Manual for the Control Teaching Laboratory

Daniel Zelazo
Dmitry Shneiderman
20.08.2015
Version 3

The purpose of this manual is to provide an overview of the Control Teaching Laboratory. The manual will review the laboratory environment covering both safety and operational procedures. Descriptions and tutorials for the hardware and software components are also provided here. This manual should be read before attempting to operate any equipment in the laboratory.

Figure 1: The control teaching laboratory pendulum.

Contents

1 The Control Teaching Laboratory

The Control Teaching Laboratory is located in the Philadelphia Flight Control Laboratory of the Department of Aerospace Engineering. The laboratory is an instructional facility to support undergraduate and graduate courses in the Department of Aerospace Engineering and the Department of Mechanical Engineering at the Technion.

Currently, the laboratory can host up to **8** experimental benches. Each bench consists of an electro-mechanical rotary pendulum and a computer that hosts supporting software (MATLAB/SIMULINK); see Figure 2. In this section, each component of the experimental bench and how they interact will be summarized.

Figure 2: The experimental bench setup.

1.1 The Rotary Pendulum

The centerpiece of each experimental bench is an electro-mechanically actuated rotary pendulum (referred to from here on as simply *the pendulum*). This physical system provides the framework for studying and modeling properties of dynamical systems and control systems. The pendulum can be operated in open-loop and closed-loop configurations making it a versatile platform for undergraduate and graduate level courses.

The pendulum consists of an arm that rotates in the horizontal plane and is driven by a DC motor. Attached to the end of the arm is a pendulum that is free to rotate in the vertical plane. The pendulum arm is fastened to the horizontal arm by a screw and is removable; see Figure 3.

The pendulum is actuated by a DC motor. The pendulum is also currently equipped with two sensors used to measure the angular position of the horizontal arm and the pendulum arm. This configuration, therefore, allows to study both Single-Input Multi-Output (SIMO) and Single-Input Single-Output (SISO) systems.

→ Section 2

Details on the motor and sensors are given in Section 2.

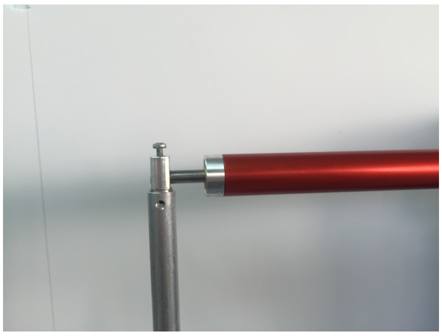

Figure 3: The pendulum arm is fastened with a screw.

1.2 The Lab Computer

Each experimental work bench is equipped with a desktop computer currently running the Windows 7 operating system. The computer plays the important role for the operation of the pendulum, and also peripheral roles required to complete laboratory assignments (internet access, data processing, simulation and computing capabilities, word processing, etc.).

For working with and operating the pendulum, the MATLAB and SIMULINK Desktop Real-Time software packages are used.

i) SIMULINK, developed by Mathworks and part of the MATLAB suite, is a graphical programming language tool used for modeling, simulating and analyzing dynamic systems. Using SIMULINK, it is possible to graphically build dynamic feedback controllers or open-loop control commands with block diagrams.

ii) SIMULINK Desktop Real-Time is a real-time kernel that enables physical devices to interface with SIMULINK models. This SIMULINK toolbox includes various I/O device drivers allowing real-time data acquisition and control from sensors and actuators.

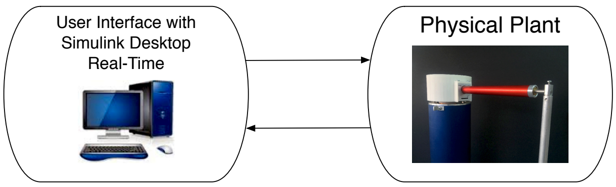

Figure 4: Real-time control of the pendulum is enabled by SIMULINK Desktop Real-Time.

