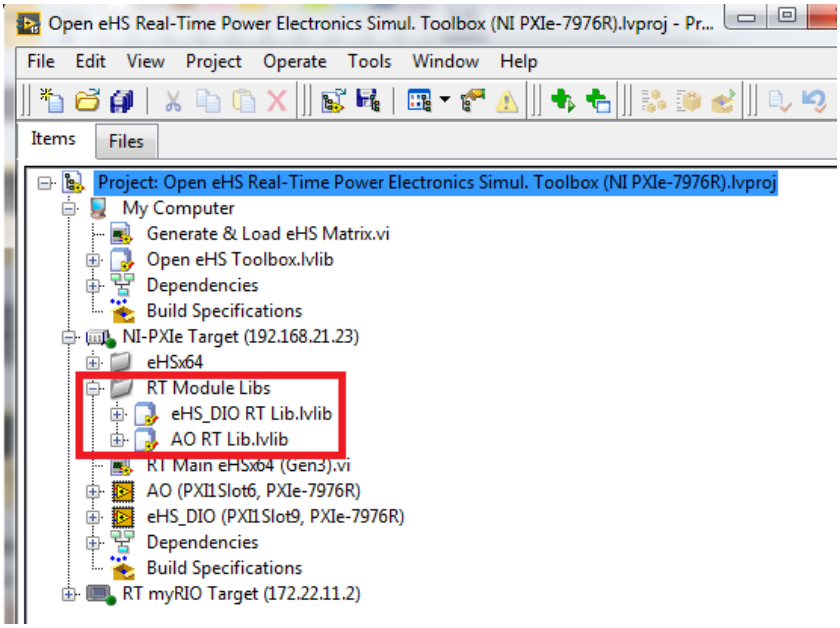


Figure 75: RT modules in LabVIEW project

Expanding any one of the libraries' tabs reveals two virtual folders. If you expand the *SubVI* folder, you will be able to view the source code of each module. Each module has a VI which is used to load the firmware onto one of the FPGAs. For example, if we look into the *eHS_DIO RT.lvlib* library, we will be able to find a VI entitled *Open eHS FPGA.vi* as seen in the figure below.

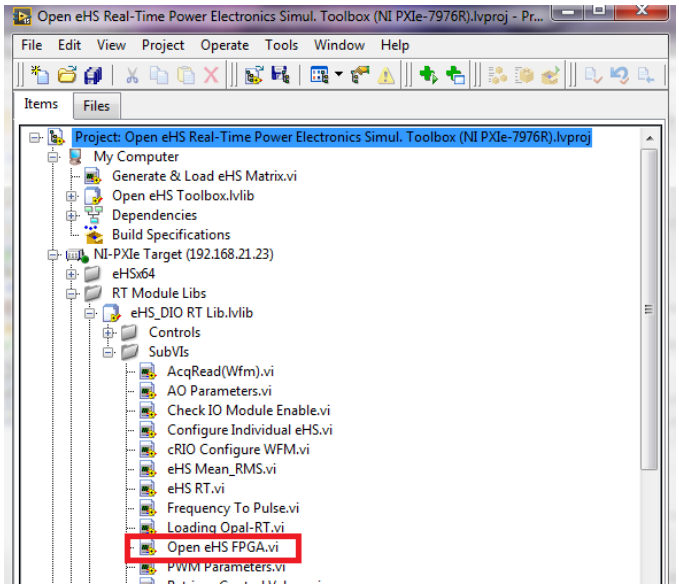


Figure 76: Open eHS FPGA.vi

This VI handles the task of loading the eHS firmware onto one (or more) of the FPGAs. By default, an FPGA personality is already configured and ready to be loaded. If you would like to change the firmware that will be loaded onto the FPGA, right-click on the *Open FPGA VI Reference.vi* and choose the option to *Configure Open FPGA VI Reference…*.

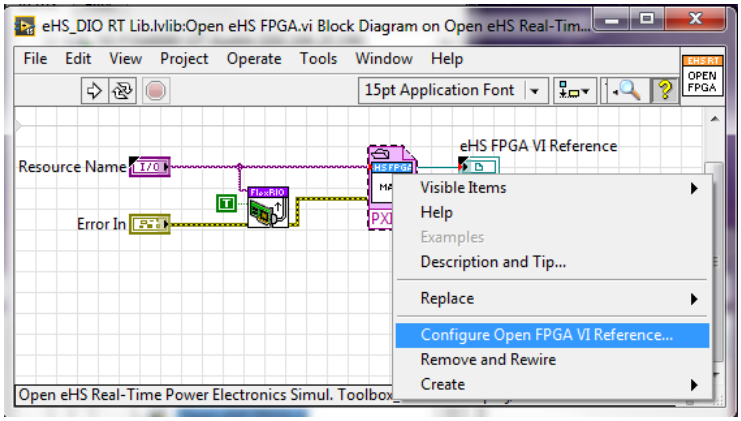


Figure 77: Configure the Open FPGA VI Reference.vi

Next, click on the **Browse** button in the *Configure Open FPGA VI Reference* window and choose the bitfile you would like to load onto the FPGA target. Click **OK.**

