Establishing a Software-Based Real-Time Simulation Platform for a Controls Laboratory for Training, Research and Development, and Experimentation

Final Project Report

Power Systems Engineering Research Center

Empowering Minds to Engineer the Future Electric Energy System

Establishing a Software-Based Real-Time Simulation Platform for a Controls Laboratory for Training, Research and Development, and Experimentation

Final Project Report

Project Team

Ali Mehrizi-Sani, Project Leader
Saleh Ziaeinejad
Washington State University

PSERC Publication 15-01

August 2015

For information about this project, contact

Ali Mehrizi-Sani
School of Electrical Engineering and Computer Science
Energy Systems Innovation Center (ESIC)
Laboratory for Integration of Power Electronics (LIPE)
Washington State University
EME 35 - 355 NE Spokane Street
Pullman, WA 99164-2752

Tel: +1 (509) 335-6249
Fax: +1 (509) 335-3818
Email: mehrizi@eecs.wsu.edu
Webpage: http://eecs.wsu.edu/~mehrizi

Power Systems Engineering Research Center

The Power Systems Engineering Research Center (PSERC) is a multi-university Center conducting research on challenges facing the electric power industry and educating the next generation of power engineers. More information about PSERC can be found at the Center's website: http://www.pserc.org.

For additional information, contact:

Power Systems Engineering Research Center
Arizona State University
527 Engineering Research Center
Tempe, Arizona 85287-5706
Phone: 480-965-1643
Fax: 480-965-0745

Notice Concerning Copyright Material

PSERC members are given permission to copy without fee all or part of this publication for internal use if appropriate attribution is given to this document as the source material. This report is available for downloading from the PSERC website.

© 2015 Washington State University. All rights reserved.

Acknowledgements

This is the final report for the Power Systems Engineering Research Center (PSERC) research project titled "Establishing a Software-Based Real-Time Simulation Platform for a Controls Laboratory for Training, Research and Development, and Experimentation" (project T-54G). We express our appreciation for the support provided by PSERC industry members and by the National Science Foundation under the Industry / University Cooperative Research Center program.

We wish to thank our industry advisors, Juan Castandea and Andy Paylan from Southern California Edison. We also thank Om Nayak (Nayak Corporation) and Craig Muller and Dharshana Muthumuni (Manitoba HVDC Research Centre) for the discussions about PSCAD/EMTDC software tool.

Executive Summary

A distribution system may include distributed energy resources (DER), capacitors, and power electronic devices such as active filters and power factor correctors. A central controller has a supervisory role over the local controllers. In the process of designing the central and local controllers, one key requirement is their evaluation with simulation tools. Traditionally, the controllers are evaluated with either offline or real-time simulators. The first approach is not comprehensive since it can not interface to the physical controller hardware, and the second approach is expensive.

This project develops a software-based real-time simulator, which is a simple but useful tool for evaluating the controller hardware that will eventually implement the control algorithms in the field. In this project, the distributed system is simulated in PSCAD (an offline power system simulator) and the control algorithms are implemented in physical controller hardware. Two structures are developed for interfacing the simulation environment with the controller hardware. An algorithm is proposed that enforces PSCAD to run in real-time. Therefore, the developed interfacing structure and real-time enforcement scheme provide a hardware-in-the-loop (HIL) real-time simulation tool that can test the performance of controllers in a system.

Table of Contents

1 Introduction **1**

1.1 Statement of Problem . . . 1

1.2 Research Objectives . . . 2

1.3 Organization of Report . . . 2

2 Software-Based Real-Time Simulation **3**

2.1 Software Comparison . . . 3

2.2 Study System . . . 3

2.3 HIL Simulation Using PSCAD, MATLAB, and LabVIEW . . . 5

2.4 HIL Simulation Using PSCAD and LabVIEW . . . 8

2.5 Synchronizing Real-Time Simulation . . . 14

3 HIL Real-Time Simulation Results **16**

3.1 Results of the HIL Real-Time Simulation Case Studies . . . 16

3.2 Video Demonstration of the Simulation Case Studies . . . 20

4 Conclusion **23**

4.1 Contributions and Conclusions . . . 23

4.2 Future Work . . . 23

A Datasheets **25**

References **33**

List of Figures

2.1 SCE distribution feeder chosen for the case studies. 4
2.2 Reduced SCE feeder. 5
2.3 Comparison of the real and reactive powers (P_s and Q_s) supplied to the original and reduced SCE feeders when all three capacitors are switched off: (a) the original feeder; (b) the reduced feeder. 6
2.4 Comparison of the real and reactive powers (P_s and Q_s) supplied to the original and reduced SCE feeders when all three capacitors are switched on: (a) the original feeder; (b) the reduced feeder. 6
2.5 HIL simulation using MATLAB as the interface. 7
2.6 Connecting MATLAB to PSCAD using a custom component. 7
2.7 Generation of an impulse train signal to trigger the MATLAB-PSCAD interface component. 8
2.8 Fortran code to interface MATLAB to PSCAD. 9
2.9 Sample MATLAB code interfaced to PSCAD. 10
2.10 Using a LabVIEW layout as the interface of the text files and DAQ module: (a) reading from text files and writing to DAQ module; (b) reading from DAQ module and writing to the text files. 11
2.11 Custom PSCAD component to manipulate text files. 11
2.12 Fortran code to manipulate text files. 12
2.13 HIL simulation without MATLAB. 13
2.14 HIL real-time simulation with multiple external controllers interfaced to the real-time simulator. 13
2.15 Fortran code to find the system time. 14
2.16 Timeline of the proposed real-time simulation. 15
2.17 Difference between the real time and the simulation time of a sample similation case study without and with the proposed real-time enforcing scheme. 15

3.1 Implemented setup of the proposed software-based HIL real-time simulation. 17
3.2 Using oscilloscope for observing the single phase voltage of a bus. 17
3.3 Simulated distribution feeder. 18
3.4 Hysteresis controllers implemented in the external controller. 18
3.5 NI-cRIO 9024 and input/output modules. 19
3.6 Hysteresis controllers implemented in NI-cRIO through a LabVIEW interface. 19

3.7 HIL real-time simulation results (readings of PSCAD plots): (a) voltages of the capacitor buses when the external controller is not interfaced to the simulation and both capacitors are switched off; (b) voltages of the capacitor buses when the external controller is interfaced to the simulation; (c) commands of the capacitor switches in the presence of the external controller (0 V means on, 5 V means off). 20
3.8 HIL real-time simulation results (measurements of the physical oscilloscope): (a) voltages of the capacitor buses when the external controller is not interfaced to the simulation and both capacitors are switched off (0.0125 pu/div); (b) voltages of the capacitor buses when the external controller is interfaced to the simulation (0.0125 pu/div); (c) commands of the capacitor switches in the presence of the external controller (0 V means on, 5 V means off) (2 V/div). 22

4.1 Improving the capabilities of the introduced HIL real-time simulation by using parallel computers to run a PSCAD file (recommended future work). 24

List of Tables

2.1 Comparison of three simulation programs for HIL real-time simulation 4

Chapter 1

Introduction

1.1 Statement of Problem

A distribution system may include distributed energy resources (DER), capacitors, and power electronic devices such as active filters and power factor correctors. Each of these components is controlled by a dedicated local controller [1], [2]. A central controller has a supervisory role over the local controllers [3], [4]. Design of the local controllers and the central controller is one of the key challenges for a distribution system operator [5].

One key question in the design of the local and central controllers is how to evaluate their performance. One simple solution is to use an offline simulation package such as MATLAB/Simulink, PSCAD/EMTDC, and DigSILENT PowerFactory. In an offline solution, the entire distribution system together with all the local and central controllers need to be implemented in the simulation environment [6]. Although this solution enables a primary evaluation of the controllers, its shortcomings are the following:

- An offline simulation software is unable to verify if a specific controller hardware is able to handle the computational burden of a given control algorithm [7]. Therefore, an offline simulator is not a useful benchmark tool for selecting the controller hardware.
- Most offline simulation software tools are unable to consider communication delays. The delays that occur in physical systems can affect the performance of the system and result in the failure of the designed controllers. Lack of the ability of an offline simulator to consider such nonidealities of real systems makes it an unreliable measure of the effectiveness of the controllers.

To devise a solution for these shortcomings, hardware-in-the-loop (HIL) real-time simulation is employed. In HIL real-time simulation, the power system is simulated in a real-time simulator and the control algorithm is simulated in the physical hardware that will eventually implement the controllers in the field. The real-time simulator and the controller hardware are interfaced through digital and analog input/output (I/O) channels as necessary and applicable [8].

In HIL real-time simulation, a key requirement is that the simulation time should be kept the same as the wall clock time (real time). If the simulation time runs slower or faster than real time, its results are not valid. This is especially important if the controller implements a control algorithm that takes actions based on the time (e.g., a PI controller).

Several companies are involved in developing real-time simulators. Examples are RTDS Technologies, Inc. (producer of RTDS simulator), Opal-RT Technologies (producer of OP4510, OP5600, and OP7020), and Typhoon HIL (producer of HIL 4 Series and HIL 6 Series). All the real-time simulators can be used to evaluate the performance of the controllers of a distribution system. However, two factors limit their application. First, since the software and hardware architecture of real-time simulators is complex, they tend to be expensive [9]. Second, since the real-time simulators are designed to perform a single task at a time, their availability is limited. Since the utilities tend to use a purchased real-time simulator for several purposes (e.g., training, research, and development), this limitation in availability hampers the usefulness of the commercial real-time simulators.

1.2 Research Objectives

The goal of this project is to develop a software-based solution for HIL real-time evaluation of the local and central controllers of a distribution system. The developed software-based real-time simulator is able to interface to external controllers and verify their performance.

1.3 Organization of Report

The rest of this report is organized as follows: Chapter 2 compares different offline power system simulators for selection of the most suitable software tool for integration with external controllers. With the chosen simulation software, an example distribution system is modeled, and the reason and the approach for reducing the size of this system are discussed. Two methods for interfacing the simulation environment to external controllers are introduced, and their advantages and disadvantages are discussed. Also a method for synchronising the real-time simulation is proposed.

Chapter 3 presents the results of an HIL real-time simulation case study. In that case study, a control algorithm for a sample distribution system is implemented by an external controller that is interfaced to the simulation environment.

Chapter 4 presents the concluding remarks and recommends future work.

Chapter 2

Software-Based Real-Time Simulation

2.1 Software Comparison

In this section, the selection of a simulation tool as the base for the proposed software-based real-time simulator is discussed. Since the goal is to simulate the controllers of a distribution feeder, the suitable power system simulator needs to have the ability to model most power system components and systems. In addition, it should be able to simulate a power system in all operating conditioned (steady-state and transient). High speed of a simulation tool helps realization of the real-time simulation. In addition, user friendliness and ease of use are two assets for a simulation tool.

based on the mentioned criteria, several simulation tools are compared. Among them, three are chosen for a more detailed comparison: MATLAB/Simulink, DigSILENT/PowerFactory, and PSCAD/EMTDC. Table 2.1 shows the results of this comparison.

MATLAB/Simulink is an easy-to-use general-purpose simulation program. However, it fails to accurately model some power system components (examples are dynamic loads and transformer configurations). In addition, it is relatively slow. Therefore, MATLAB/Simulink is not a good option for HIL real-time simulation.

DigSILENT/PowerFactory is a specialized power system simulator suitable for a variety of power system applications. It has the ability to perform 1) electromagnetic transient (EMT)-based simulation to study fast system dynamics and 2) quasi-stationary simulation to study slower dynamics. It has a built-in real-time simulation feature but this feature can not be used in EMT-based simulation. In EMT-based simulation, PowerFactory is relatively slow. The slowness hampers the usefulness of DigSILENT/PowerFactory for HIL real-time simulation.

PSCAD/EMTDC is a powerful EMT-based simulation program widely used in the industry, research centers, and educational institutes. It includes the models of most power system components and is the fastest simulation program compared to the other two. Because of its high speed, ability to simulate power system components and systems in both transient and steady state conditions, and ease of use and user-friendliness, PSCAD is chosen as the simulation tool for this project.

2.2 Study System

In this project, a distribution feeder of Southern California Edison (SCE) is chosen as the test system. Fig. 2.1 shows the feeder. It includes several balanced and unbalanced loads, distribution lines, and three switched capacitors.

