AC 2008-838: VIRTUAL INSTRUMENTATION INTERFACES FOR REAL-TIME CONTROL AND DISPLAY OF ELECTRIC MACHINE DRIVES

Ramakrishnan Sundaram, Gannon University

RAM SUNDARAM received his B.S. degree in Electrical Engineering from I.I.T., New Delhi, India, the M.S. degree and the E.E. degree from M.I.T., Cambridge, MA in 1985 and 1987, and Ph.D. in Electrical Engineering from Purdue University in 1994. He is currently a faculty member in the Electrical and Computer Engineering Department at Gannon University.

Fong Mak, Gannon University

FONG MAK, P.E. received his B.S.E.E. degree from West Virginia University in 1983, M.S.E.E. and Ph.D. in Electrical Engineering from the University of Illinois in 1986 and 1990. He is currently the Chair of Electrical and Computer Engineering at Gannon University. He is also the Program Director for the professional-track Gannon/GE Transportation Embedded System Graduate Program.

Sunil Tandle, Gannon University

SUNIL TANDLE is a graduate student in the Electrical and Computer Engineering Department at Gannon University. He is working toward his M.S. degree in Electrical and Computer Engineering.

Virtual Instrumentation Interfaces For Real-Time Control And Display Of Electric Machine Drives

Introduction

This paper presents a real-time instrumentation setup that benefits the subject matter in two courses: *electric drives* and *test and measurement*. The paper discusses virtual instrumentation-based interfaces for real-time control and display of electric machine drives. This approach will lead to the flexibility of applying this setup as a platform to study electric drives as well as the *LabVIEW*-based (from *National Instruments, Inc.*) experimentation design.

In this lab, the machine drive is implemented using a distributed real-time simulation system from *Opal-RT* [1] with machines and power electronics drive board as hardware-in-the-loop (HIL). The simulation executes on a hardware configuration consisting of the Command Station (host PC) communicating with the target node (another PC) via Ethernet communication links, I/O boards interfaced to HIL. The Command Station serves as the user interface to edit and modify models, to view model data, to execute the model, to compile the model into C code and load the code onto each target node. The target nodes perform real-time execution of the model simulation and include a real-time communication interface between the nodes and I/O modules. The models are designed and implemented in *Simulink* (from *The Mathworks Inc*.). For instance, the experiment to perform an open-loop control of a dc motor consists of the following setup: a dc motor under test coupled with a DC motor as load. For the experiment, the control to the dc motor under test is fed to a chopper. The steps may involve (a) creating a *Simulink* model for a pulse-width modulation control to run the dc motor at the desired speed, and (b) building the *LabVIEW* interface with controls for variables such as the reference speed of the motor (rpm), the reference frequency (Hz), numerical indicators to display the frequency of the induction motor, the speed of the DC motor, and graphs for phase voltage and phase current waveforms.

The experimental setup is modeled after the one proposed by the University of Minnesota [2, 3]. However, the setup along with the modification to the model with the dc motor assembly in the loop (HIL) is first correctly executed under the *RT-LAB* real-time system solution. This paper will give a brief description of a real-time controlled machine drive experiment, but the main focus is on the *LabVIEW* interface and measurement design to enhance the flexibility and capability to provide measurement analysis and control with visual interfaces. Not only does this *LabVIEW* application enhance the functionality of this experiment, but it also provides a platform of implementation and testing for the students studying *LabVIEW* design as part of the *Test and Measurement* class.

The paper consists of six sections. Section 2 provides a brief description of a real-time controlled dc machine drive experiment as an illustration. Section 3 presents the *LabVIEW*-based virtual instrumentation (*VI*) interface to this experiment. Section 4 summarizes the assessment of learning outcomes based on an on-line survey prepared for the students in the *Advanced Instrumentation and Measurement* class offered during the fall semester of 2007. Section 5 outlines the conclusions and future considerations. Section 6 lists the references.