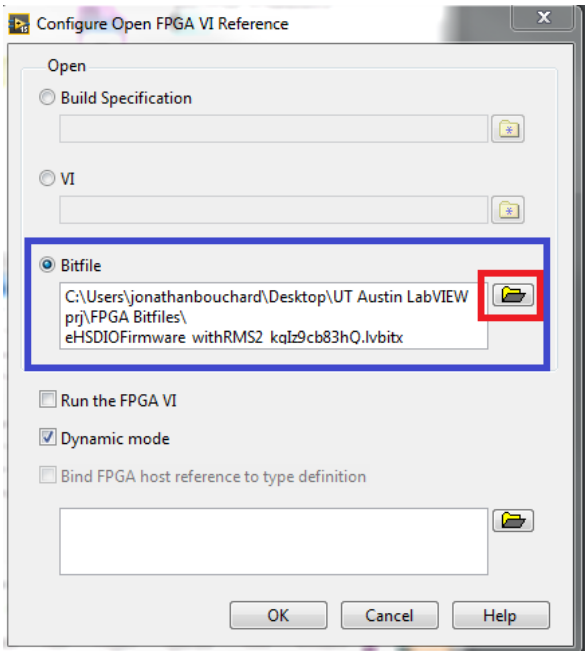


Figure 78: Choose the bitfile that will be loaded to the target

Usually, when you load a new FPGA bitfile, you will notice a broken run arrow at the output of the *Open FPGA VI Reference.vi.* This is caused by the indicator (LabVIEW type definition) which carries the FPGA reference throughout the application. We will have to update this type definition to reflect the new bitfile.

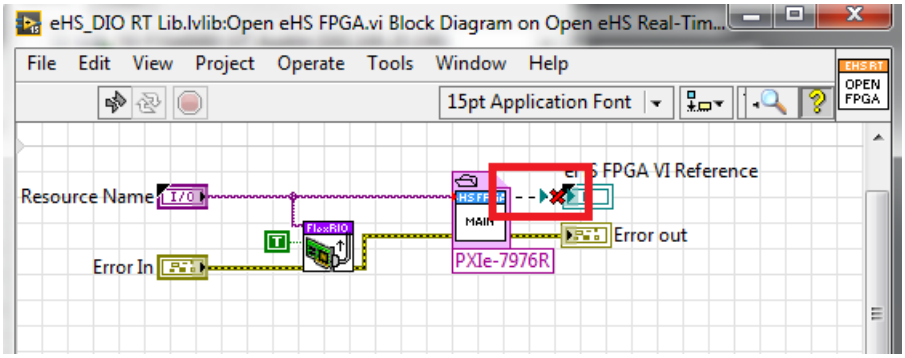


Figure 79: Broken run arrow

To resolve the broken run arrow, right click on the **eHS FPGA VI Reference** indicator and choose **Open Type Def.** This will open the type defined control.