Changing the state of the capacitor switches results in a change in the voltage profile of the system. Although this distribution feeder can be used for the evaluation of different controllers that

Table 2.1
Comparison of three simulation programs for HIL real-time simulation

Feature	Simulink	PowerFactory	PSCAD
Availability of power system components	Limited	Good	Good
Control blocks/functions	Extensive	Limited	Good
Parallel implementation	Yes	No	No
Developing new components	Yes	Yes	Yes
Steady state simulation	Yes	Yes	Yes
Dynamic simulation	Yes	Yes	Yes
Model transfer to RTDS	Difficult	Difficult	Easy
Compatibility with Opal-RT	Yes	No	No
User friendliness	Yes	No	Yes
Limitation in simulating large systems (Educational version)	No	Yes	Yes
Learning curve	Easy	Difficult	Medium
Capability to interface with other software	Good	Limited	Limited
Support team response	Normal	Normal	Fast
Simulation speed	Slow	Slow	Fast

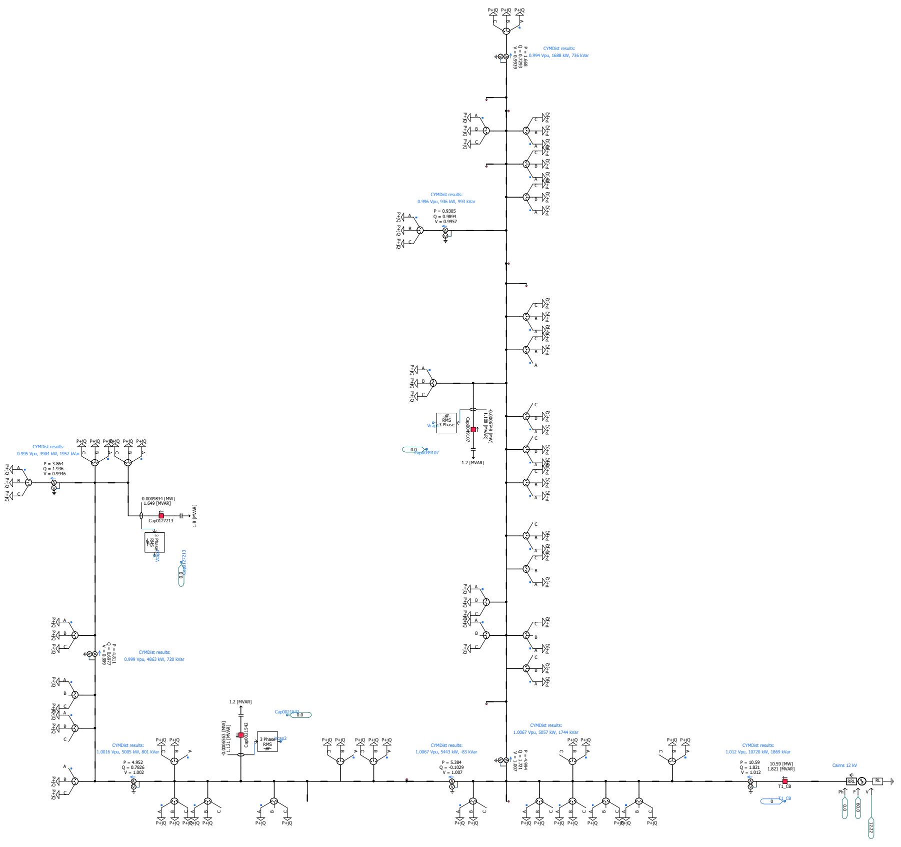


Figure 2.1: SCE distribution feeder chosen for the case studies.

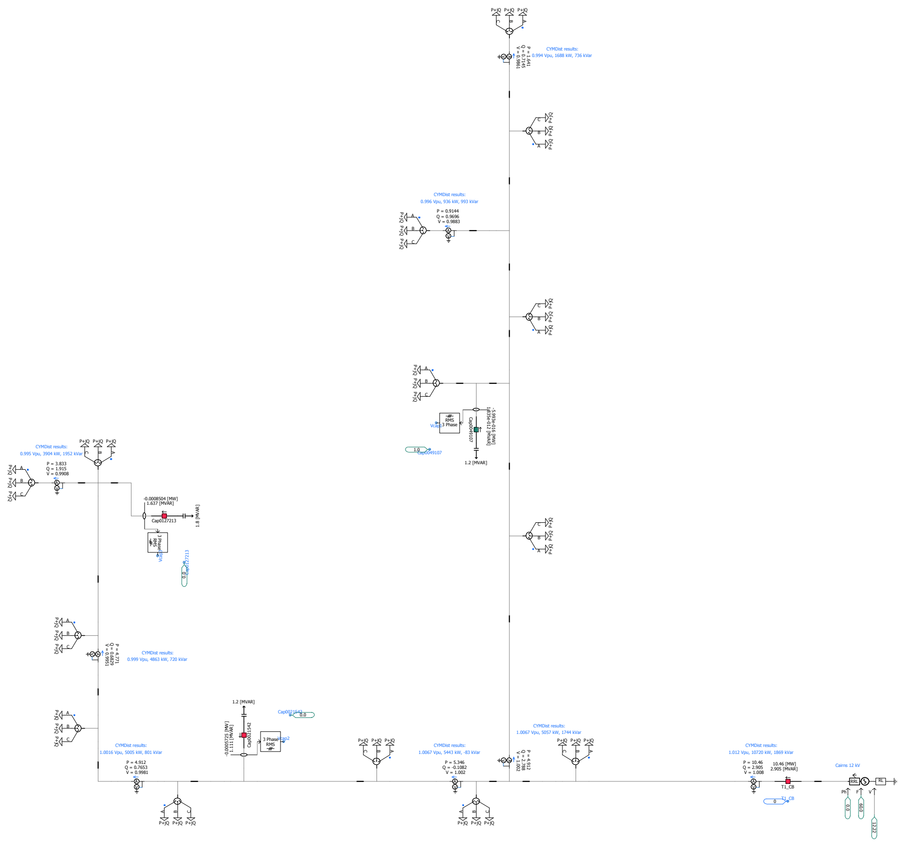


Figure 2.2: Reduced SCE feeder.

regulate the voltage, the large number of its nodes (324) poses a challenge for real-time simulation on a typical computer. In order to reduce the size of the system, the loads that are located close to each other are aggregated. The corresponding distribution lines are also merged. Fig. 2.2 shows the resulting system with 174 nodes. In our case studies, it is observed that the time needed to simulate the reduced system is almost half of the time needed for the original system.

While reducing the system size results in considerable reduction in simulation time, it is important to verify that the reduced and original systems behave similarly. For this purpose, two cases are studied. In the first case study, all the capacitors are switched off, and the real and reactive powers that the rest of the power system (modeled by an ideal voltage source behind an impedance) supplies to the distribution feeder are measured. Fig. 2.3 compares the results of the original and reduced feeders. In the second case study, all the capacitors are switched on, and again the supplied real and reactive powers are measured. Fig. 2.4 compares the results with the original and reduced feeders. Based on Figs. 2.3 and 2.4, the reduced system closely resembles the behavior of the original system.

2.3 HIL Simulation Using PSCAD, MATLAB, and LabVIEW

A software interface can facilitate the data exchange between PSCAD and the external controller. In this proposed approach, because of its ease of use, MATLAB is chosen as the software interface. Fig. 2.5 shows the overall structure of the proposed HIL simulation with MATLAB. MATLAB receives output PSCAD data (any required simulation variable) and writes each data point to a

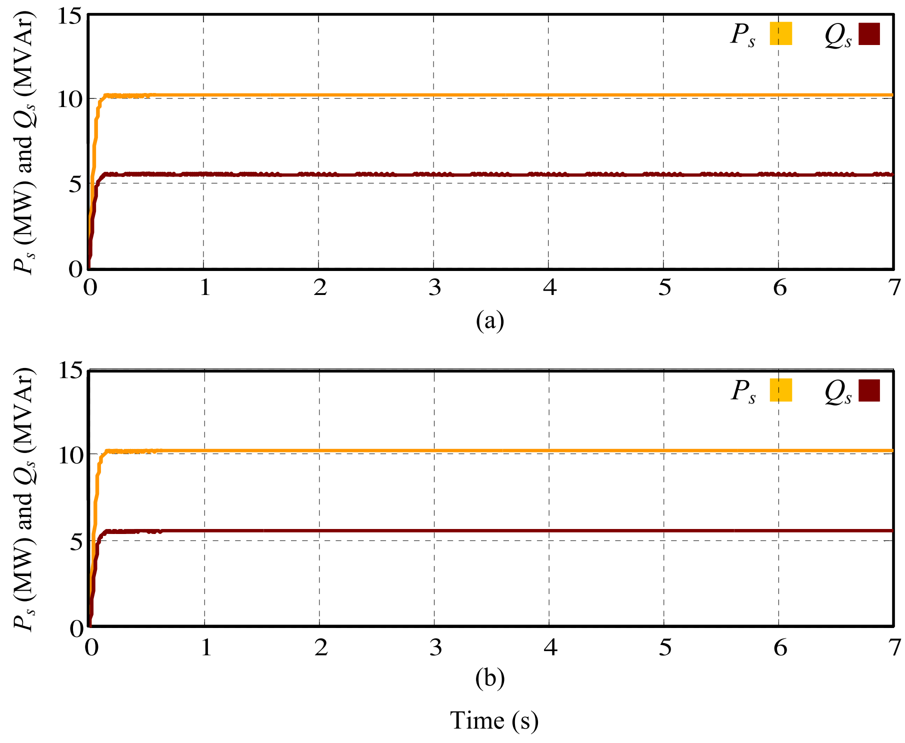


Figure 2.3: Comparison of the real and reactive powers (P_s and Q_s) supplied to the original and reduced SCE feeders when all three capacitors are switched off: (a) the original feeder; (b) the reduced feeder.

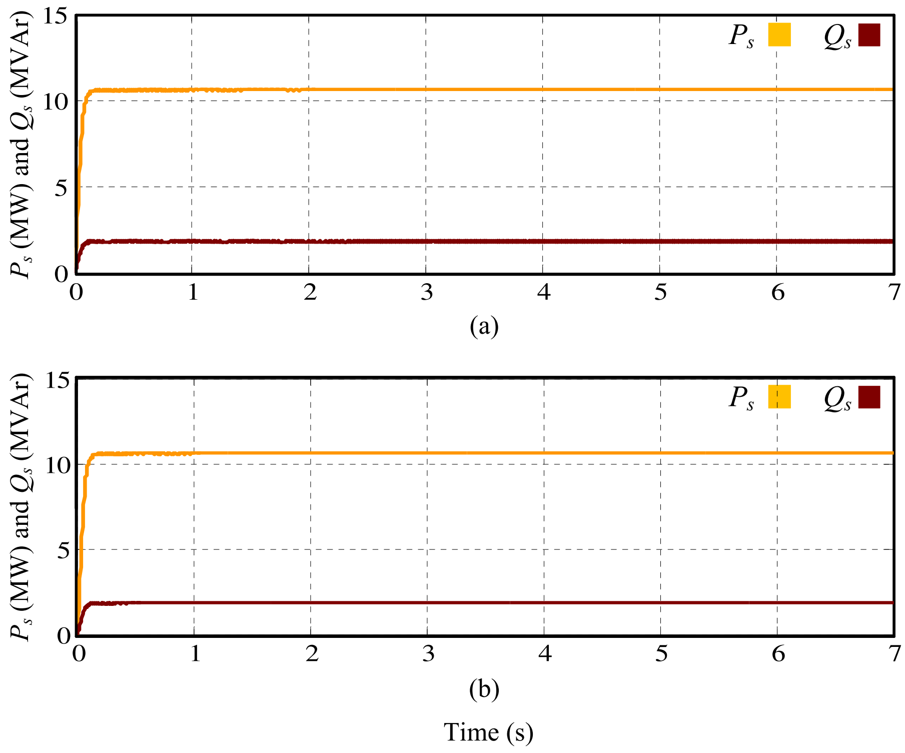


Figure 2.4: Comparison of the real and reactive powers (P_s and Q_s) supplied to the original and reduced SCE feeders when all three capacitors are switched on: (a) the original feeder; (b) the reduced feeder.

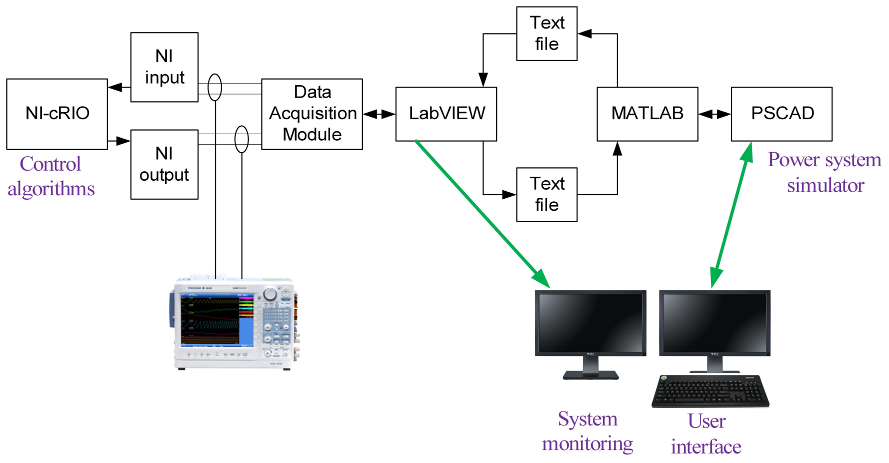


Figure 2.5: HIL simulation using MATLAB as the interface.