Section 2: HIL Experiment Setup

Figure 1 displays the configuration of a typical dc drive system as an illustration. The machine drive[1] shown in Figure 1 is implemented originally using a distributed real-time simulation system from *Opal-RT* with electric machines and power electronics drive board as hardware-in-the-loop (*HIL*). The control algorithm for the dc drive system and its associated monitoring graphic user interface (software model) are modeled in *Simulink* and resides in the *Opal-RT* simulation system. The software model simulation executing on the Command Station (host PC) communicates with the target node (another PC) via Ethernet communication links, and I/O boards interfaced to *HIL*. The Command Station serves also as the user interface to edit and modify models, to view model data, to execute the model, to compile the model into C code and load the code onto the target node. The target node performs real-time execution of the model simulation and includes a real-time communication interface between the node and I/O modules.

The software model for the dc drive system under study consists of an open-loop control algorithm and the user interface for controlling inputs and monitoring selected signals as the outputs. The *Opal-RT* simulation system requires two subsystem blocks. The main computational elements of the model are always contained in the master subsystem that starts with "SM". The console subsystem is the subsystem operating on the command station that enables the user to interact with the system. It contains all the *Simulink* blocks related to acquiring and displaying data. In *RT-LAB, OpComm* blocks are used to enable and save communication setup information. All inputs to top-level subsystems must first go through an *OpComm* block before they can be used. The two subsystems are shown in Figure 2.

The details of the master subsystem, SM_Motor_Control, are shown in Figure 3. The master subsystem consists of the following blocks:

(a) open-loop speed control model labeled as *DC_Motor*
(b) encoder model, namely *Encoder*
(c) an *Analog In* model