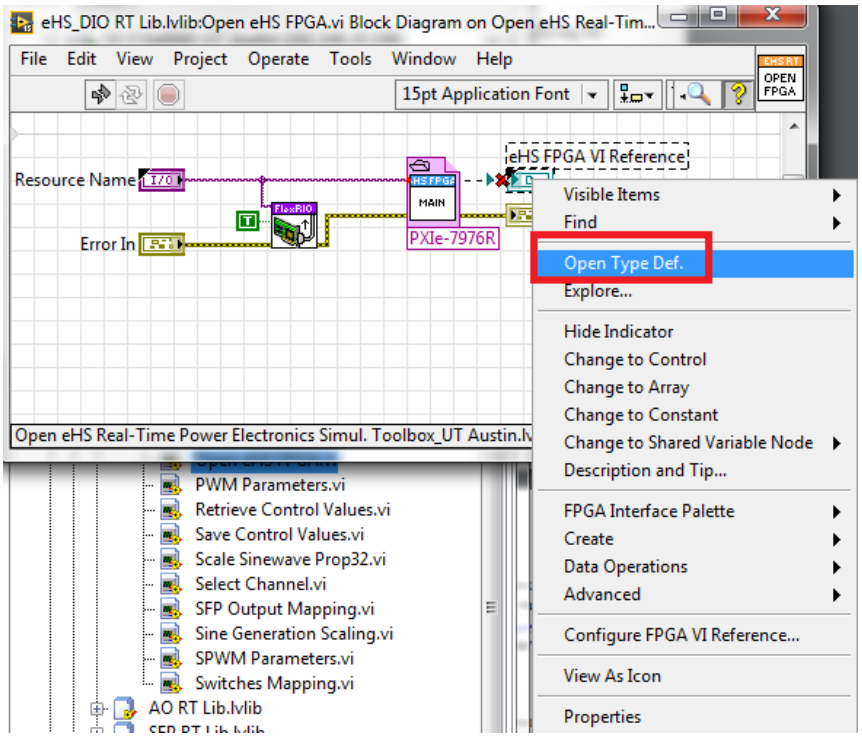


Figure 80: Opening the type def

The goal here is to change the FPGA reference contained in this type definition such that it matches the reference for the new bitfile we want to use. The easiest way to do this is to right click on the output pin of the **Open FPGA VI Reference.vi** and choose the option to create a constant.

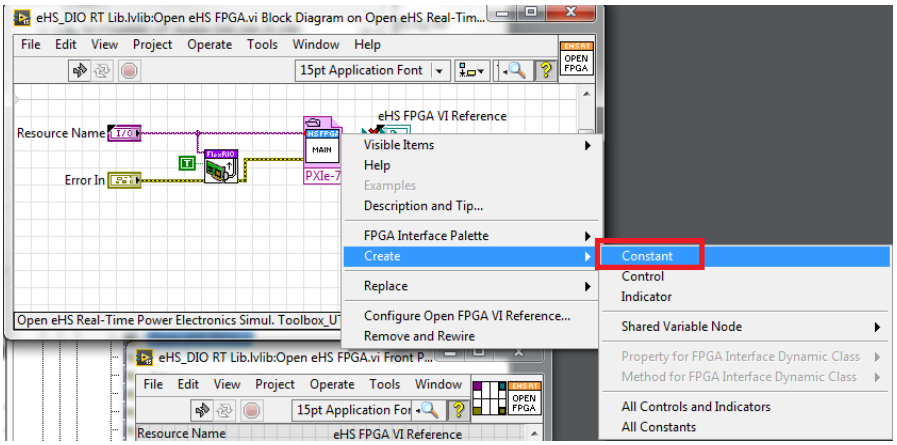


Figure 81: Creating a constant to update the type def

Replace the *eHS FPGA VI Reference* that is currently in the type definition with this new one. Close the type definition to force it to save and apply changes. The broken wire should no longer be present.

APPENDIX 3: CONFIGURING THE EHS_DIO FIRMWARE TO USE EXTERNAL OR INTERNAL POWER SUPPLIES

The package makes use of a NI 6851B FlexRIO Adapter module (connected to a FlexRIO 7976R FPGA) to perform digital input tasks. Before starting to acquire digital signals, it may be necessary to configure the eHS_DIO firmware such that it uses either an internal or external power supply as its voltage reference. By default, the eHS_DIO firmware is configured to use an external power supply for the DDCA Connector and the NI6581B internal 3.3V power supply for the DDCB Connector. The figure below details the two connector terminals on the NI 6581B. Please refer to the user manual of the NI6581B (ni.com\) for supported voltage levels.