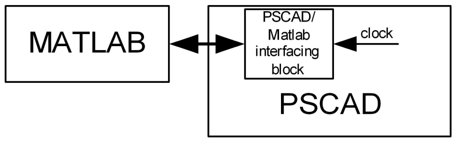


Figure 2.6: Connecting MATLAB to PSCAD using a custom component.

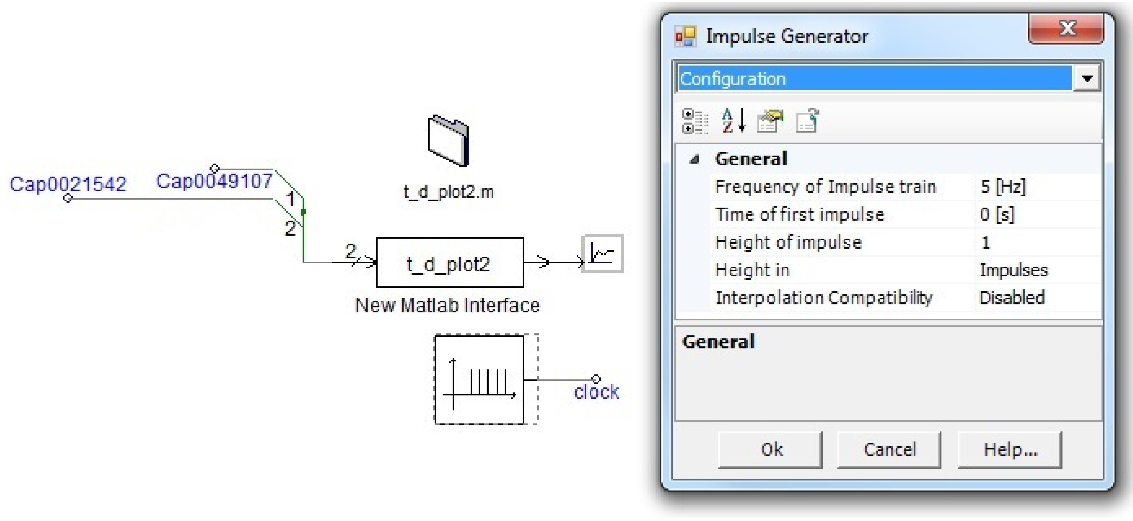
dedicated text file. These text files are used as the interface between MATLAB and LabVIEW (interface to a data acquisition [DAQ] module). After reading these files, LabVIEW writes their contents to the output channels of the DAQ module. The external controller then reads the data from the DAQ module

National Instruments NI-cRIO series is a commercial controller with applications in power system [10]. In this project, an NI-cRIO 9024 is chosen as the external controller. It can communicate with LabVIEW via NI input/output modules.

The output control commands of NI-cRIO are written to an NI output module, read by the DAQ module, and transferred to LabVIEW. LabVIEW writes each command in a dedicated text file. The text files are then read by MATLAB, and transferred to PSCAD. Oscilloscopes or voltmeters can read the physical signals that are taken from the simulation file (simulation variables) or the external controller (control commands).

Fig. 2.6 shows the connection of MATLAB to PSCAD. A custom PSCAD component is used to specify the number and the type of inputs and outputs of the MATLAB function that is being called. This MATLAB interface component is activated using a clock signal. In our studies, an impulse train with an adjustable frequency determines the rate of data exchange between PSCAD and MATLAB, as shown in Fig. 2.7. Increasing the frequency of this impulse train results in a more frequent data exchange between PSCAD and MATLAB and enables capturing fast transients. However, it also reduces the speed of simulation. The frequency of the impulse train signal is set appropriately to avoid slower-than-real-time simulation.

In our studies, the custom MATLAB interface component is designed to handle two inputs and two outputs of the MATLAB code. The associated Fortran code is shown in Fig. 2.8. All the inputs and outputs are defined as real numbers. Fig. 2.9 shows a sample MATLAB code that writes two signal values to text files and reads the contents of two other text files. Each signal can be a measurement (e.g., voltage, current, and real or reactive power) of a node of the simulated



General

Frequency of Impulse train	5 [Hz]
Time of first impulse	0 [s]
Height of impulse	1
Height in	Impulses
Interpolation Compatibility	Disabled

General

Figure 2.7: Generation of an impulse train signal to trigger the MATLAB-PSCAD interface component.

system. The number of input and output ports of the interface component can change based on the requirements of the external controller (the number of required measurements and the number of control commands). Increasing the number of ports results in a slower communication.

Fig. 2.10 shows the LabVIEW layout that is used for interfacing the text files and the DAQ module. The selection of the DAQ module depends on the required number of inputs and outputs and also the required communication rate. In this project, an NI-6009 DAQ module is used. It has twelve digital inputs/outputs, eight analog inputs, and two analog outputs.

2.4 HIL Simulation Using PSCAD and LabVIEW

The structure shown in Fig. 2.5 (HIL simulation with MATLAB interface) is simple to implement because MATLAB is an advanced and user-friendly programming tool. Another advantage of MATLAB is its ability to read a text file without locking the file, hence allowing the other software (LabVIEW) to write on the same file simultaneously. However, the need to run MATLAB in addition to another software tool makes the simulation slow. The proposed solution is to establish a PSCAD-LabVIEW link without MATLAB. In this case, a custom PSCAD component is developed to read/write to text files. Fig. 2.11 shows a developed PSCAD component to read from two text files and write to two text files. Fig. 2.12 shows the associated Fortran code.

In order to control the rate of data exchange with the text files, the Fortran code shown in Fig. 2.12 allows triggering read/write actions by an external impulse train signal. Increasing the frequency of the impulse train results in a more frequent data exchange between PSCAD and text files and enables capturing fast transients but it results in a slower simulation. Similar to Subsection 2.3, the frequency of the impulse train should be set to avoid slower-than-real-time simulation.

When the Fortran code of the custom component reads the content of a text file, that file can not be simultaneously modified by other software tool. When LabVIEW attempts to write to a file that is being read by Fortran, a LabVIEW error halts the simulation. The proposed solution uses an extra signal to prohibit LabVIEW from writing to a file that is being read by PSCAD.

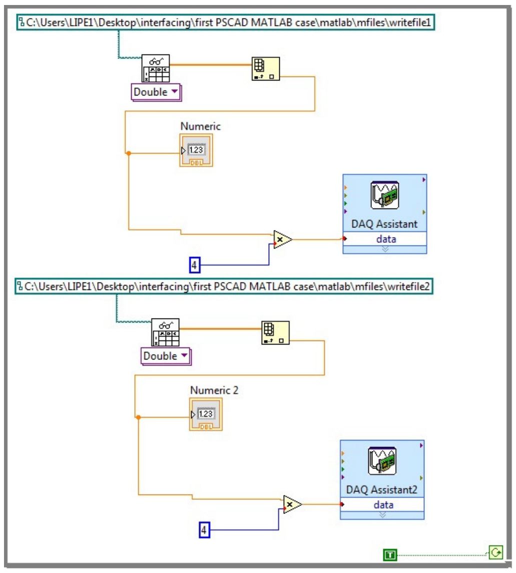
Fig. 2.13 shows the structure of HIL simulation without MATLAB. The connections between LabVIEW and DAQ, DAQ and NI input/output modules, and NI input/output modules and NI-

```
#STORAGE REAL:6
! ------------------------------------------------------------
! ------------------------------------------------------------
! PSCAD/EMTDC - MATLAB INTERFACE
! Module: $Name
#LOCAL INTEGER IVD1_1
      IF($Enabl.GT.0.9) THEN
! ------------------------------------------------------------
! Transfer EMTDC Input Variables to Matlab Interface
! ------------------------------------------------------------
!
! First Input Array (REAL(2))
       DO IVD1_1 = 1,2
          STORF(NSTORF+IVD1_1-1) = $INPUT(IVD1_1)
       END DO
!
! ------------------------------------------------------------
! Call PSCAD/EMTDC Matlab Interface:
! CALL MLAB_INT("MFILEPATH","MFILENAME","I/O Formats")
! ------------------------------------------------------------
       CALL COMPONENT_ID(ICALL_NO,$#Component)
       CALL MLAB_INT("%:Dir\$Path", "$Name", "R(2)" , "R" )
!
      ENDIF
! ------------------------------------------------------------
! Transfer Matlab Output Variables from Matlab Interface
! ------------------------------------------------------------
!
! First Output Array (REAL(1))
       DO IVD1_1 = 1,2
          $OUTPUT(IVD1_1)=STORF(NSTORF+IVD1_1+2)
       END DO
! Update STORx Pointers
      NSTORF = NSTORF + 6
! ------------------------------------------------------------
! ------------------------------------------------------------
```

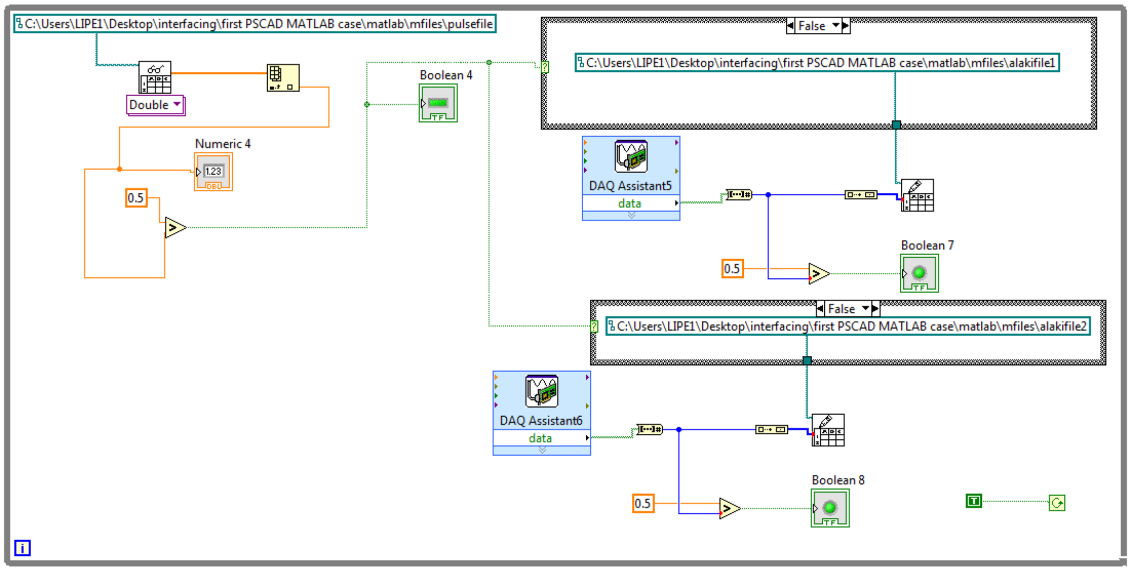
Figure 2.8: Fortran code to interface MATLAB to PSCAD.

```
function [out] = T_D_plot(in_2) %Defining the output size
global x y  w1 w2 %Parameter definition

%Reading the first text file
%If not blank, the first text file is stored.
x=csvread('readfile1');
w1=size(x);
w1=w1(1);
if w1>0
  csvwrite('auxfile1', x);
end

%Reading the second text file
%If not blank, the second text file is stored.
y=csvread('readfile2');
w2=size(y);
w2=w2(1);
if w2>0
  csvwrite('auxfile2', y);
end

%Sending the stored data to the output ports
out(1)=csvread('auxfile1');
out(2)=csvread('auxfile2');

%Writing the inputs (PSCAD variables) to text files
csvwrite('writefile1', in_3(1));
csvwrite('writefile2', in_3(2));
```

Figure 2.9: Sample MATLAB code interfaced to PSCAD.


C:\Users\LIPE1\Desktop\interfacing\first PSCAD MATLAB case\matlab\mfiles\writefile1
Double
Numeric
DAQ Assistant
data
4
C:\Users\LIPE1\Desktop\interfacing\first PSCAD MATLAB case\matlab\mfiles\writefile2
Double
Numeric 2
DAQ Assistant2
data
4

(a)


C:\Users\LIPE1\Desktop\interfacing\first PSCAD MATLAB case\matlab\mfiles\pulsefile
Double
Numeric 4
0.5
Boolean 4
False
C:\Users\LIPE1\Desktop\interfacing\first PSCAD MATLAB case\matlab\mfiles\alakifile1
DAQ Assistant5
data
0.5
Boolean 7
False
C:\Users\LIPE1\Desktop\interfacing\first PSCAD MATLAB case\matlab\mfiles\alakifile2
DAQ Assistant6
data
0.5
Boolean 8

(b)

Figure 2.10: Using a LabVIEW layout as the interface of the text files and DAQ module: (a) reading from text files and writing to DAQ module; (b) reading from DAQ module and writing to the text files.