2 Detailed Description of the Pendulum

Currently, the laboratory can hold up to 8 experimental benches. The pendulum is constructed using the components listed below.

1. **DC Motor** - The rotary pendulum is actuated by a Maxon A-max 32mm DC motor. The technical specifications of the motor will be available to students on the course website for the appropriate class, or by request to the lab instructor. Power is supplied to the motor by an external power supply physically enclosed in the pendulum

stand (see item 4). There is a transmission connecting the shaft of the motor to rotary arm with transmission ratio of 4.5:1.

2. **Pendulum Arm** - The pendulum can be connected to the horizontal arm with a screw. It is free to move in a circular motion in the vertical plane; see Figure 3

3. **Shaft Encoders** - A shaft encoder is an opto-electro-mechanical device that converts the angular position or motion of a shaft to an analog or digital signal. The pendulum setup uses two shaft encoders; one for the pendulum arm, and one for the motor shaft. The installed encoders measure the position at a resolution of 500 pulses per revolution.

4. **Pendulum Base** - The Pendulum Base integrates in its design a power supply with a power switch; see Figure 7. In addition

Figure 5: The base of the pendulum stand.

to supplying power to the DC motor, data link connection cables from the motor and sensors to the computer are also integrated in the base. SIMULINK outputs a digital signal between $-10V$ and $+10V$. A digital-to-analog converter converts this to a $\pm 10V$ analog signal. A power amplifier is used to supply power to the motor and has a gain of 1.2. This is summarized by the diagram in Figure 6.

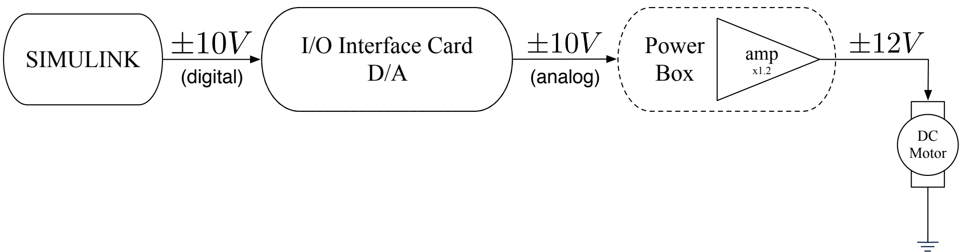

Figure 6: SIMULINK outputs a $\pm 10V$ digital signal. A D/A converter and the power amplifiers then deliveres $\pm 12V$ to the DC Motor.

5. **Main Power Switch** - Attached to each lab bench is a master ON/OFF switch used to supply power to the Pendulum stand. The

switch also serves as an "emergency abort" option to cut power to the pendulum in the event of undesirable or unstable behavior. During operation of the pendulum, one user should always be positioned near the switch and ready to cut power when needed. **! →** *The master switch must be in the OFF position when the bench is not in use.*

Figure 7: The main power switch.

3 The SIMULINK with Real-Time Desktop Environment

3.1 Overview

SIMULINK, developed by Mathworks and part of the MATLAB suite, is a graphical programming language tool used for modeling, simulating and analyzing dynamic systems. Within the control laboratory environment, SIMULINK serves two primary functions. The first is for the development of dynamic simulation models of the pendulum (with or without feedback), and the second is to provide real-time data acquisition and control of the pendulum using SIMULINK Desktop Real-Time.

When using the pendulum lab bench, two SIMULINK models (`*.mdl` files) must therefore be created. The first will contain a complete simulation model of the system being studied. This model is especially important when using the pendulum in a feedback configuration to validate the performance of the control design. The second model is based on the first simulation model and includes additional special SIMULINK blocks that are used to interface in real-time with the pendulum.

Figure 8 shows the SIMULINK Library Browser window containing all the available blocks. The block libraries that will primarily used in the laboratory are listed below (and highlighted in red in Figure 8):

- **Continuous** - blocks related to the modeling of continuous plants and the design of continuous controllers can be found here (i.e., integrator blocks, transfer-function blocks, etc.).
- **Math Operations** - blocks related to general mathematical operations can be found here (i.e., summation operators, trigonometric operators, etc.).
- **Sinks** - blocks for displaying and saving data generated during simulations can be found here (i.e., data scopes, `.mat` file generators).
- **Sources** - blocks for generating input signals can be found here (i.e., step signals, sinusoidal signals).