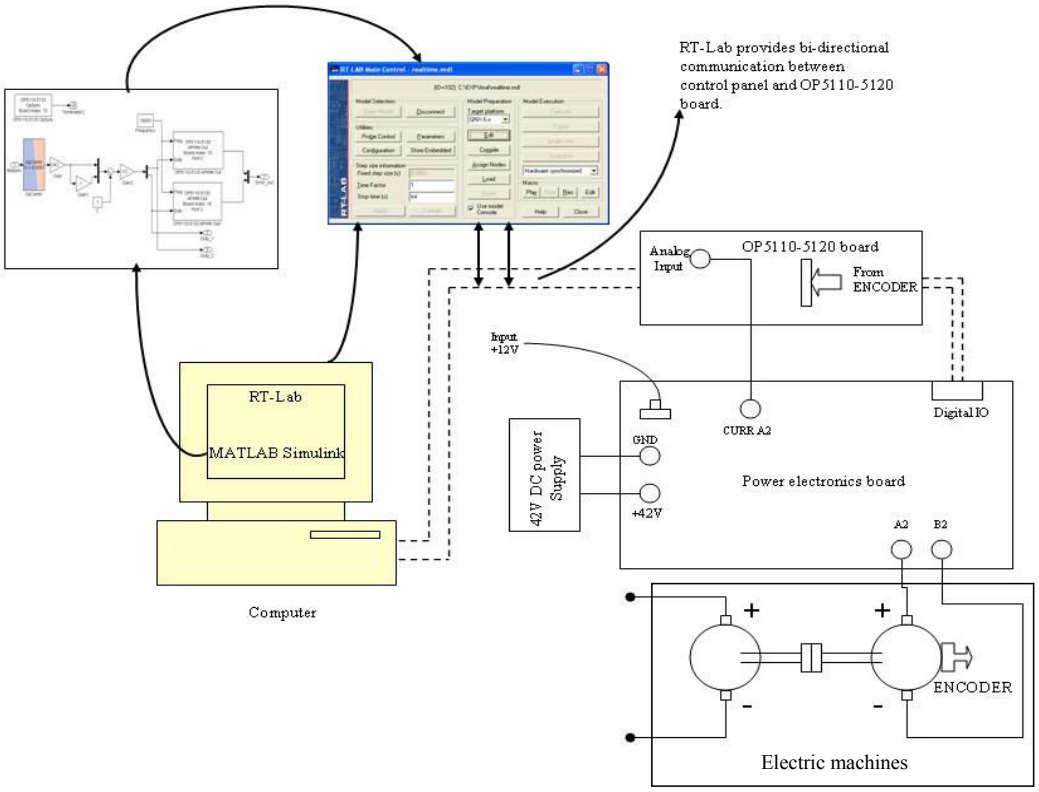

Figure 1: Real-time controlled dc drive system configuration

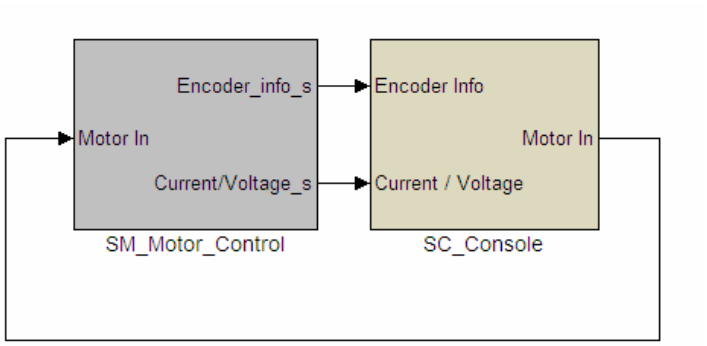

Figure 2: Subsystems of the motor model

The speed-control model receives the input signal, *Motor_in*, from the Console subsystem, *SC_Console*. This signal is a slider gain (shown in Figure 4) and is used to alter the duty cycle of the PWM pulses that are applied to the Power Electronics Drive Board via the multi-function OP5110-5120 digital I/O board for speed control of the DC motor. This board has the capability to generate two independent PWM voltage sources (A1B1C1 and A2B2C2) from a constant DC voltage source. Hence two machines can be independently controlled for independent control variables, at the same time. However, for this experiment, only phase A1 and B1 of one PWM voltage source is used as in a H-bridge format for the dc motor control.

The encoder model, on the other hand, provides information on the speed, frequency, position, and direction of rotation of the motor. The encoder makes use of the event detector feature in the OP5110-5120 and the Real-Time Events (RTE) library block from *Opal-RT* to generate signals into states and times before converting into digital waveform. The motor current, dc-voltage etc. from the Power Electronics Drive Board are fed through analog-to-digital module before going into the OP5110-5120.