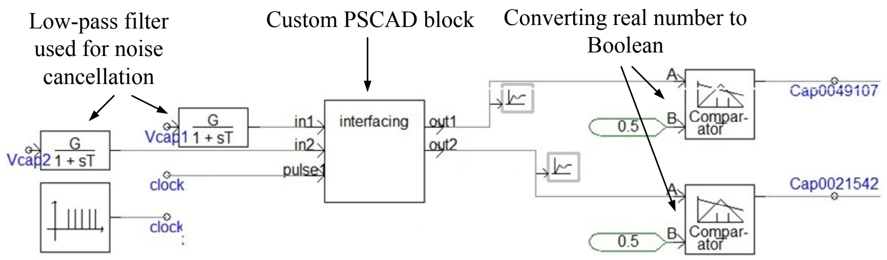


Figure 2.11: Custom PSCAD component to manipulate text files.

```fortran
if ($pulse1 .gt. 0.5) then ! Checking the clock signal

! Prohibiting LabVIEW write action
open(unit=206,file='C:\Users\LIPE1\Desktop\interfacing\first PSCAD MATLAB case\
    matlab\mfiles\pulsefile')
       write(206,*)1.1
       close(unit=206)

! Write actions
open(unit=104,file='C:\Users\LIPE1\Desktop\interfacing\first PSCAD MATLAB case\
    matlab\mfiles\writefile1')
       write(104,*)$in1
       close(unit=104)

open(unit=105,file='C:\Users\LIPE1\Desktop\interfacing\first PSCAD MATLAB case\
    matlab\mfiles\writefile2')
       write(105,*)$in2
       close(unit=105)

! Read actions
open(unit=101,file='C:\Users\LIPE1\Desktop\interfacing\first PSCAD MATLAB case\
    matlab\mfiles\readfile1')
       read(101,*,IOSTAT=IERR)$out1
       close(unit=101)

open(unit=102,file='C:\Users\LIPE1\Desktop\interfacing\first PSCAD MATLAB case\
    matlab\mfiles\readfile2')
       read(102,*,IOSTAT=IERR)$out2
       close(unit=102)

! Allowing LabVIEW write action
open(unit=206,file='C:\Users\LIPE1\Desktop\interfacing\first PSCAD MATLAB case\
    matlab\mfiles\pulsefile')
       write(206,*)0.1
       close(unit=206)

endif
```

Figure 2.12: Fortran code to manipulate text files.

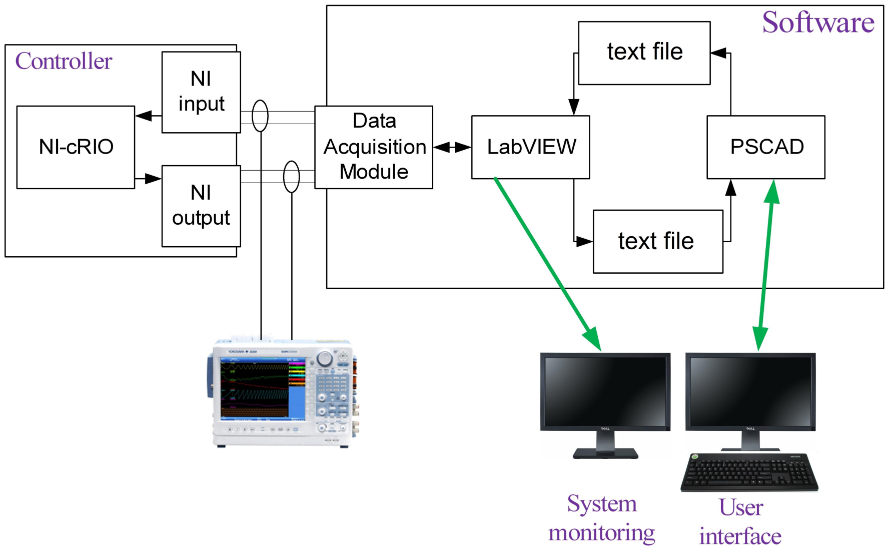


Figure 2.13: HIL simulation without MATLAB.

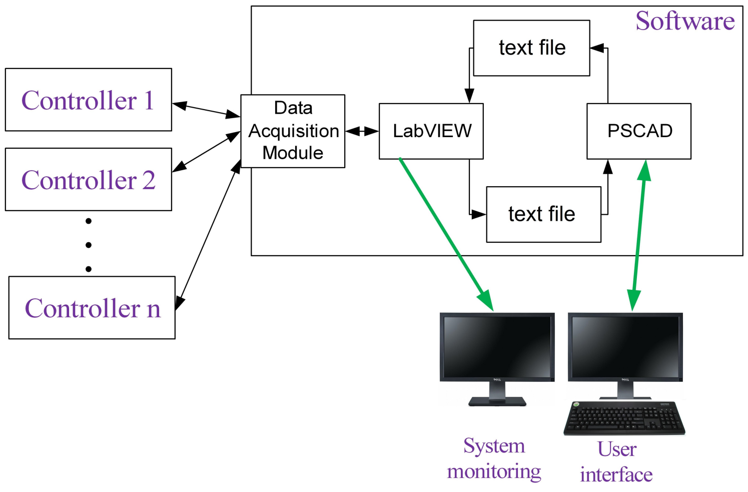


Figure 2.14: HIL real-time simulation with multiple external controllers interfaced to the real-time simulator.

cRIO controller are all similar to the case with MATLAB (Fig 2.5).

As discussed in Chapter 1, a distribution system operator simultaneously exploits multiple controllers. Therefore, the ability of the real-time simulator to simultaneously evaluate multiple external controllers with HIL real-time simulation results in a credible evaluation of the performance of a distribution system. The proposed software-based real-time simulator (with both structures shown in Figs. 2.5 and 2.13) is able to run HIL real-time simulation with multiple external controllers interfaced to the simulator. It should be noted that increasing the number of external controllers results in an increased number of input/output ports of the PSCAD custom component and hence a reduced simulation speed and possibility of losing real-time performance. The maximum number of external controllers that are interfaced to the proposed HIL real-time simulator depends on the computing power available (for example, by employing a faster CPU, more high-speed RAM, and a solid-state drive, more external controllers can be interfaced). Fig. 2.14 shows interfacing multiple external controllers during simulation of a distribution system.

```
      SUBROUTINE AUX_CSUBB(out,outt)
      INTEGER values(8)
      REAL rTime,rTimemili

       call date_and_time(values=values)

       rTime=(values(5))*60.        ! Hours to minutes
       rTime=(rTime+values(6))*60. ! Minutes to seconds
       rTime=(rTime+values(7)) ! Seconds to milliseconds

       rTimemili=values(8)

       out=rTime
       outt=rTimemili

      END
```

Figure 2.15: Fortran code to find the system time.

2.5 Synchronizing Real-Time Simulation

The external controllers interfaced to PSCAD simulator use real-time operating systems (RTOS) that always run in real time. However, PSCAD itself can run slower or faster than real time. A crucial step toward realization of HIL real-time simulation is to synchronize and enforce PSCAD to run in real-time.

The first step to enforce a real-time simulation is to avoid slower-than-real-time simulation by appropriately choosing the simulation time step, plotting time step, and the rate of data exchange with the external controllers.

The second step is to slow the simulation down if it runs faster than real time. This requires knowing the simulation time and the real time, and hence the difference between these two. PSCAD has a built-in component that returns the simulation time. To find the real time (system time), a Fortran code (Fig. 2.15) is developed that reads the system time with accuracy of one millisecond. PSCAD uses an older version of Fortran (Fortran 88) and can not compile the developed code that includes accessing the real time. Therefore, the code is written using a newer version of Fortran (Fortran 95) and is linked to PSCAD.

In the developed real-time enforcement scheme, at the instances that data exchange with text files occurs, a developed custom PSCAD component checks whether the simulation time is ahead of the real time. If needed, it executes a mathematically complex command to slow down the simulation. Fig. 2.16 shows the timeline of the simulation process. After each step of regulating the time, the simulation time is the same as the real time, and the real-time simulation is enforced.

Fig. 2.17 shows the difference between the real time and the simulation time of a sample simulation case study (the study distribution feeder without external controllers) without and with the real-time enforcing scheme. Without real-time enforcement, the simulation runs faster than the real time and the difference between the two linearly increases with time. With the proposed real-time enforcement scheme, the difference between the real time and the simulation time is always around zero and real-time simulation is realized.

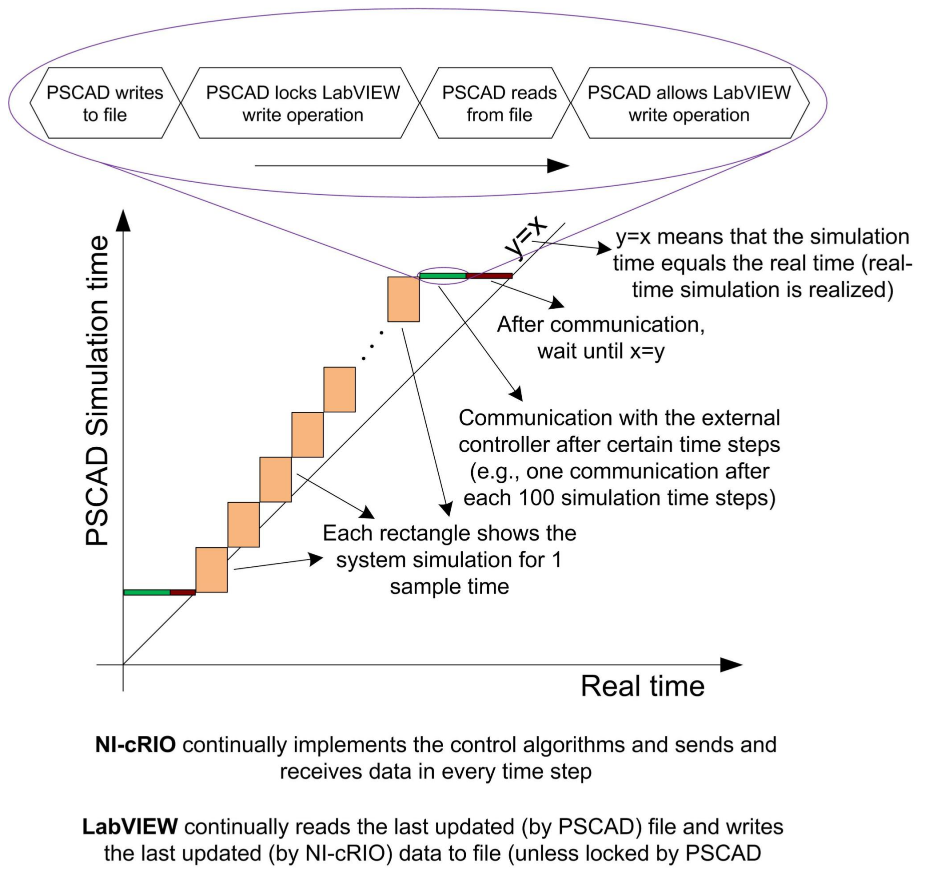

PSCAD writes to file
PSCAD locks LabVIEW write operation
PSCAD reads from file
PSCAD allows LabVIEW write operation
PSCAD Simulation time
y=x
y=x means that the simulation time equals the real time (real-time simulation is realized)
After communication, wait until x=y
Communication with the external controller after certain time steps (e.g., one communication after each 100 simulation time steps)
Each rectangle shows the system simulation for 1 sample time
Real time
NI-cRIO continually implements the control algorithms and sends and receives data in every time step
LabVIEW continually reads the last updated (by PSCAD) file and writes the last updated (by NI-cRIO) data to file (unless locked by PSCAD

Figure 2.16: Timeline of the proposed real-time simulation.

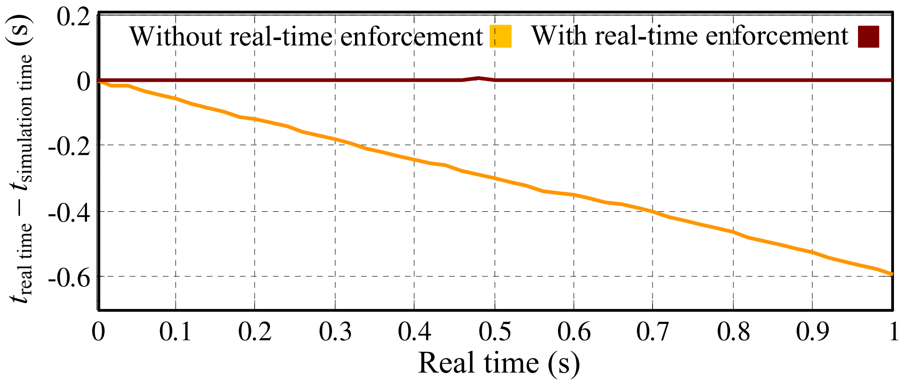


Figure 2.17: Difference between the real time and the simulation time of a sample similation case study without and with the proposed real-time enforcing scheme.

Chapter 3

HIL Real-Time Simulation Results

3.1 Results of the HIL Real-Time Simulation Case Studies

In order to evaluate the performance of the proposed software-based HIL real-time simulator, an experimental setup as shown in Fig. 3.1 is developed. Fig. 3.2 shows using a physical oscilloscope to measure the instantaneous value of a single phase voltage of a bus of the test feeder. Fig. 3.3 shows the test feeder that is the reduced SCE feeder augmented with three extra loads. The extra loads can be manually switched on and off. The control objective is to maintain the voltages of the buses of C_1 and C_2 within the specified limits. This objective can be realized by appropriate switching of C_1 and C_2.