- **SIMULINK Desktop Real-Time** - special blocks that provide connections between physical I/O devices and real-time models.

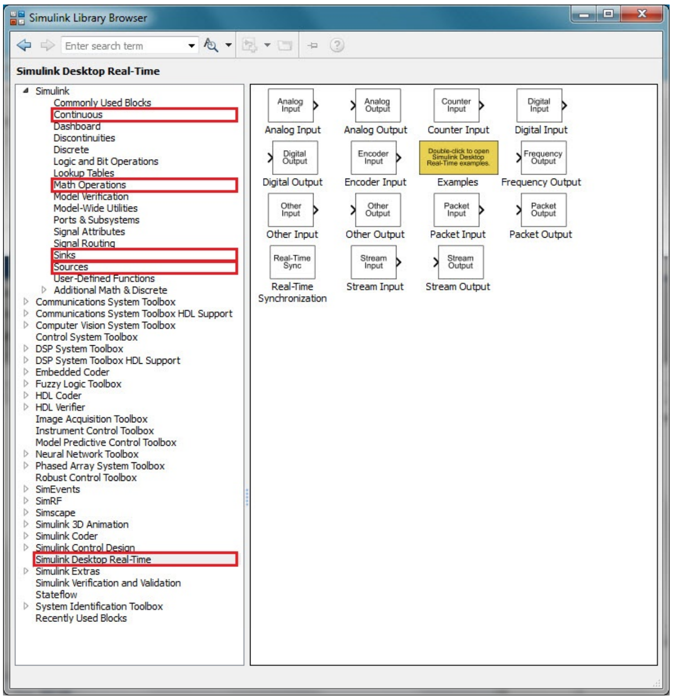

Figure 8: The SIMULINK Library Browser.

! → *When creating SIMULINK models, it is* **strongly advised** *to label all signals and blocks in a way that reflects their function.* Labeling of signals and blocks can be accomplished by a 'double-click' in the SIMULINK model on the desired location.

! → *Users of the pendulum must be comfortable working with the SIMULINK environment. This means understanding how to build basic simulation models and how to modify the various simulation parameters within SIMULINK. This manual focuses on how to utilize SIMULINK with SIMULINK Desktop Real-Time in order to work with physical systems and real-time simulation.*

3.2 SIMULINK Desktop Real-Time

In this subsection we describe how to create a SIMULINK model that can be for controlling the pendulum in real-time.

3.2.1 SIMULINK Model Configuration Settings

When creating a SIMULINK model for use with the SIMULINK Desktop Real-Time toolbox, the configuration parameters of the model must be set-up appropriately. This section reviews how the `*.slx` should be configured.

Create SIMULINK Model : Create a new SIMULINK model and save it with an appropriate file name. *The file should be saved in the appropriate folder directory. This directory depends on the course. Please consult the laboratory assistant for details.*

! →

Edit Configuration Parameters : Open the configuration parameters dialog window (Simulation -> Model Configuration Parameters).

1. Select the 'Solver' option in the left-hand window pane, and ensure the parameters are set as follows:

   - **Start time** : 0.0 **Stop time** : Inf
   - Solver options **Type**: Fixed-step
     Solver options **Solver**: ode4 (Runge-Kutta)
   - **Fixed-step size (fundamental sample time)**: 0.001

   A screen shot of the solver configuration is given in Figure 9.