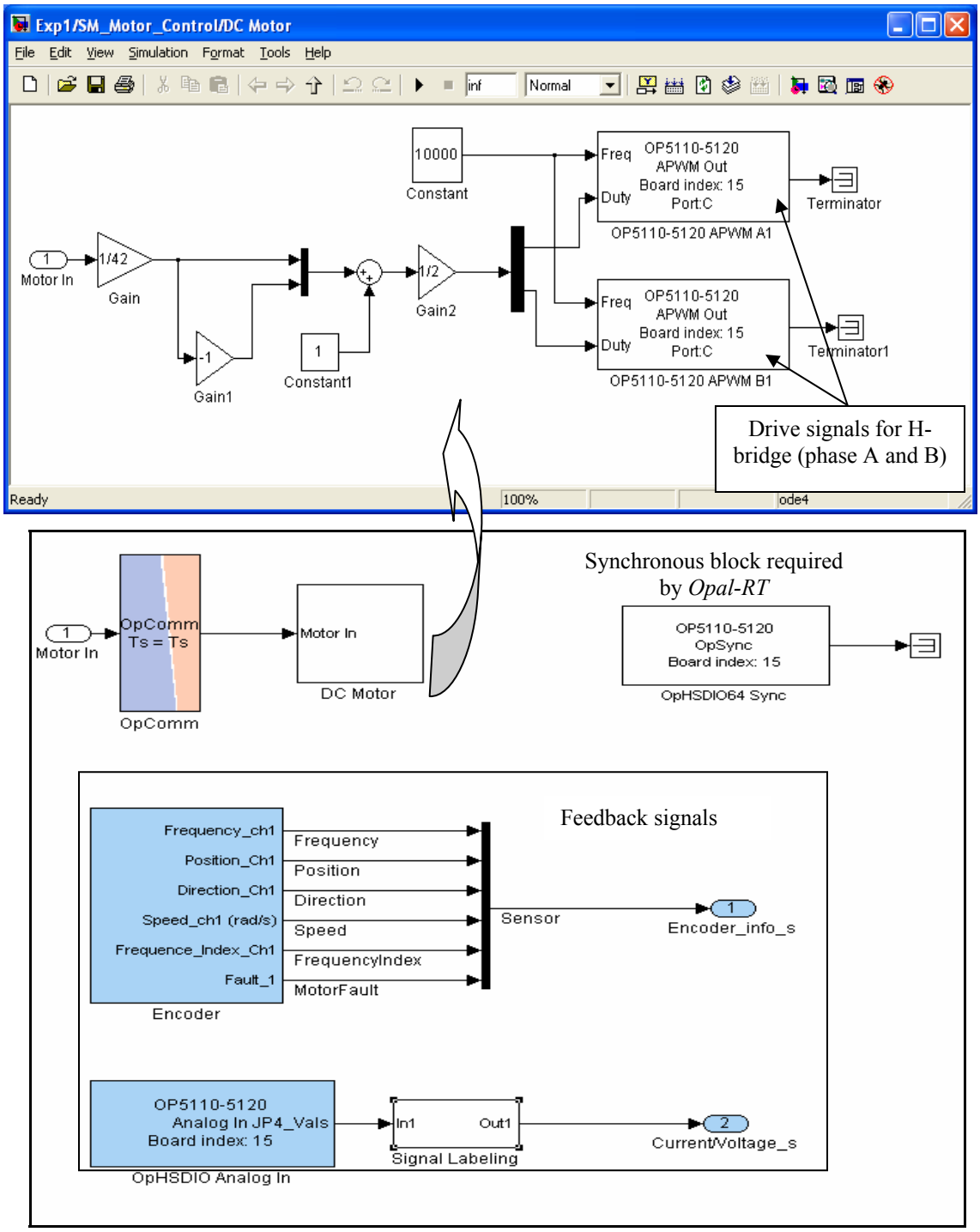

Figure 3: Master subsystem

The Console subsystem, shown in Figure 4, is used as the *human-machine interface* (*HMI*). A slider is used as the desired set point for voltage, the output of which is fed back as input to the master subsystem. The signals to be monitored, such as speed, current, and voltage are connected to the display scope as in indicated in Figure 4. Figure 5 shows a typical display on the console indicators.