Fig. 3.4 shows the diagram of a sample control algorithm that is implemented in NI-cRIO 9024 (the external controller). It compares the voltages of the capacitor buses with their corresponding reference values. Hysteresis controllers decide whether each capacitor should be switched on or off. If the voltage of a capacitor bus is above the maximum allowed value, the controller switches the associated capacitor off. If the voltage of a capacitor bus is below the minimum allowed value, the controller switches the associated capacitor on. If the voltage of a capacitor bus is within the allowed limits, the controller keeps the status of the associated capacitor switch unchanged.

In the implemented control algorithm, reference voltages of the capacitor buses are chosen to be $V^*_{C_1} = 0.9875$pu and $V^*_{C_2} = 0.9940$pu, and the bands of the hysteresis controllers are chosen to be $\Delta V_{C_1} = 0.0075$ pu and $\Delta V_{C_2} = 0.0040$ pu. With this selection of the reference voltages and the hysteresis bands, when the extra loads are switched on and off, the voltage of the capacitor buses violate the specified limits but can return to allowed limits if the external controller reacts appropriately.

Based on the diagram shown in Fig. 3.4, the external controller needs to read two analog signals from the simulation file (V_{C_1} and V_{C_2}) and send two digital commands to the simulation environment. Therefore, one analog input module and one digital output module are required. The analog input module that is used is an NI 9229, which has four input channels. The digital output module that is used is an NI 9474, which has eight output channels. Fig. 3.5 Shows NI-cRIO 9024 with two I/O modules mounted on its chassis and Fig. 3.6 shows the hysteresis controllers implemented in NI-cRIO through a LabVIEW interface.

In the simulation case studies, the HIL real-time simulation runs for 110 seconds. At $t = 70$ s, all the extra loads are switched on. Fig 3.7 shows the simulation results shown in PSCAD without and with the external controller interfaced to the simulator. Prior to $t = 70$ s, both C_1 and C_2 are switched off and their bus voltages are within the allowed limits ($V_{C_1} \in [0.98, 0.995]$, $V_{C_2} \in [0.990, 0.998]$.) After $t = 70$ s, switching the extra loads on results in a decrease in the voltages of

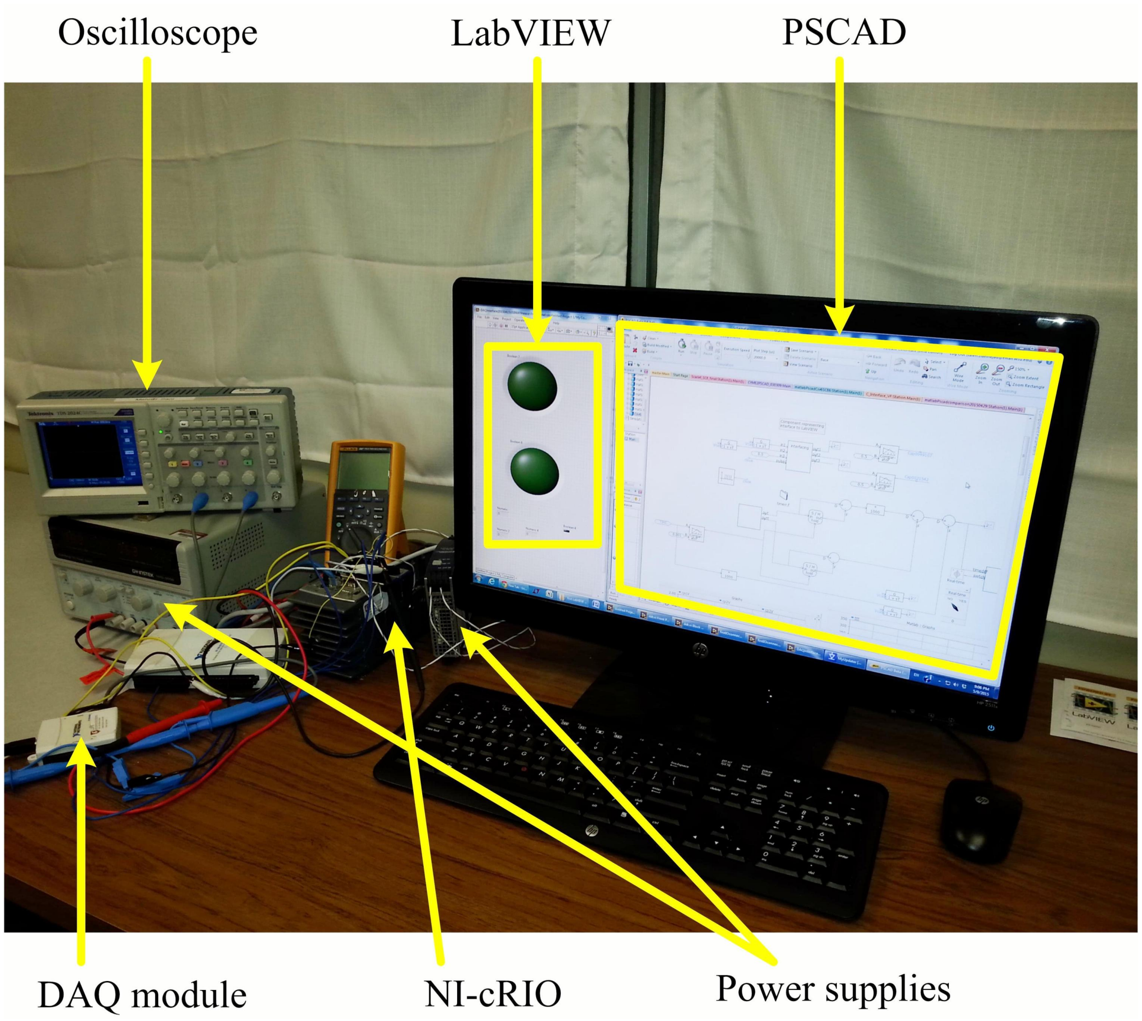


Figure 3.1: Implemented setup of the proposed software-based HIL real-time simulation.

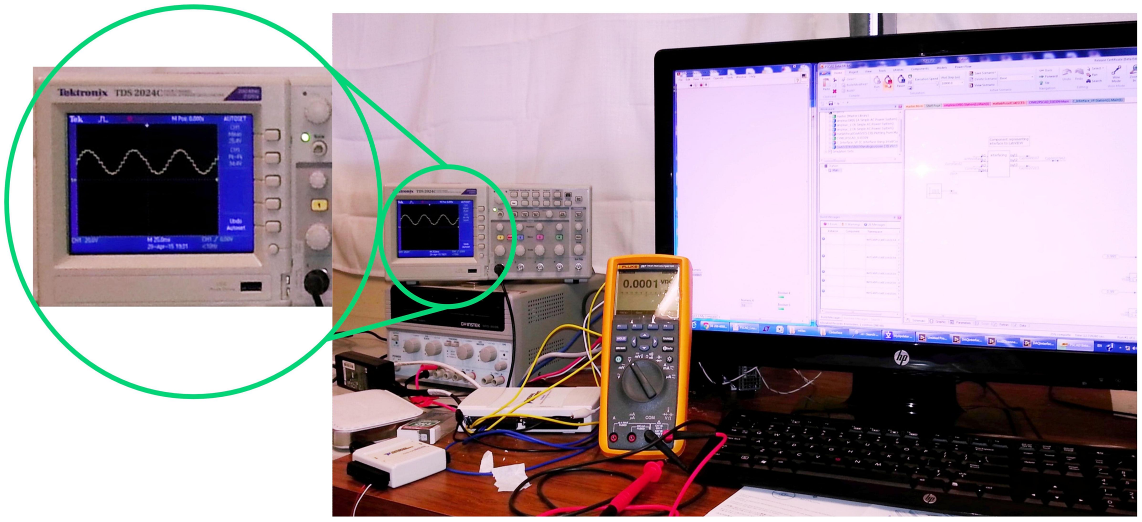


Figure 3.2: Using oscilloscope for observing the single phase voltage of a bus.

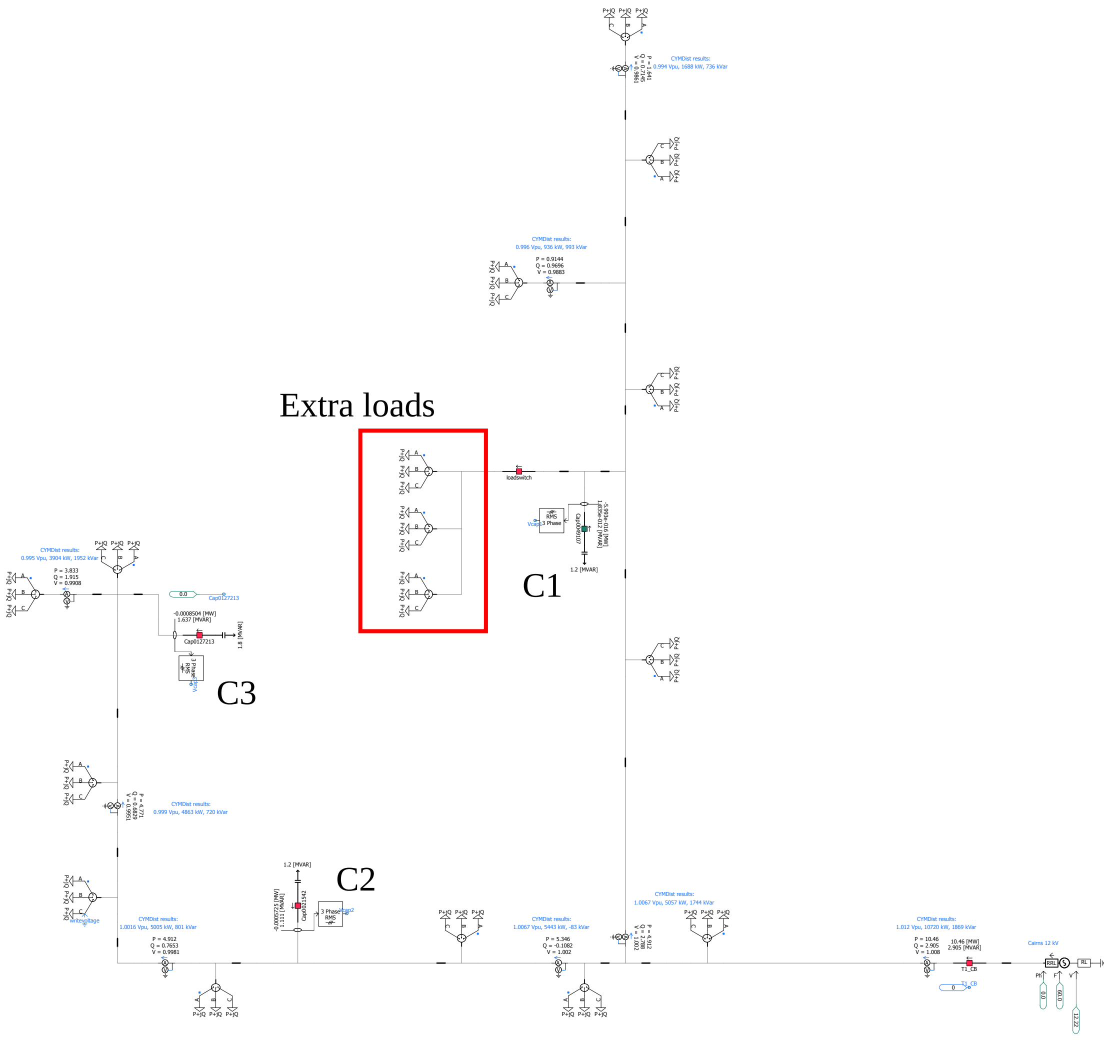


Figure 3.3: Simulated distribution feeder.

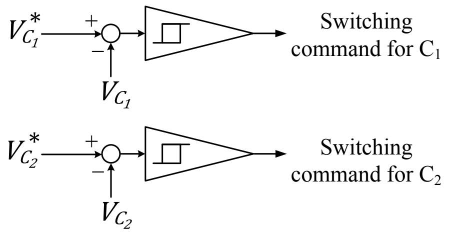


Figure 3.4: Hysteresis controllers implemented in the external controller.