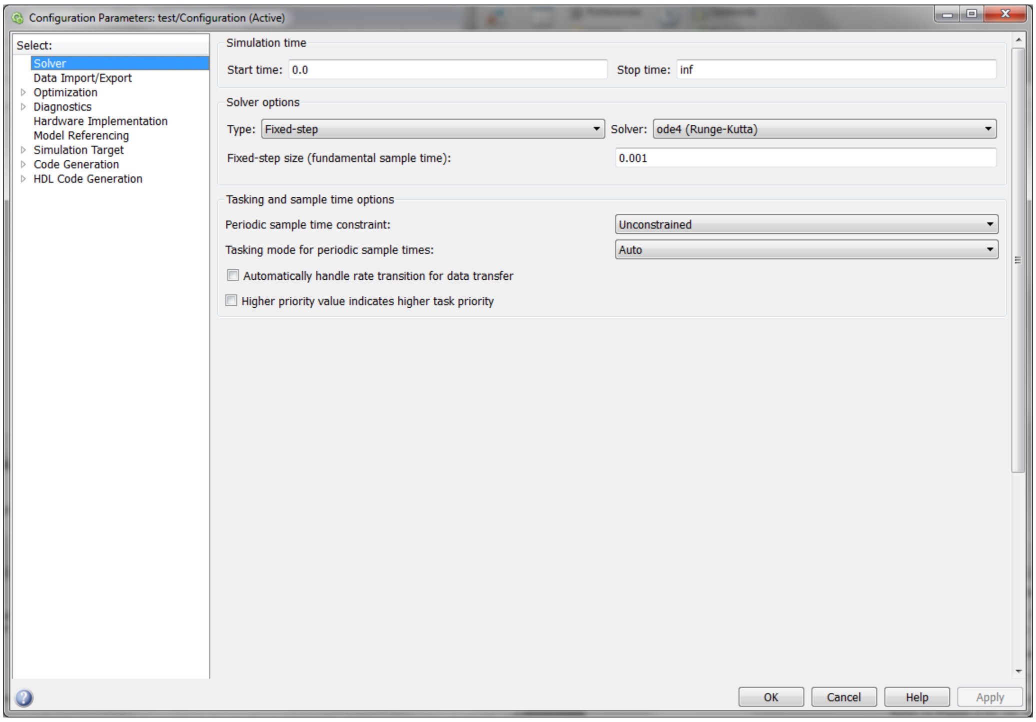

Figure 9: Configuration settings for Simulink solver.

2. Select the 'Optimization' option in the left-hand window pane. Refer to Figure 10 for the appropriate settings.

3. Click on the triangle symbol next to the 'Optimization' option in the left-hand window pane to open the 'Signals and Parameters' configuration window. Refer to Figure 11 for the appropriate settings.

4. Select the 'Code Generation' option in the left-hand window pane (this option used by Simulink Coder code generation software for creating C code and building a real-time application) and ensure the parameters are set as follows:

   - **System target file**: rtwin.tlc
     (this file can be located by clicking the 'Browse...' button next to the field)

   Refer to Figure 12 for the appropriate settings in this section.