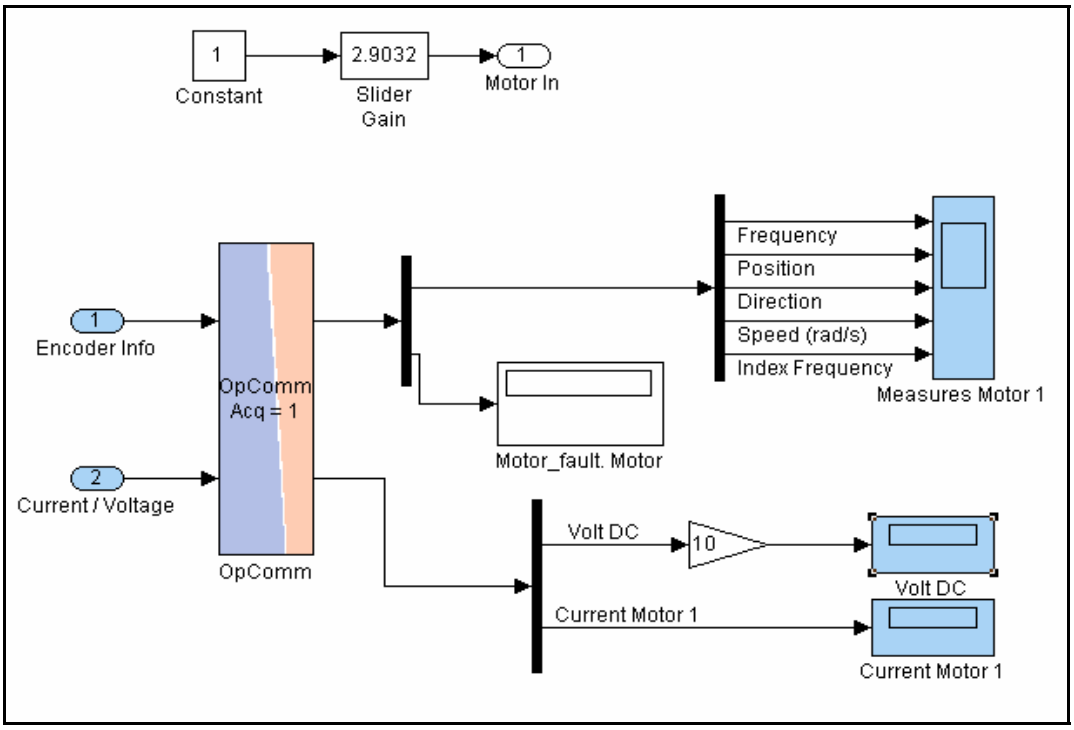

Figure 4: Console subsystem

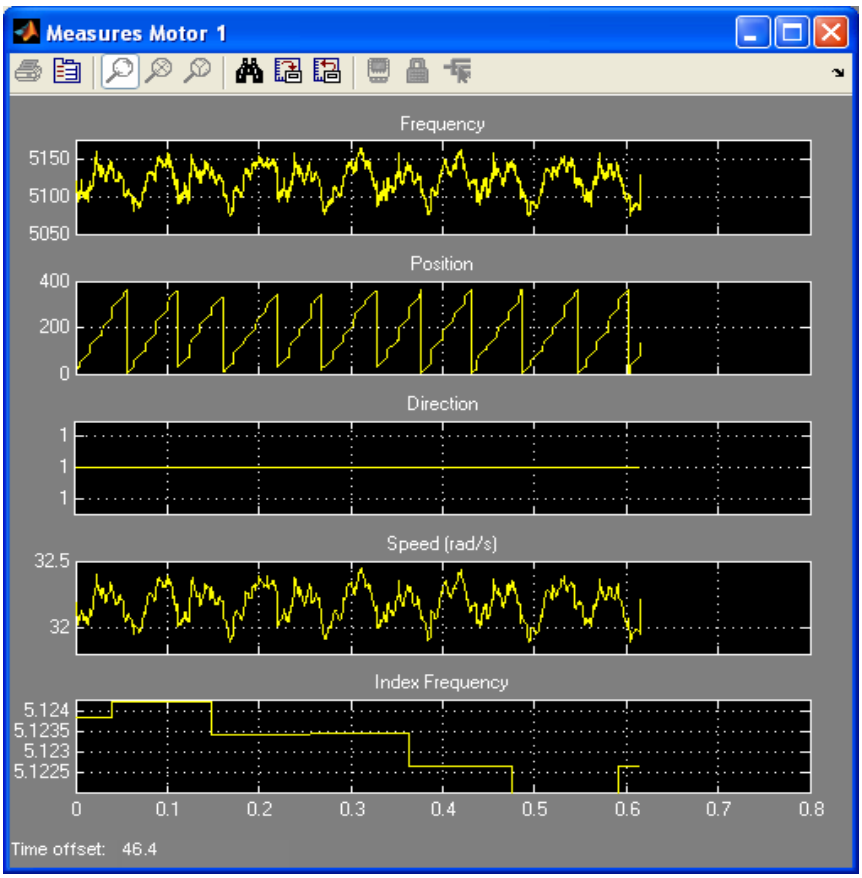

Figure 5: Console display

Section 3: The *LabVIEW* Interface

The *LabVIEW* interface for this experiment is displayed in Figure 6. This is a *real-time* interface with a single panel comprising (a) controls for motor input variables such as the reference speed and direction of the motor, (b) numerical and graphical indicators to display the speed, position, and frequency of the DC motor, and (c) graphs for the current waveform. Clearly, the advantages of the *LabVIEW*-based *HMI* are as follows:

(a) organized record of control inputs,
(b) systematic tracking of motor responses,
(c) clear presentation of the evidence of the experiment, and
(d) offers tools for advanced measurement analysis (e.g. Fourier spectra, THD)