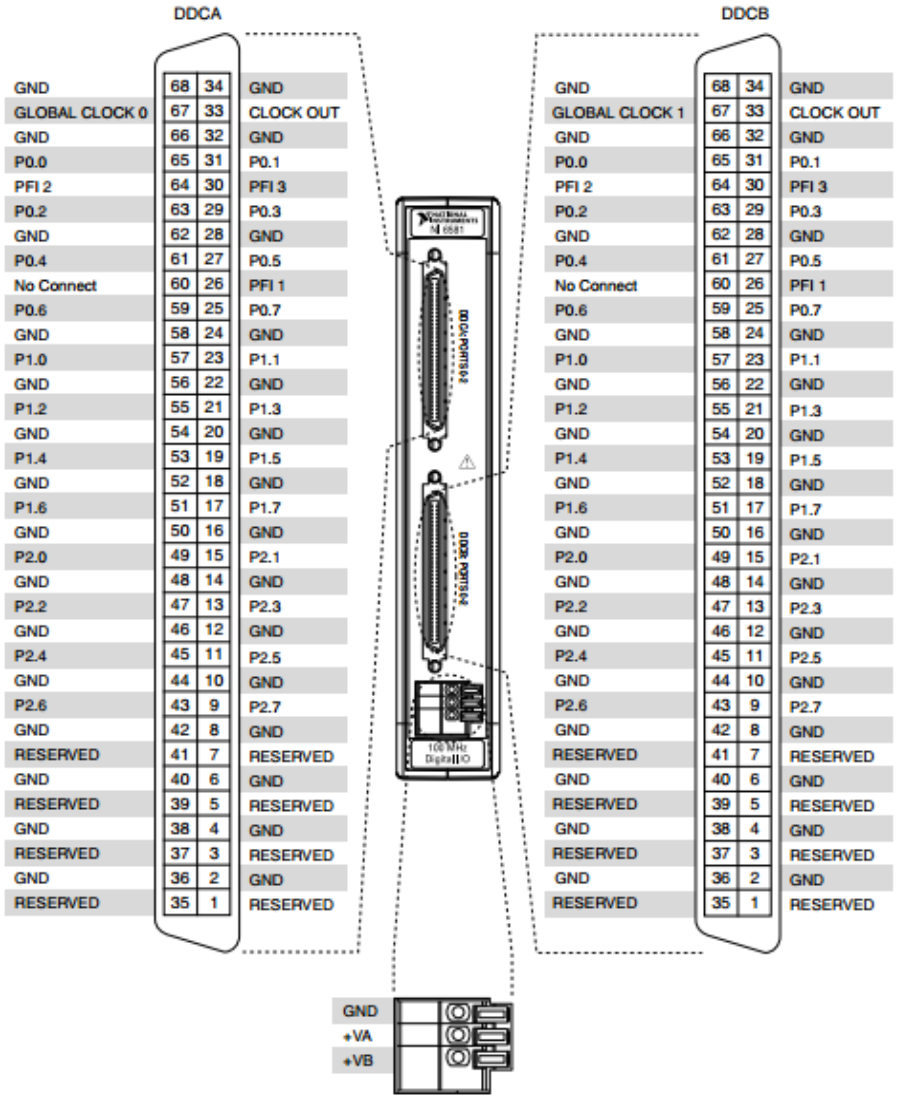


Figure 82: NI 6581B connector terminals

To modify the firmware such that either connector uses a different power supply and voltage level, you will need to modify the power supply configuration as detailed in the figure below. Change the inputs of the *IO Module**DDCA_SUPPLY*_SELECT and\or *IO Module**DDCB_SUPPLY*_SELECT nodes to either **External** or **Internal.** If the firmware is configured to use an internal power supply, make sure to choose the voltage level you want to use by modifying the *IO Module**LOCAL_SUPPLY*_SELECT node (see figure below).