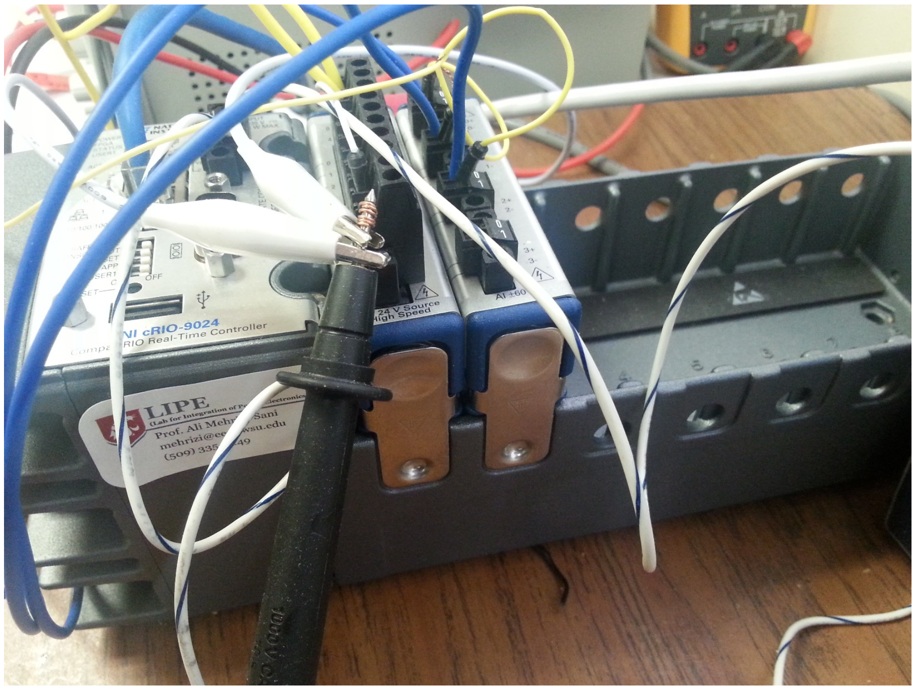


Figure 3.5: NI-cRIO 9024 and input/output modules.

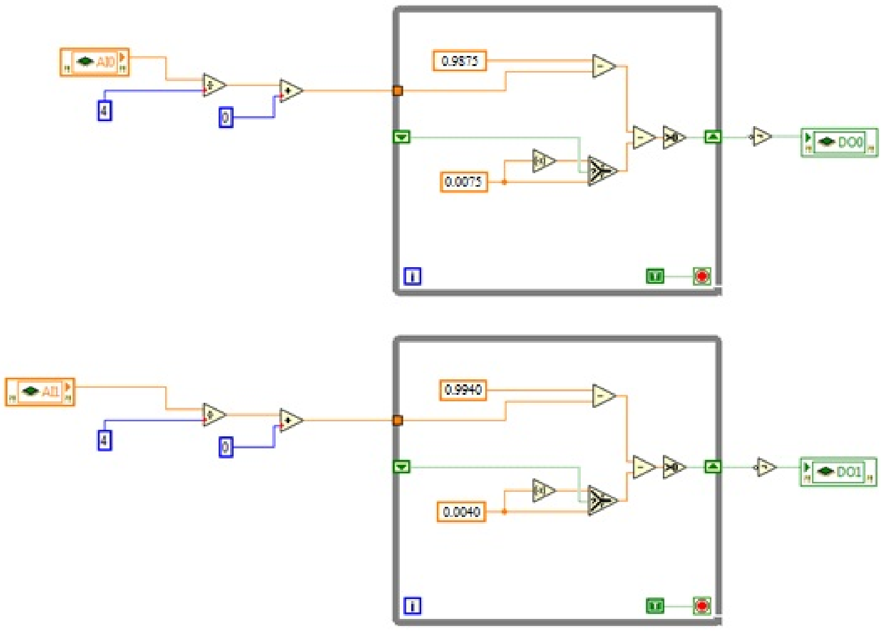


Figure 3.6: Hysteresis controllers implemented in NI-cRIO through a LabVIEW interface.

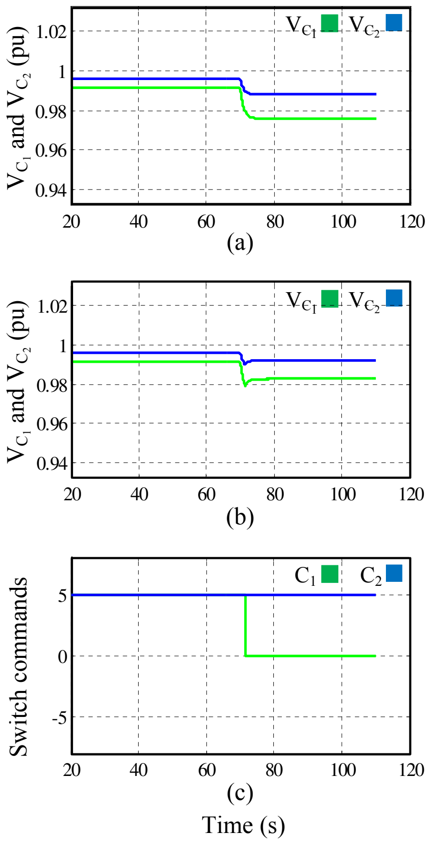


Figure 3.7: HIL real-time simulation results (readings of PSCAD plots): (a) voltages of the capacitor buses when the external controller is not interfaced to the simulation and both capacitors are switched off; (b) voltages of the capacitor buses when the external controller is interfaced to the simulation; (c) commands of the capacitor switches in the presence of the external controller (0 V means on, 5 V means off).

both capacitor buses. In the simulation case without the external controller, the voltages of both capacitor buses become less than the minimum allowed limits, Fig 3.7(a). In the simulation case with the external controller interfaced to the simulation, the controller (NI-cRIO) switches C_1 on. This action of the controller regulates the voltages of the capacitor buses within the allowed limits ($V_{C_1} = 0.9827$ pu and $V_{C_2} = 0.9923$ pu), Fig 3.7(b).

Fig. 3.8 shows the results of the same simulation case study measured by a physical oscilloscope. Similarity of the results shown in Figs. 3.7 and 3.8 show that an external measuring devices (e.g., voltmeter and oscilloscope) can be used to measure the variables of the simulated distribution system when the HIL real-time simulation is running. The time scales of Fig. 3.8 are the same as those of Fig. 3.7. This shows the realization of the real-time simulation using the proposed structure for real-time enforcement.

3.2 Video Demonstration of the Simulation Case Studies

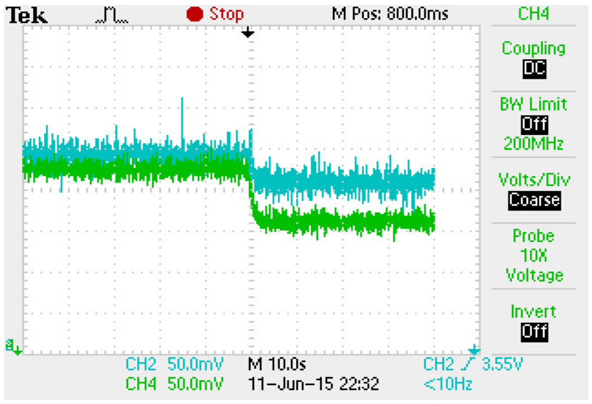
Different parts of the simulation case studies are recorded as video clips. This section briefly describes the contents of the recorded video clips.

In the first video clip, http://goo.gl/q9QSb5, the goal of the proposed software-based HIL real-time simulator is described and the hardware and software are introduced. In addition, the test distribution feeder is introduced, and the structure of the controllers that are used to regulate the voltages of this feeder is described.

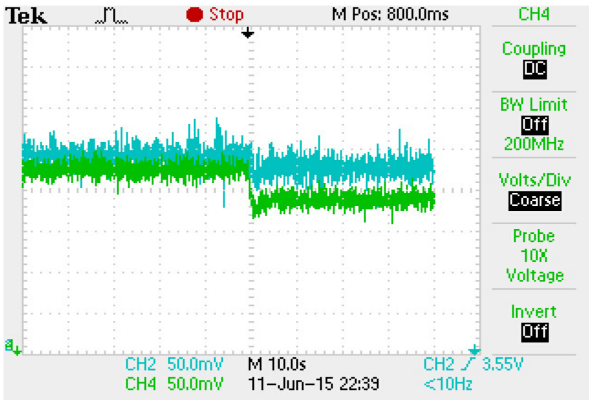
In the second video clip, http://goo.gl/f1ZWZS, an HIL real-time simulation case study is performed. In this simulation case study, MATLAB is used to interface the simulation environment (PSCAD) with LabVIEW (interface to the external controller). The simulation environment and the external controller exchange digital data. The methodology to interface the external controller to the simulation environment is described, and the process to evaluate the performance of the external controller is explained.

In the third video clip, http://goo.gl/VddIi4, the role of different data intarfaces (MATLAB, text files, DAQ module, and NI input/output modules) is discussed, and the process of the using these interfaces for data exchange between PSCAD and the external controller is explained in details.

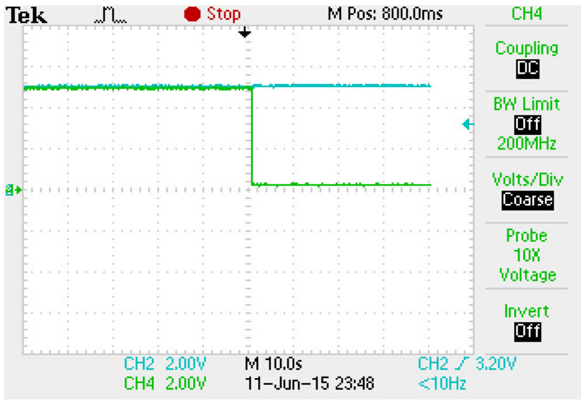
In the fourth video clip, http://goo.gl/2cFWJO, another simulation case study is performed. In this case study, PSCAD is directly interfaced to LabVIEW, and the two software tools exchange both analog and digital data. In this simulation case study, the effectiveness of the external controller in regulating the voltages of the test system is evaluated. In addition, it is shown that a physical oscilloscope can correctly measure the simulation variables.


Tek Stop M Pos: 800.0ms CH4
Coupling DC
BW Limit Off 200MHz
Volts/Div Coarse
Probe 10X Voltage
Invert Off
CH2 50.0mV M 10.0s CH2 ⨍ 3.55V
CH4 50.0mV 11-Jun-15 22:32 <10Hz

(a)


Tek Stop M Pos: 800.0ms CH4
Coupling DC
BW Limit Off 200MHz
Volts/Div Coarse
Probe 10X Voltage
Invert Off
CH2 50.0mV M 10.0s CH2 ⨍ 3.55V
CH4 50.0mV 11-Jun-15 22:39 <10Hz

(b)


Tek Stop M Pos: 800.0ms CH4
Coupling DC
BW Limit Off 200MHz
Volts/Div Coarse
Probe 10X Voltage
Invert Off
CH2 2.00V M 10.0s CH2 ⨍ 3.20V
CH4 2.00V 11-Jun-15 23:48 <10Hz

(c)

Figure 3.8: HIL real-time simulation results (measurements of the physical oscilloscope): (a) voltages of the capacitor buses when the external controller is not interfaced to the simulation and both capacitors are switched off (0.0125 pu/div); (b) voltages of the capacitor buses when the external controller is interfaced to the simulation (0.0125 pu/div); (c) commands of the capacitor switches in the presence of the external controller (0 V means on, 5 V means off) (2 V/div).

Chapter 4

Conclusion

4.1 Contributions and Conclusions

This project develops a software-based HIL real-time simulation tool that can be used for evaluating the controllers of a distribution system. Two different structures for interfacing the external controllers to the simulation environment are proposed. The first structure uses MATLAB for interfacing. This structure is simple to implement, but it is relatively slow. The second structure establishes a direct link between the simulation environment and LabVIEW (interface to the external controllers). Because of the requirement for running the simulation study in real-time, an algorithm for synchronizing and enforcing the real-time simulation is proposed. The proposed algorithm compares the simulation time with real time and slows the simulation down if needed.

An SCE distribution feeder is chosen for the studies. First, by aggregating the loads and the distribution lines, the the number of nodes of the feeder system is reduced. This reduction increases the simulation speed and facilitates the real-time simulation. The presented simulation case studies show the similarity of the performance of the original system and the reduced one and hence the validity of the use of the reduced feeder as the test system.

The goal of the control algorithm is to regulate the bus voltages by appropriately switching the capacitors of the distribution system on and off. In the simulation case studies, the proposed HIL real-time simulator is used to evaluate the performance of a sample hysteresis-based voltage control algorithm implemented in an external controller hardware (NI-cRIO 9024). The simulation case studies show the ability of the proposed HIL real-time simulator to verify the effectiveness of the studied hysteresis-based voltage controller.

4.2 Future Work

The possible future works can be listed as

Parallel processing: In order to obviate the limitation on processing the simulation file, parallel personal computers (PC) can be used to run a single simulation file. Each computer can run a part of the distribution feeder, and all the computers exchange the data. One computer should be responsible for communicating with the external controller through a LabVIEW interface and DAQ module. Fig. 4.1 shows the plan. The key challenge in implementing the structure shown in Fig. 4.1 is the separation of a PSCAD simulation file into several subsystems. All these subsystems should run with the same pace, and should be managed by a central real-time enforcing scheme. The communication pattern and the management of the interfacing algorithm needs investigation.