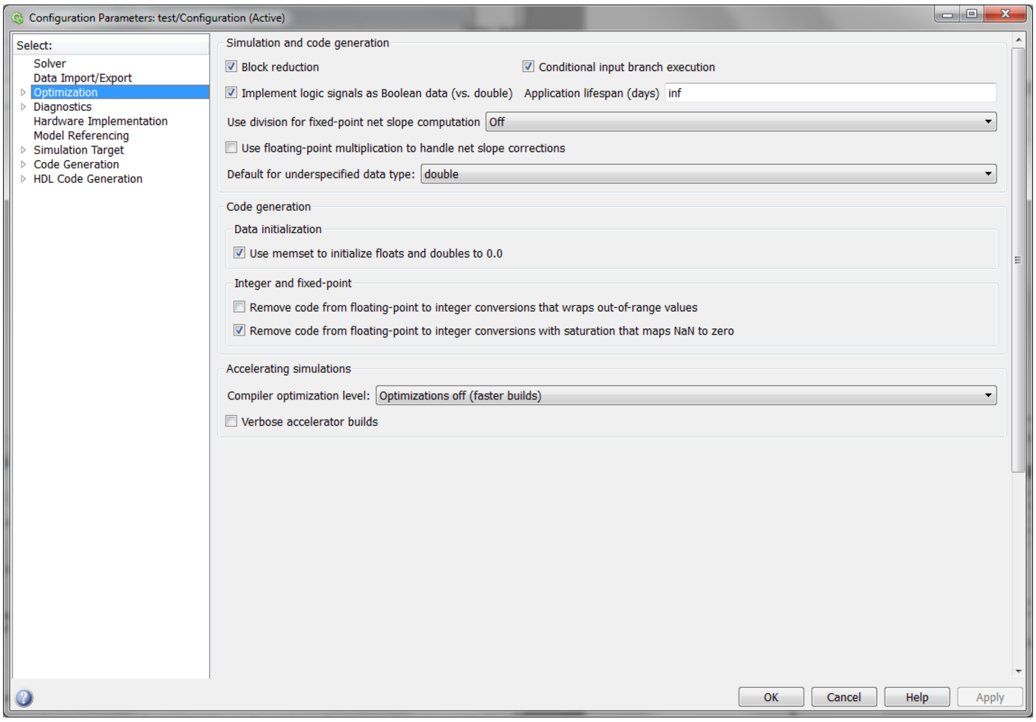

Figure 10: Configuration settings for Optimization.

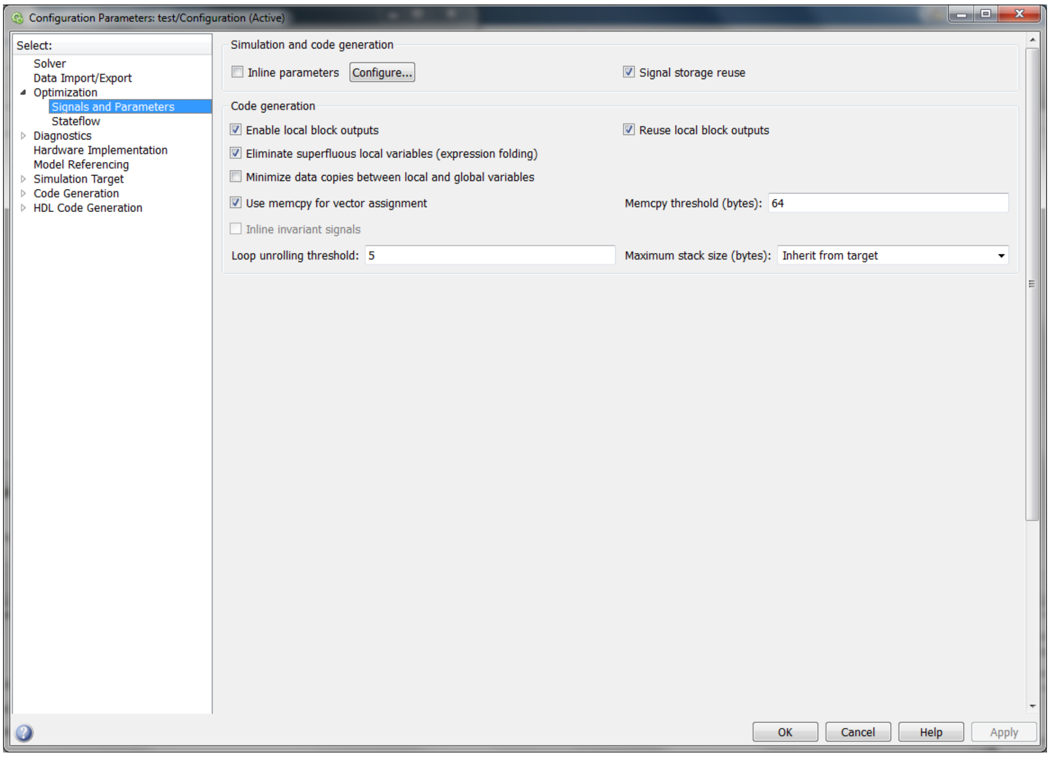

Figure 11: Configuration settings for Signals and Parameters.