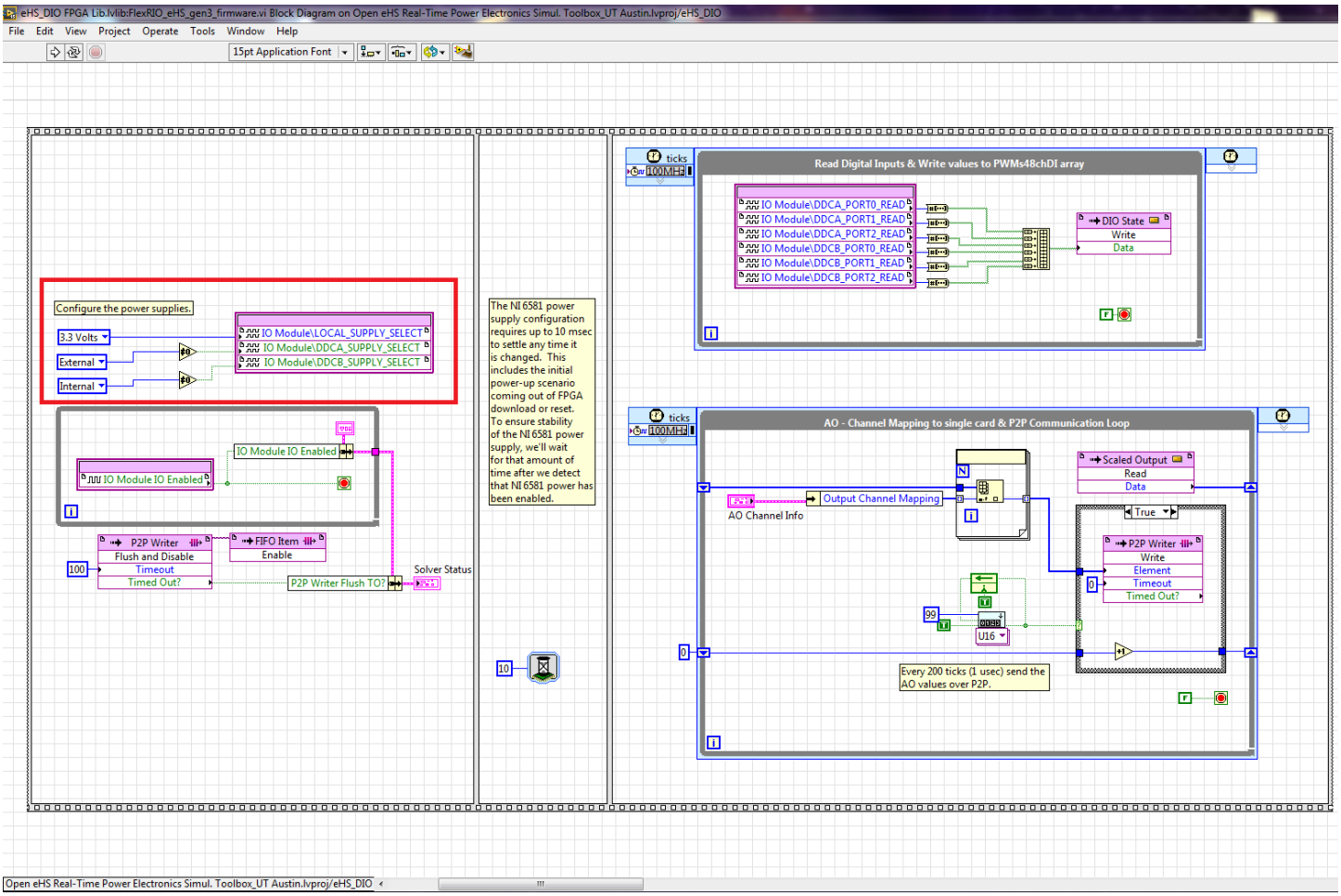


Figure 83: Modifying the eHS_DIO firmware

APPENDIX 4: BROKEN RUN ARROW ON THE EHS_DIO FPGA VI IN THE TEMPLATE

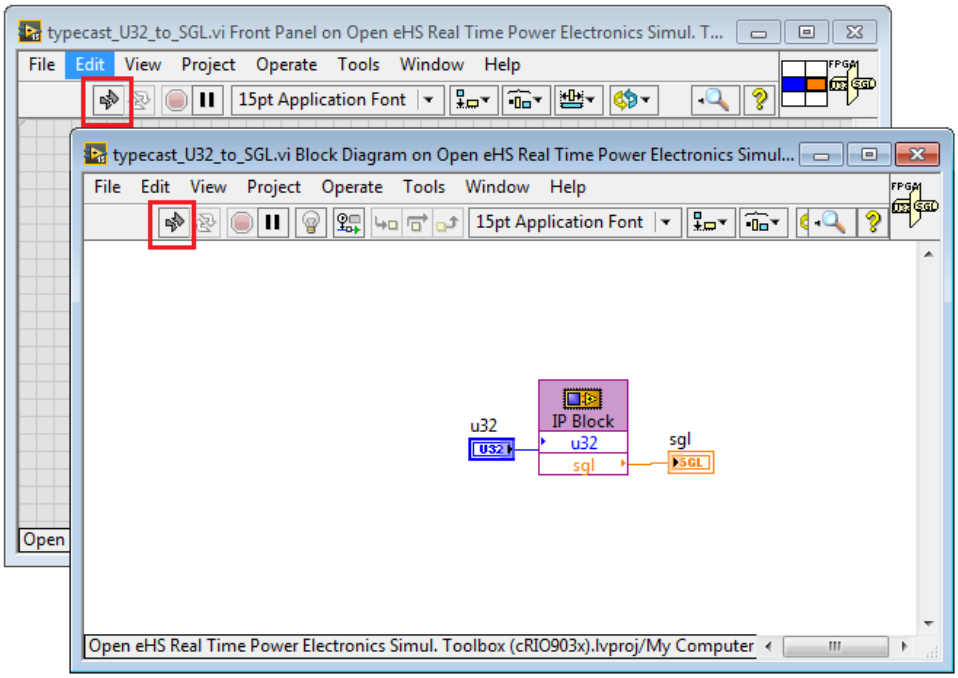
Why might the eHS_DIO Firmware.vi be broken in the template project?

The outputs of the eHS core are in single floating point (SGL) data type, which is interpreted as an unsigned integer (U32) data type. CLIP tools are used to implement the eHS core in LabVIEW FPGA, which only support fixed-point (FXP) data types as input and output terminals. IP Integrated Node tools, on the other hand, support single floating point (SGL) data types. Therefore, the eHS outputs that come from the CLIP have to be typecast to SGL using an IP Integration Node. Moreover, LabVIEW 2014 and LabVIEW 2015 (e.t.c) use different versions of Vivado to compile the FPGA bitfile, which is why some users may see that the eHS_DIO Firmware.vi is broken.