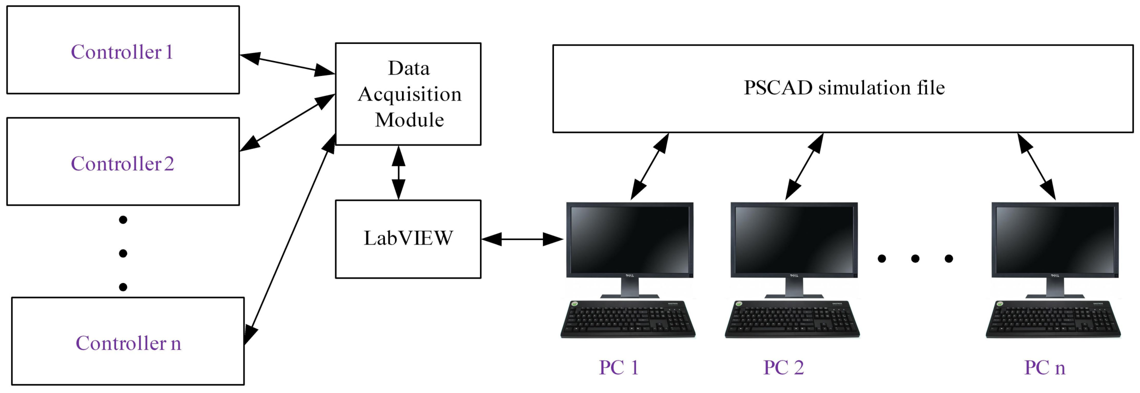


Figure 4.1: Improving the capabilities of the introduced HIL real-time simulation by using parallel computers to run a PSCAD file (recommended future work).

Power hardware in the loop: The capability of performing power hardware in the loop (PHIL) adds to the validity of the real-time simulation results. The challenges and possibilities of a software-based power-hardware-in-the-loop (PHIL) real-time simulator can be investigated.

On-line enforcement of real-time: A structure that synchronizes and enforces real-time simulation by on-line adaptation of the simulation settings can be proposed. This sructure can obviate the need for the first step of the two-step real-time enforcement scheme that is proposed in this report.

Evaluation of remote controllers: Interfacing an external hardware to a remote computer can be investigated. This realizes a remote assessment of the distribution system controllers.

Waveform relaxation: Waveform relaxation (WR) is a time-domain method used for the analysis of large, nonlinear dynamical systems. WR method decomposes the large system into several decoupled subsystems and uses an iterative method, such as Newton-Raphson method, to analyze the subsystems for the entire simulation interval [11]. The simulated distribution system and the external controller that is interfaced to it form a large system of nonlinear equations. A waveform relaxation–based method can be developed to solve this system.

Communicating through ports: In this report, the software-based real-time simulator and the external controllers communicate via input/output modules. A software-based HIL real-time simulation can be investigated in which the real-time simulator and the external controllers communicate through ports. This obviates the need for input/output modules and hence simplifies the structure of the HIL real-time simulation.

Appendix A

Datasheets

This appendix provides the datasheets of the hardware tools that are used in this project.

Low-Cost, Bus-Powered Multifunction DAQ for USB – 12- or 14-Bit, up to 48 kS/s, 8 Analog Inputs

NI USB-6008, NI USB-6009

- 8 analog inputs at 12 or 14 bits, up to 48 kS/s
- 2 analog outputs at 12 bits, software-timed
- 12 TTL/CMOS digital I/O lines
- 32-bit, 5 MHz counter
- Digital triggering
- Bus-powered
- 1-year warranty

Operating Systems

- Windows Vista (32- and 64-bit)/XP/2000
- Mac OS X[1]
- Linux®[1]
- Windows Mobile[1]
- Windows CE[1]

Recommended Software

- LabVIEW
- LabVIEW SignalExpress
- LabWindows™/CVI
- Measurement Studio

Other Compatible Software

- C#, Visual Basic .NET
- ANSI C/C++

Measurement Services Software (included)

- NI-DAQmx driver software
- Measurement & Automation Explorer configuration utility
- LabVIEW SignalExpress LE

[1]You need to download NI-DAQmx Base for these operating systems.

Product	Bus	Analog Inputs[1]	Input Resolution (bits)	Max Sampling Rate (kS/s)	Input Range (V)	Analog Outputs	Output Resolution (bits)	Output Rate (Hz)	Output Range (V)	Digital I/O Lines	32-Bit Counter	Trigger
USB-6009	USB	8 SE/4 DI	14	48	±1 to ±20	2	12	150	0 to 5	12	1	Digital
USB-6008	USB	8 SE/4 DI	12	10	±1 to ±20	2	12	150	0 to 5	12	1	Digital

[1]SE = single ended, DI = differential [2]Software-timed

Overview and Applications

With recent bandwidth improvements and new innovations from National Instruments, USB has evolved into a core bus of choice for measurement applications. The NI USB-6008 and USB-6009 are low-cost entry points to NI flagship data acquisition (DAQ) devices. With plug-and-play USB connectivity, these modules are simple enough for quick measurements but versatile enough for more complex measurement applications.

The USB-6008 and USB-6009 are ideal for a number of applications where low cost, small form factor, and simplicity are essential. Examples include:

- Data logging – quick and easy environmental or voltage data logging
- Academic lab use – student ownership of DAQ hardware for completely interactive lab-based courses (Academic pricing available. Visit **ni.com/academic** for details.)
- OEM applications as I/O for embedded systems

Recommended Software

National Instruments measurement services software, built around NI-DAQmx driver software, includes intuitive application programming interfaces, configuration tools, I/O assistants, and other tools designed to reduce system setup, configuration, and development time. National Instruments recommends using the latest version of NI-DAQmx driver software for application development in NI LabVIEW, LabVIEW SignalExpress, LabWindows/CVI, and Measurement Studio software. To obtain the latest version of NI-DAQmx, visit **ni.com/support/daq/versions**.

NI measurement services software speeds up your development with features including:

- A guide to create fast and accurate measurements with no programming using the DAQ Assistant.
- Automatic code generation to create your application in LabVIEW.
- LabWindows/CVI; LabVIEW SignalExpress; and C#, Visual Studio .NET, ANSI C/C++, or Visual Basic using Measurement Studio.
- Multithreaded streaming technology for 1,000 times performance improvements.
- Automatic timing, triggering, and synchronization routing to make advanced applications easy.
- More than 3,000 free software downloads available at **ni.com/zone** to jump-start your project.
- Software configuration of all digital I/O features without hardware switches/jumpers.
- Single programming interface for analog input, analog output, digital I/O, and counters on hundreds of multifunction DAQ hardware devices. M Series devices are compatible with the following versions (or later) of NI application software – LabVIEW, LabWindows/CVI, or Measurement Studio versions 7.x; and LabVIEW SignalExpress 2.x.

Every M Series data acquisition device also includes a copy of LabVIEW SignalExpress LE data-logging software, so you can quickly acquire, analyze, and present data without programming. The NI-DAQmx Base driver software is provided for use with Linux, Mac OS X, Windows Mobile, and Windows CE operating systems.

Recommended Accessories

The USB-6008 and USB-6009 have removable screw terminals for easy signal connectivity. For extra flexibility when handling multiple wiring configurations, NI offers the USB-600x Connectivity Kit, which includes two extra sets of screw terminals, extra labels, and a screwdriver.

In addition, the USB-600x Prototyping Kit provides space for adding more circuitry to the inputs of the USB-6008 or USB-6009.

NI USB DAQ for OEMs

Shorten your time to market by integrating world-class National Instruments OEM measurement products into your embedded system design. Board-only versions of NI USB DAQ devices are available for OEM applications, with competitive quantity pricing and available software customization. The NI OEM Elite Program offers free 30-day trial kits for qualified customers. Visit **ni.com/oem** for more information.

Information for Student Ownership

To supplement simulation, measurement, and automation theory courses with practical experiments, NI has developed the USB-6008 and USB-6009 student kits, which include the LabVIEW Student Edition and a ready-to-run data logger application. These kits are exclusively for students, giving them a powerful, low-cost, hands-on learning tool. Visit **ni.com/academic** for more details.

Information for OEM Customers

For information on special configurations and pricing, call (800) 813 3693 (U.S. only) or visit **ni.com/oem**. Go to the Ordering Information section for part numbers.

Ordering Information

NI USB-6008[1]	779051-01
NI USB-6009[1]	779026-01
NI USB-6008 OEM	193132-02
NI USB-6009 OEM	193132-01
NI USB-6008 Student Kit[1],[2]	779320-22
NI USB-6009 Student Kit[1],[2]	779321-22
NI USB-600x Connectivity Kit	779371-01
NI USB-600x Prototyping Kit	779511-01

[1] Includes NI-DAQmx software, LabVIEW SignalExpress LE, and a USB cable.

[2] Includes LabVIEW Student Edition.

BUY NOW!

For complete product specifications, pricing, and accessory information, call 800 813 3693 (U.S. only) or go to **ni.com/usb**.

Last Revised: 2014-11-06 07:14:18.0

NI 9229, NI 9239

±10 V or ±60 V, Simultaneous Analog Input, 50 kS/s, 4 Ch Modules

- 4 differential channels, 50 kS/s per channel sample rate
- ±10 V (NI 9239) or ±60 V (NI 9229) measurement range, 24-bit resolution
- Antialias filter
- 250 Vrms ch-ch, CAT II (screw terminal), or 60 VDC ch-ch, CAT I (BNC) isolation
- Screw-terminal or BNC connectivity
- -40 °C to 70 °C operating, 5 g vibration, 50 g shock

Overview

The NI 9229 is a 4-channel, 24-bit C Series analog input module for use in any NI CompactDAQ or CompactRIO chassis. The NI 9239 is similar to the NI 9229 in functionality except for the input range. You can find detailed specifications for the NI 9229 and NI 9239 in the same manual for comparison.

With channel-to-channel isolation, your entire system, including the device under test, is protected from harmful voltage spikes up to the isolation rating. In addition to safety, isolation eliminates measurement errors caused by ground loops because the front end of the module is floating.
EMC Performance To ensure EMC compliance for BNC, you must use a ferrite bead, such as NI part number 782801-01.

Recommended Accessories
-NI 9971 strain relief and operator protection (for screw-terminal variant)
-EMI suppression ferrite for NI 9229/9239 BNC (for BNC variant)

Box Contents
-1 NI 9229 or NI 9239 C Series module
-1 NI 9229/9239 Operating Instructions and Specifications manual
-4 NI 9976 two-position screw-terminal connectors (for screw-terminal variant)

Back to Top

Comparison Tables

Product Name	Signal Ranges	Channels	Sample Rate	Simultaneous	Resolution	Isolation	Connectivity
NI 9201	±10 V	8 Single-Ended	500 kS/s/ch	No	12-Bit	250 Vrms Ch-Earth (Screw Terminal), 60 VDC Ch-Earth (D-SUB)	Screw Terminal, 25-Pin D-SUB
NI 9205	±200 mV, ±1 V, ±5 V, ±10 V	32 Single-Ended, 16 Differential	250 kS/s	No	16-Bit	250 Vrms Ch-Earth (Spring Terminal), 60 VDC Ch-Earth (D-SUB))	Spring Terminal, 37-Pin D-SUB
NI 9206	±200 mV, ±1 V, ±5 V, ±10 V	32 Single-Ended, 16 Differential	250 kS/s	No	16-Bit	600 VDC Ch-Earth	Spring Terminal
NI 9215	±10 V	4 Differential	100 kS/s/ch	Yes	16-Bit	250 Vrms Ch-Earth (Screw Terminal), 60 VDC Ch-Earth (BNC)	Screw Terminal, BNC
NI 9220	±10 V	16 Differential	100 kS/s/ch	Yes	16-Bit	250 Vrms Ch-Earth (Spring Terminal), 60 VDC Ch-Earth (D-SUB)	Spring Terminal, 37-Pin D-SUB
NI 9221	±60 V	8 Single-Ended	800 kS/s	No	12-Bit	250 Vrms Ch-Earth (Screw Terminal), 60 VDC Ch-Earth (D-SUB)	Screw Terminal, 25-Pin D-SUB

Product Name	Signal Ranges	Channels	Sample Rate	Simultaneous	Resolution	Isolation	Connectivity
NI 9222	±10 V	4 Differential	500 kS/s/ch	Yes	16-Bit	60 VDC Ch-Ch	Screw Terminal
NI 9223	±10 V	4 Differential	1 MS/s/ch	Yes	16-Bit	60 VDC Ch-Ch	Screw Terminal
NI 9229	±60 V	4 Differential	50 kS/s/ch	Yes	24-Bit	250 Vrms Ch-Ch (Screw Terminal), 60 VDC Ch-Ch (BNC)	Screw Terminal, BNC
NI 9239	±10 V	4 Differential	50 kS/s/ch	Yes	24-Bit	250 Vrms Ch-Ch (Screw Terminal), 60 VDC Ch-Ch (BNC)	Screw Terminal, BNC

Back to Top

Application and Technology

NI C Series Overview

NI C Series modules are engineered to provide high-accuracy measurements to meet the demands of advanced DAQ and control applications. Each module contains measurement-specific signal conditioning to connect to an array of sensors and signals, bank and channel-to-channel isolation options, and support for wide temperature ranges to meet a variety of application and environmental needs all in a single rugged package. You can choose from more than 100 C Series modules for measurement, control, and communication to connect your applications to any sensor on any bus.