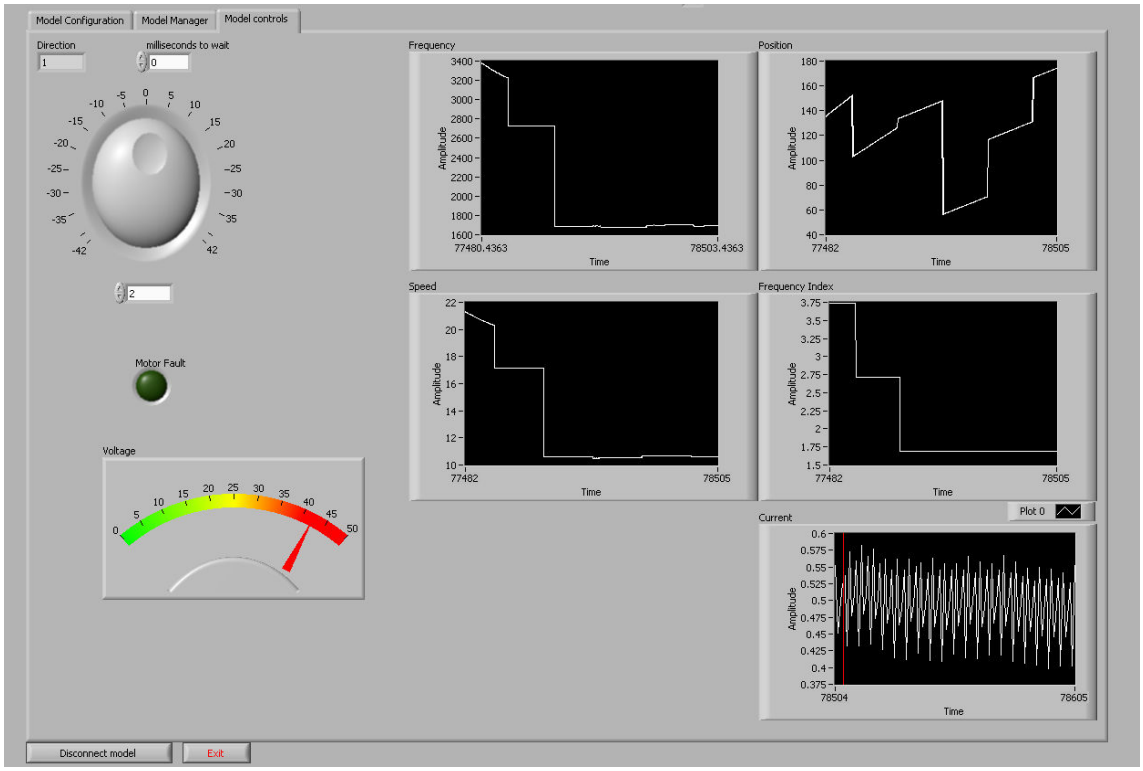

Figure 6: *LabVIEW*-based *RT* interface

Typical *LabVIEW* experiment setup:

The laboratory experiment titled 'Design *LabVIEW APIs* for *RT* system models with *HIL*' was performed by students enrolled in the *Advanced Instrumentation & Measurement* course. The students were assigned to work in teams comprising 2 to 3 members. The students were given a working *Simulink* model of the DC motor drive. This model consisted of the Master and Console subsystems as shown in Figure 3 and Figure 4, respectively. First, the students ensured that the model executed under *RT-Lab* control. To do so they had to compile the model, assign the model to a target, load the model, and execute the model. If this was successful, they reset the model in *RT-Lab*.

In this experiment, the students had to build the *LabVIEW GUI* [4, 5] shown in Figure 6 and execute the DC motor model using their *LabVIEW GUI*. A template of the *LabVIEW GUI* was provided to the students.

The front panel of this *LabVIEW GUI* consisted of the following three tabs.

(a) Model Configuration
(b) Model Manager
(c) Model Controls

The steps to build and execute the model using this *GUI* were described as follows:

Model configuration:
To load and connect to the model, first specify the path to the model. The model is executed under the mode "Hardware synchronized". The target platform operates in Windows NT and the embedded node is QNX195. The model is executed on startup and reset before it is quit.
Model Manager:
In *LabVIEW*, the three choices to manage the model are (1) Execute (2) Pause (3) Restart The model state is "Running" when the model is executing.
Model Controls:
This portion of the front panel contains the user-defined control signals and model responses or indicators. The control signals such as the desired direction of rotation are sent to the model while the model responses such as frequency and position are received from the model.

To send signals to the model, first find the sequence titled *Send data to target* in the *Acquisition section* of the block diagram of the *LabVIEW GUI* . Provide the signals to a *Build Array* node. Then, the wire coming out from this node is sent to the *SM_Master* subsystem through a *subVI* called *OpalSetSignal2.vi*. If one must send more signals to the model, increase the number of inputs in the *Build Array* node. The signals connected to the *Build Array* must be in the same order as they are connected to the *OpComm* block. Similarly, if a vector is connected to the *OpComm* block, the order must be the same as they are in the vector.