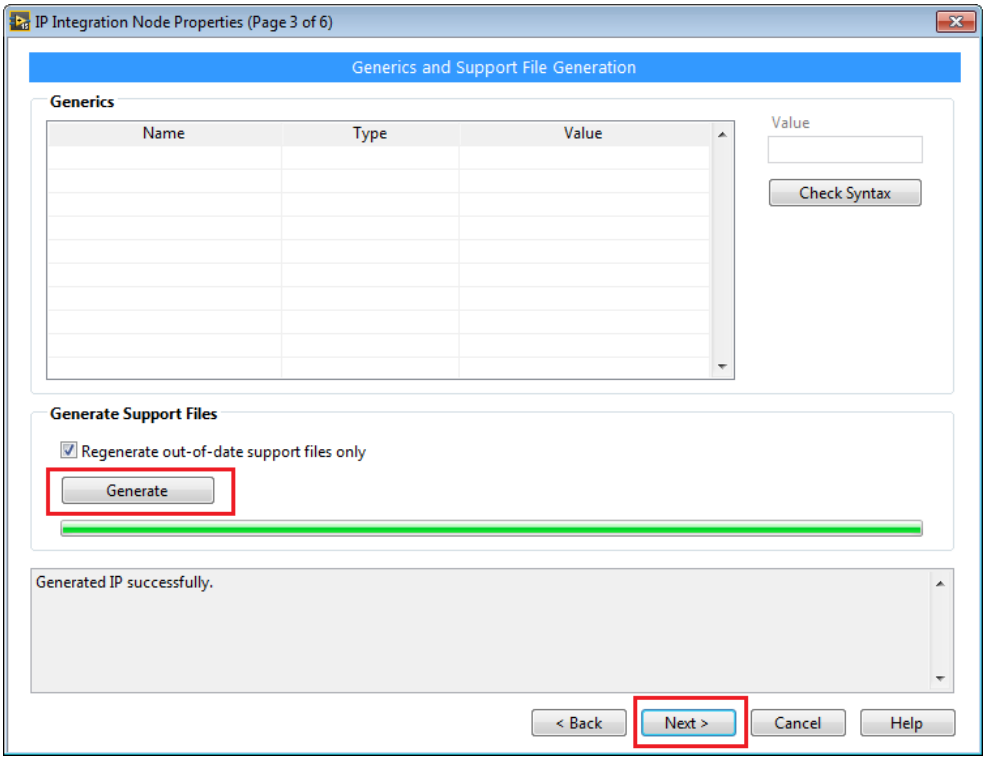
Follow these steps to regenerate the related U32 to SGL typecast DLL file in your PC:

1. Browse to *<National Instruments>\LabVIEW 201x\examples\Open eHS (7976R) OPAL-RT Libs\DIO_eHS\eHS_DIO FPGA\SubVI\Data Type Conversion.*

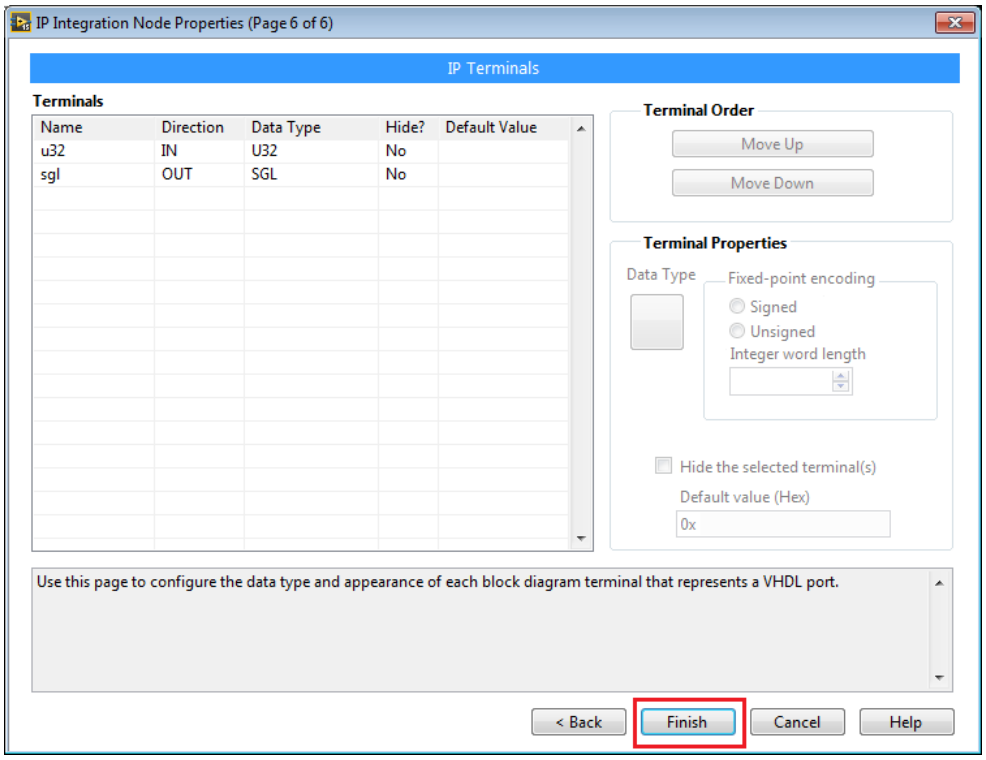
2. Open *U32 to SGL (CLIP).vi* and press **Ctrl+E** to see the block diagram for this VI.