Most C Series I/O modules work with both the NI CompactDAQ and NI CompactRIO platforms. The modules are identical, and you can move them from one platform to the other with no modification.

NI CompactRIO Platform

Powered by the NI LabVIEW reconfigurable I/O (RIO) architecture, NI CompactRIO combines an open embedded architecture with small size, extreme ruggedness, and hot-swappable industrial I/O modules. Each system contains an FPGA for custom timing, triggering, and processing with a wide array of modular I/O to meet any embedded application requirement.

Configure Your Complete NI CompactRIO System

NI CompactDAQ Platform

Ordering Information | Detailed Specifications | Pinouts/Front Panel Connections
For user manuals and dimensional drawings, visit the product page resources tab on ni.com.

Last Revised: 2014-11-06 07:14:25.0

NI 9474
24 V, Sourcing Digital Output, 8 Ch Module

- 8 channels, 1 µs high-speed digital output
- 5 V to 30 V, sourcing digital output
- Compatibility with NI CompactDAQ counters
- 250 Vrms, CAT II isolation
- 10-position screw-terminal connector
- -40 °C to 70 °C operating, 5 g vibration, 50 g shock

Overview

The NI 9474 is a C Series 8-channel, 1 µs high-speed sourcing digital output module. It works in any NI CompactDAQ or CompactRIO chassis. Each channel is compatible with 5 V to 30 V signals and features transient overvoltage protection of 2,300 Vrms between the output channels and earth ground. Each channel also has an LED that indicates the state of that channel. With the NI 9474, you can connect directly to a variety of industrial devices such as motors, actuators, and relays.

The NI 9474 module is a correlated digital module, which means it can perform correlated operations, triggering, and synchronization when installed in an NI CompactDAQ chassis.

Recommended Accessories
-NI 9927 strain relief and operator protection

Optional Accessories
-NI 9936 extra screw-terminal block (quantity 10)
-NI 9980 extra spring-terminal block (quantity 10)
Note: The NI 9980 is not compatible with the NI 9927 and must be used with low or nonhazardous voltages or installed in a properly rated enclosure.

Box Contents
-1 NI 9474 C Series module
-1 NI 9474 Operating Instructions and Specifications manual
-1 NI 9936 10-position screw-terminal connector

Back to Top

Comparison Tables

Product Name	Signal Levels	Direction	Channels	Update Rate	Continuous Current	Connectivity
NI 9375	12, 24 V	Sinking Input, Sourcing Output	16 In, 16 Out	7 µs In, 500 µs Out	100 mA/ch	Spring Terminal, 37-Pin D-SUB
NI 9472	12, 24 V	Sourcing Output	8	100 µs	750 mA/ch	Screw Terminal, 25-Pin D-SUB
NI 9474	5, 12, 24 V	Sourcing Output	8	1 µs	1 A/ch	Screw Terminal
NI 9475	5, 12, 24, 48, 60 V	Sourcing Output	8	1 µs	1 A/ch	25-Pin D-SUB
NI 9476	12, 24 V	Sourcing Output	32	500 µs	250 mA/ch	37-Pin D-SUB
NI 9477	5, 12, 24, 48, 60 V	Sinking Output	32	8 µs	1 A/ch (20 A per Module)	37-Pin D-SUB
NI 9478	5, 12, 24, 48 V	Sinking Output	16	7 µs	1.2 A/ch	37-Pin D-SUB

Back to Top

Last Revised: 2014-11-06 07:14:19.0

Real-Time Controller with 800 MHz, 512 MB DRAM, 4 GB Storage

NI cRIO-9024

- Embedded controller runs LabVIEW Real-Time for deterministic control, data logging, and analysis
- 800 MHz processor, 4 GB nonvolatile storage, 512 MB DDR2 memory
- Dual Ethernet ports with embedded Web and file servers for remote user interfacing
- Hi-Speed USB host port for connection to USB flash and memory devices
- RS232 serial port for connection to peripherals; dual 9 to 35 VDC supply inputs
- -20 to 55 °C operating temperature range

Overview

The NI cRIO-9024 embedded real-time controller is part of the high-performance NI CompactRIO programmable automation controller (PAC) platform. It features an industrial 800 MHz real-time Freescale processor for deterministic, reliable real-time applications and contains 512 MB of DDR2 RAM and 4 GB of nonvolatile storage for holding programs and logging data.

Back to Top

Requirements and Compatibility

OS Information	Driver Information	Software Compatibility
▪ VxWorks	▪ NI-RIO	▪ LabVIEW
		▪ LabVIEW FPGA Module
		▪ LabVIEW Professional Development System
		▪ LabVIEW Real-Time Module

Back to Top

Application and Technology

System Configuration

The NI cRIO-9024 rugged, reliable controller is designed for low-power consumption with dual 9 to 35 VDC supply inputs that deliver isolated power to the CompactRIO chassis/modules and a -20 to 55 °C operating temperature range. The cRIO-9024 accepts 9 to 35 VDC power supply inputs on power up and 6 to 35 VDC power supply inputs during operation, so it can function for long periods of time in remote applications using a battery or solar power.

The controller provides two Ethernet ports - 10/100 and 10/100/1000 - that you can use to conduct programmatic communication over the network and built-in Web (HTTP) and file (FTP) servers. The ports also are compatible with the NI 9144 C Series expansion chassis, so you can connect more deterministic I/O for your application.

To create additional storage capability for your embedded logging applications, the cRIO-9024 has a Hi-Speed USB host port to which you can connect external USB-based storage media (flash drives and hard drives). In addition, the controller features a fault-tolerant file system that provides increased reliability for data-logging applications.

The CompactRIO real-time controller connects to any four- or eight-slot CompactRIO reconfigurable chassis. The user-defined FPGA circuitry in the chassis controls each I/O module and passes data to the controller through a local PCI bus using built-in communication functions.

Embedded Software

The cRIO-9024 runs NI LabVIEW Real-Time Module software on the Wind River VxWorks real-time operating system (RTOS) for extreme reliability and determinism. You can now use leading VxWorks RTOS technology and LabVIEW graphical programming tools to quickly design, prototype, and deploy a customizable, commercial off-the-shelf embedded system.

You can synchronize embedded code execution to an FPGA-generated interrupt request (IRQ) or an internal millisecond real-time clock source. The LabVIEW Real-Time ETS OS provides reliability and simplifies the development of complete embedded applications that include time-critical control and acquisition loops in addition to lower-priority loops for postprocessing, data logging, and Ethernet/serial communication. Built-in elemental I/O functions such as the FPGA Read/Write function provide a communication interface to the highly optimized reconfigurable FPGA circuitry. Data values are read from the FPGA in integer format and then converted to scaled engineering units in the controller.

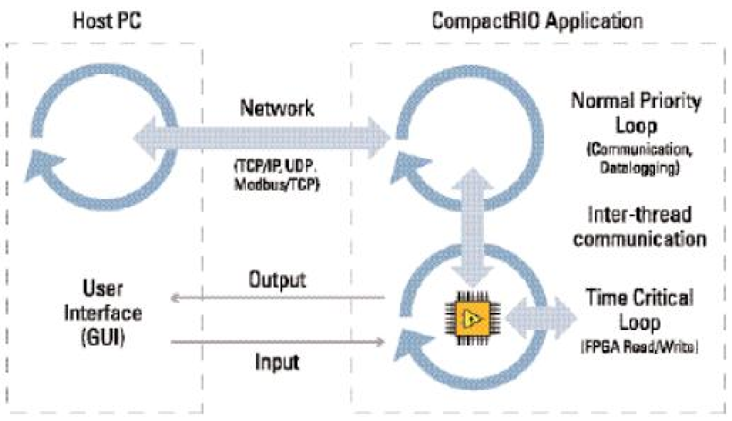


Figure 1. CompactRIO Software Architecture

Built-In Servers

In addition to programmatic communication via TCP/IP, UDP, Modbus/TCP, IrDA, and serial protocols, the CompactRIO controllers include built-in servers for Virtual Instrument Software Architecture (VISA), HTTP, and FTP. The VISA server provides remote download and communication access to the reconfigurable I/O (RIO) FPGA over Ethernet. The HTTP server provides a Web browser user interface to HTML pages, files, and the user interface of embedded LabVIEW applications through a Web browser plug-in. The FTP server provides access to logged data or configuration files.

Back to Top

Ordering Information

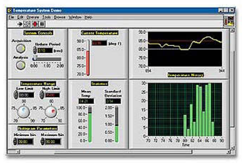
For a complete list of accessories, visit the product page on ni.com.

Products	Part Number	Recommended Accessories	Part Number
NI cRIO-9024			
cRIO-9024, Real-Time PowerPC Controller for cRIO, 800 MHz Requires:	781174-01	**Connector Block:** Not Applicable - NI 9978 4-pos screw terminal power supply plugs (quantity 5)	196938-01
		Connector Block: Not Applicable - NI 9979 Strain relief kit for 4-pos power connector	196939-01

Back to Top

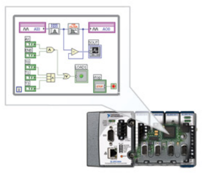
Software Recommendations

LabVIEW Professional Development System for Windows



- Advanced software tools for large project development
- Automatic code generation using DAQ Assistant and Instrument I/O Assistant
- Tight integration with a wide range of hardware
- Advanced measurement analysis and digital signal processing
- Open connectivity with DLLs, ActiveX, and .NET objects
- Capability to build DLLs, executables, and MSI installers

NI LabVIEW FPGA Module



- Design FPGA applications for NI reconfigurable I/O (RIO) hardware targets
- Program with the same graphical environment used for desktop and real-time applications
- Execute control algorithms with loop rates up to 300 MHz
- Implement custom timing and triggering logic, digital protocols, and DSP algorithms
- Incorporate existing HDL code and third-party IP including Xilinx CORE Generator functions
- Included in the LabVIEW Embedded Control and Monitoring Suite

References

[1] I. Roytelman and V. Ganesan, “Modeling of local controllers in distribution network applications,” *IEEE Trans. Power Del.*, vol. 15, no. 4, pp. 1232–1237, Oct. 2000.

[2] I. Dzafic, R. Jabr, E. Halilovic, and B. Pal, “A sensitivity approach to model local voltage controllers in distribution networks,” *IEEE Trans. Power Syst.*, vol. 29, no. 3, pp. 1419–1428, May 2014.

[3] I. Roytelman and V. Ganesan, “Coordinated local and centralized control in distribution management systems,” *IEEE Trans. Power Del.*, vol. 15, no. 2, pp. 718–724, Apr. 2000.

[4] B. Robbins, C. Hadjicostis, and A. Dominguez-Garcia, “A two-stage distributed architecture for voltage control in power distribution systems,” *IEEE Trans. Power Syst.*, vol. 28, no. 2, pp. 1470–1482, May 2013.

[5] H. Hatta, S. Uemura, and H. Kobayashi, “Demonstrative study of control system for distribution system with distributed generation,” in *Power Syst. Conf. and Expo.*, Seattle, WA, Mar. 2009.

[6] J. Mahseredjian, V. Dinavahi, and J. Martinez, “Simulation tools for electromagnetic transients in power systems: overview and challenges,” *IEEE Trans. Power Del.*, vol. 24, no. 3, pp. 1657–1669, Jul. 2009.

[7] V. Dinavahi, M. Iravani, and R. Bonert, “Real-time digital simulation of power electronic apparatus interfaced with digital controllers,” *IEEE Trans. Power Del.*, vol. 16, no. 4, pp. 775–781, Oct. 2001.

[8] A. Barry, F. Guay, S. Guerette, and P. Giroux, “Digital real-time simulation for distribution systems,” in *Transmission and Distribution Construction, Operation and Live-Line Maintenance Proc.*, Montreal, QC, 2000.

[9] X. Wu, H. Figueroa, and A. Monti, “Testing of digital controllers using real-time hardware in the loop simulation,” in *Power Electron. Specialists Conf.*, Aachen, Germany, Jun. 2004.

[10] M. Davarpanah, M. Sanaye-Pasand, and R. Iravani, “A saturation suppression approach for the current transformer—part II: performance evaluation,” *IEEE Trans. Power Del.*, vol. 28, no. 3, pp. 1936–1943, Jul. 2013.

[11] E. Lelarasmee, A. E. Ruehli, and A. Sangiovanni-Vincentelli, “The waveform relaxation method for time-domain analysis of large scale integrated circuits,” *IEEE Trans. Comput-Aided Design of Integrated Circuits and Syst.*, vol. 1, no. 3, pp. 131–145, Jul. 1982.