To receive signals from the model, find the sequence titled *Receive data from target* in the *Acquisition section* of the block diagram. Then use the *subVI* called *OpalGetAcqGroupSignals2.vi*. Provide the number of signals that must be received in that group and the acquisition group one wishes to receive the data from.
To send the desired signal to the appropriate indicator, use an *Index Array* and specify the index which corresponds to the position of the signal in the *OpComm* of the *SC_Console* subsystem. Then wire the output of the *Index Array* to the terminal of the indicator chosen. Note that both *OpalSetSignal2.vi* and *OpalGetAcqGroupSignals2.vi* are subVIs provided by *Opal-RT* as part of their API capability to allow interface to other custom built interface.

The students in each team had to properly identify and wire the signals sent to the model (control inputs) as well as signals received from the model (motor responses) as described above. Thereafter, the DC Motor model was executed using their *LabVIEW GUI*. The students had to observe and record the motor variables for different settings of the real-time control inputs.

The *LabVIEW* interface improves the laboratory experience for students in both courses. Students in the electric drives course can monitor and analyze the signals with more precision while those in the Instrumentation & Measurement course can build and customize the *HMI* for specific applications. In addition, the *LabVIEW*-based interface promotes cross-course interaction without either group required to have the knowledge of the other group. For instance, the detail of the DC motor experiment (motor equations, field analysis) is not required in the design of the interface by the students of the instrumentation course. Likewise, the execution of the DC motor experiment with a well designed custom *HMI* does not require knowledge of advanced *LabVIEW* concepts. The next section presents the outcomes of the on-line survey completed by the students in the *Advanced Instrumentation & Measurement* course.

The details of the experimental setup for both the electric drives[6] and the *LabVIEW* interface[7]are provided at the web site.

Section 4: On-line Survey and Learning Outcomes Assessment

Each student completed an on-line survey at the end of the semester. The on-line survey consisted of two sections (a) quantitative (b) qualitative. The quantitative section comprised graded responses (on a scale from 0 to 5) to questions in the following broad categories.

(a) Effectiveness of integration
(b) Participation on team

The qualitative section asked each student to comment on the experiment, and propose approaches to streamline and improve the presentation of the experiment. There were 48 students participated in the survey.

(a) Effectiveness of integration

Students were asked to respond to questions in the following question.

Did the integration of the electric drives experiment with LabVIEW-based virtual instrumentation (VI) aid in the following categories?

Table 1 summarizes the graded response to the questions. The students expressed strong agreement (greater than 75% in all but one case) in all questions. Since this was only one experiment during the term with the RT component, the students were not inclined to strongly agree that the experiment helped them design an *HMI* with superior features. At present, the content of the course is being reorganized to incorporate additional

experiments requiring the design of advanced real-time instrumentation interfaces for drive system models with HIL.

Table 1: Effectiveness of integration

	Questions	Strong Agree	Agree	Neutral	Disagree	Strongly Disagree	N.A.	Mean (5)	Important (5)
1	Learning of basic and advanced LabVIEW concepts	87.5	12.5	0.0	0.0	0.0	0.0	**4.9**	**5.0**
2	Application of these concepts to real-time (RT) concepts	79.2	18.8	2.1	0.0	0.0	0.0	**4.8**	**5.0**
3	Understanding of data acquisition principles	77.1	22.9	0.0	0.0	0.0	0.0	**4.8**	**5.0**
4	Implementation of data measurement techniques	83.3	14.6	2.1	0.0	0.0	0.0	**4.8**	**5.0**
5	Design of Human-Machine Interfaces (HMI) with superior features	62.5	27.1	2.1	2.1	0.0	6.3	**4.6**	**5.0**
6	Demonstrate the capability of LabVIEW as a real-time interface as opposed to Opal-RT	75.0	20.8	4.2	0.0	0.0	0.0	**4.7**	**5.0**
7	Improve your understanding of how LabVIEW can be applied to hardware-in-the-loop (HIL) real-time systems	75.0	20.8	4.2	0.0	0.0	0.0	**4.7**	**5.0**
	Total Class Response:	**77.1**	**19.6**	**2.1**	**0.3**	**0.0**	**0.9**	**4.7**	**5.0**

(b) Participation on team

The next quantitative section asked the students to respond to the following:
Rate your contribution as an individual to the team-based execution of the experiment in the following categories.