3. Double-click on the IP Integration Node and continue clicking on the Next button until page 3.



4. Press the **Generate** button. Then, continue clicking on the **Next** button until you reach page 6 and finally click on the **Finish** button to regenerate the DLL file.

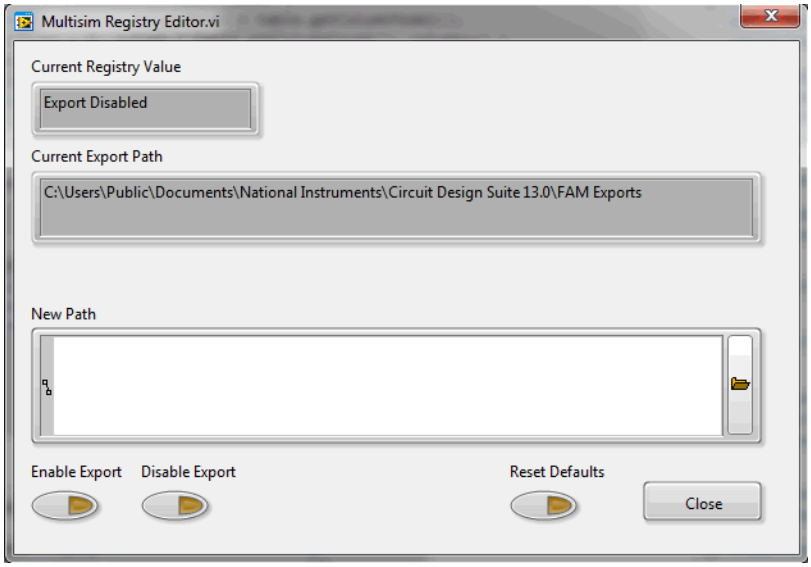


APPENDIX 5: ENABLING XML EXPORT IN NI MULTISIM

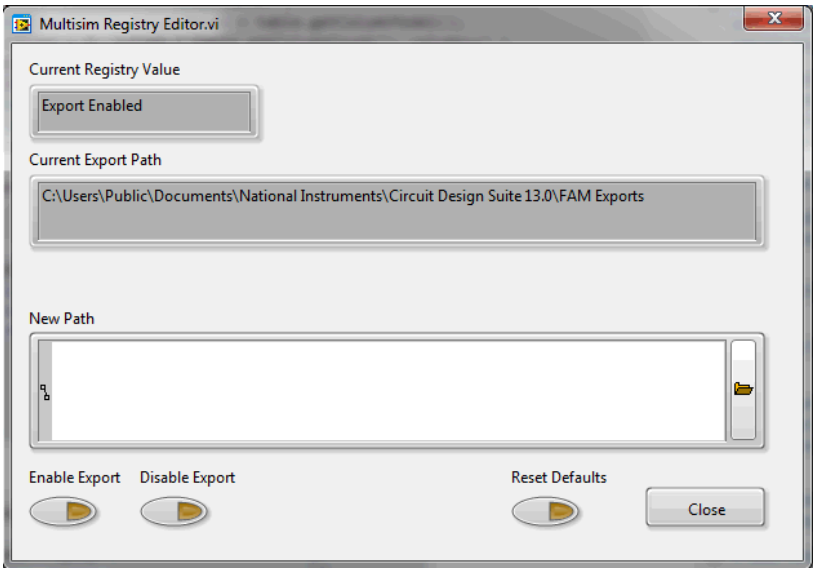
Follow the following steps to enable the NI Multisim v13 to export the XML file related to the NI Multisim model.

1. After installing the eHS package, navigate to the directory *PSIM_Multisim_SPS_Models.*

2. Right click on *Multisim Export Setup.exe* and select Run as administrator.



3. Click on the **Enable Export** button

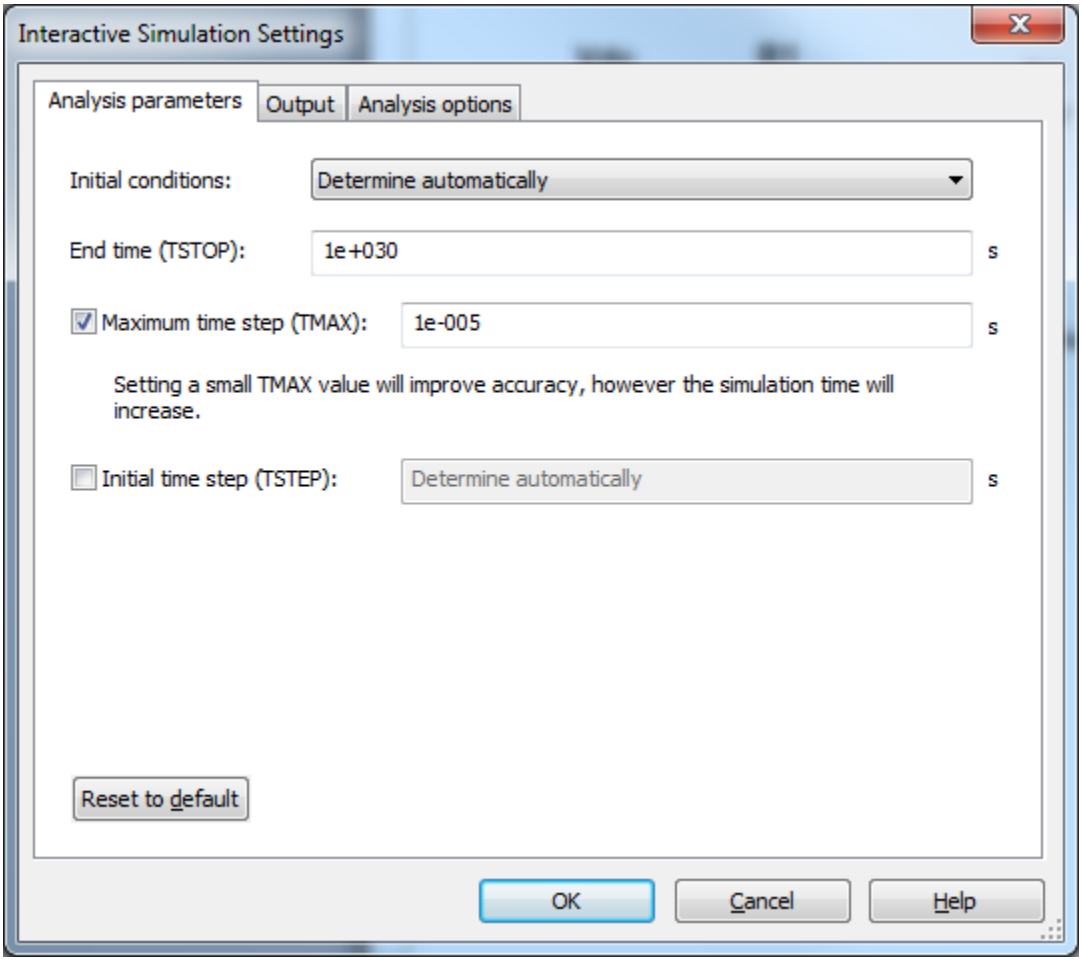


4. Click on the folder icon and browse to the *…/Psim_Multisim_SPS_Models* folder where you want the matrix exports to be saved.

The XML data is exported whenever a simulation is started.

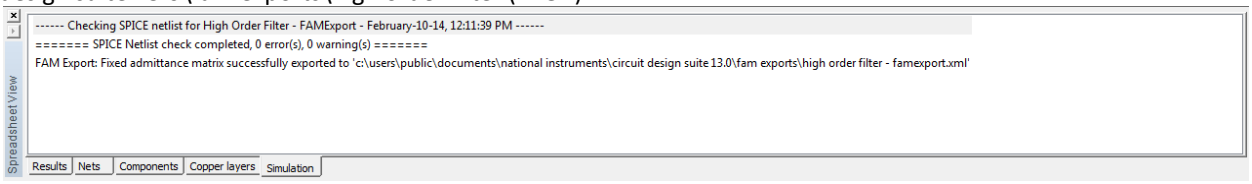
The easiest way to invoke a simulation is to press the green run icon button in the toolbar:

The simulator may not respond correctly because it cannot determine a time step. Go into **Simulate>Interactive Simulation Settings** and enter any time step value:



Once you press Run, it will switch to the simulation tab and show the following message, if you have spreadsheet view enabled:

FAM Export: Fixed admittance matrix successfully exported to 'c:\users\public\documents\national instruments\circuit design suite 13.0\fam exports\high order filter (FPGA).xml'



The exported XML file is ready.

You can press the Stop button immediately (since no meaningful simulation results are produced in NI Multisim using the exportable models).

CONTACT

OPAL-RT Corporate Headquarters

1751 Richardson, Suite 2525
Montréal, Québec, Canada
H3K 1G6
Tel.: 514-935-2323
Toll free: 1-877-935-2323

Technical Services
www.opal-rt.com/support

Note:

While every effort has been made to ensure accuracy in this publication, no responsibility can be accepted for errors or omissions. Data may change, as well as legislation, and you are strongly advised to obtain copies of the most recently issued regulations, standards, and guidelines.
This publication is not intended to form the basis of a contract.