Table 2: Participation on team

	Questions	Major Contribution	Above Average	Average	Less than average	No	N.A.	Mean (5)	Important (6)
1	VI design components	77.1	20.8	2.1	0.0	0.0	0.0	**4.8**	**6.0**
2	Data measurement and analysis	77.1	20.8	2.1	0.0	0.0	0.0	**4.8**	**6.0**
3	Documentation of the outcomes	70.8	22.9	6.3	0.0	0.0	0.0	**4.6**	**6.0**
	Total Class Response:	**75.0**	**21.5**	**3.5**	**0.0**	**0.0**	**0.0**	**4.7**	**6.0**

Table 2 shows that at least 70% of the students felt that they made major contributions to VI design, data measurement and analysis, and the documentation of the outcomes. In the

three categories, almost all the students believed they made significantly more than an average contribution to the team.

(c) Qualitative section

This section asked the students to make observations on each of three questions:

(1) *Are there any components of the integration which must be altered?*
Most students did not see any reason to alter the experiment. However, some students felt that simpler examples to practice with *RT lab* interface before exploring the *LabVIEW* interface would be helpful. Others expressed the need for additional experiments with *RT-lab* and *RT-LabVIEW* interfaces. One student felt that the course was *the best course in his entire Masters' degree program with full practical training into LabVIEW*.

(2) *Are there any components of the integration which must not be altered?*
Students did not identify any major components that must be altered. They felt that the integration of *LabVIEW* with real-time control and display must not be altered. According to some of them, the approach is in a good flow which trains the students in the field of real-time.

(3) *Propose ways to improve the presentation.*
Students expressed the need for more practice and experimentation with the tool. They felt that they had to rush with the experiment because of lack of lab use. They did not have much time to spend in the lab. One student would have liked to have a *few more labs based on this RT lab because he thought students can learn a lot from them*. Another student felt that *the presentation must be very interactive so that we can share the information about that model*. Another opinion expressed was *the need for some theory classes as well*.

Section 5: Conclusions and Future Considerations

The integration of the *LabVIEW*-based *virtual instrumentation* with real-time control and display of electric machine drives was successful. The introduction of real-time system control and data acquisition in the *Advanced Instrumentation & Measurement* course is the logical first step to train the students to become effective design engineers in the workforce. However, the following issues must be addressed for continuous growth and improvement.

(a) include some theory sessions to reinforce *RT* concepts and support the lab activity
(b) develop additional *LabVIEW* exercises with *RT-HIL* control & analysis
(c) emphasize the use of *LabVIEW* tools for *RT-HMI* design
(d) incorporate in-class lab demonstrations and student presentations
(e) promote goal-oriented collaborative learning by encouraging the students in the instrumentation class to interact with the students in the electric drives class
(f) extend the coverage of the course or offer follow up courses to include advanced *RT* data analysis and interpretation

The powerful visual aids of the *LabVIEW*-based *HMI* design combined with the ease of the *RT* interface are suited for all subjects where the input/output (I/O) behavior of systems or sub-systems must be characterized.

Bibliography:

[1] "Distributed Real-Time Power System," *Opal-RT manuals*, 2007. http://www.opal-rt.com.

[2] Ned Mohan, "DSP Based Electric Drives Laboratory – User Manual," Department of Electrical and Computer Engineering, University of Minnesota, July 2007.

[3] Ned Mohan, "Electric Drives – An Integrative Approach," MNPERE, 2003.

[4] Nesimi Ertugrul, "*LabVIEW* for Electric Circuits, Machines, Drives, and Laboratories," Prentice Hall, 2002.

[5] Jeffrey Travis and Jim Kring, "*LabVIEW* for Everyone," Third Edition, Prentice Hall, 2007.

[6] Fong Mak, "*OpalRT* based Real-Time Controlled Electric Drives Laboratory – User Manual," Department of Electrical and Computer Engineering, Gannon University, December 2007. http://ece.gannon.edu/labs.

[7] Ram Sundaram, "*LabVIEW*-based *Virtual Instrumentation* with Real-Time Control Experiments," Department of Electrical and Computer Engineering, Gannon University, December 2007. http://ece.gannon.edu/labs.