5. Select the 'Hardware Implementation' option in the left-hand window pane (the default values are derived from the architecture of the host computer) and ensure the parameters are

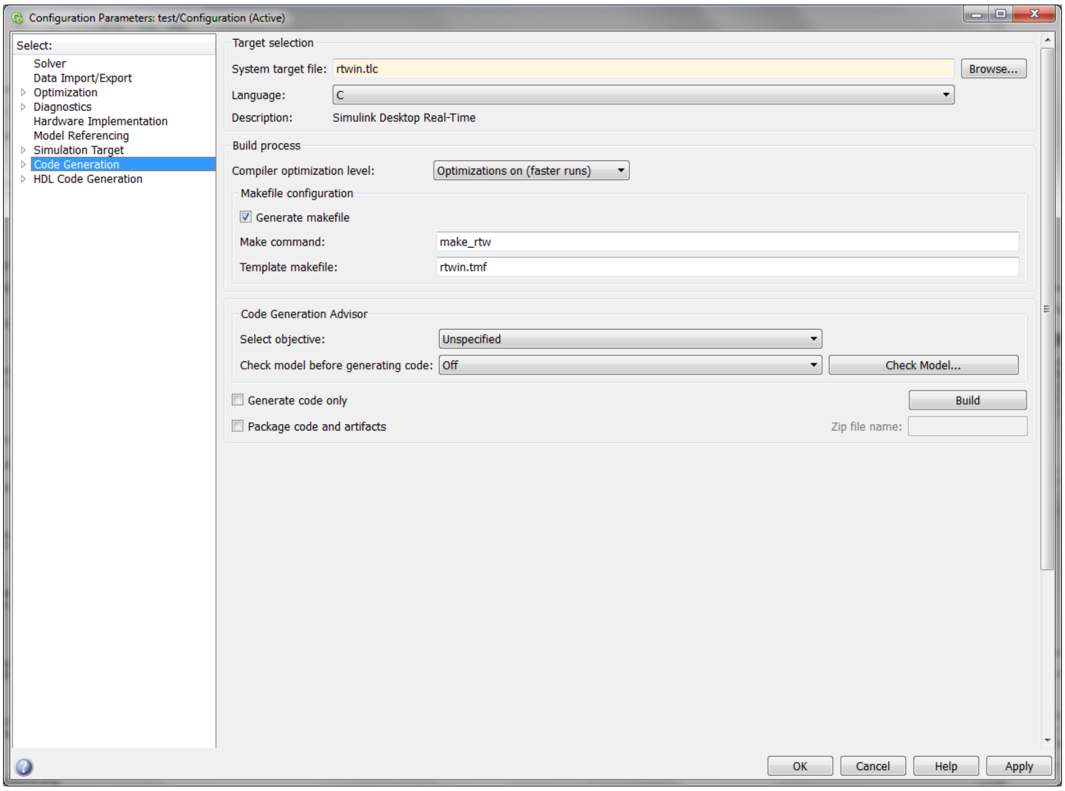

Figure 12: Configuration settings for Code Generation.

set as follows:

- **Device vendor**: Intel
- **Device type**: x86-64
- The option **Test hardware is the same as production hardware** in **Test hardware** is selected

Refer to Figure 13 for the appropriate settings in this section.

! → *Save the simulink model after all the configuration settings have been made.*

Add Encoders : In the Simulink Library Browser under the `Simulink Desktop Real-Time` library is a special block for using the encoders on the pendulum.

1. Open the Simulink Library Browser, and click on the `Simulink Desktop Real-Time` library. Under the `Simulink Desktop Real-Time` library, select the `Encoder Input` block; see Figure 14.
2. Add the `Encoder Input` block to the Simulink model. Double-click on the block to open the settings dialog and set the parameters as follows:
   - select from the list of registered boards (near 'Board setup' button) : International Instruments PCI-6221 [auto]
   - **Sample time** : 0.001

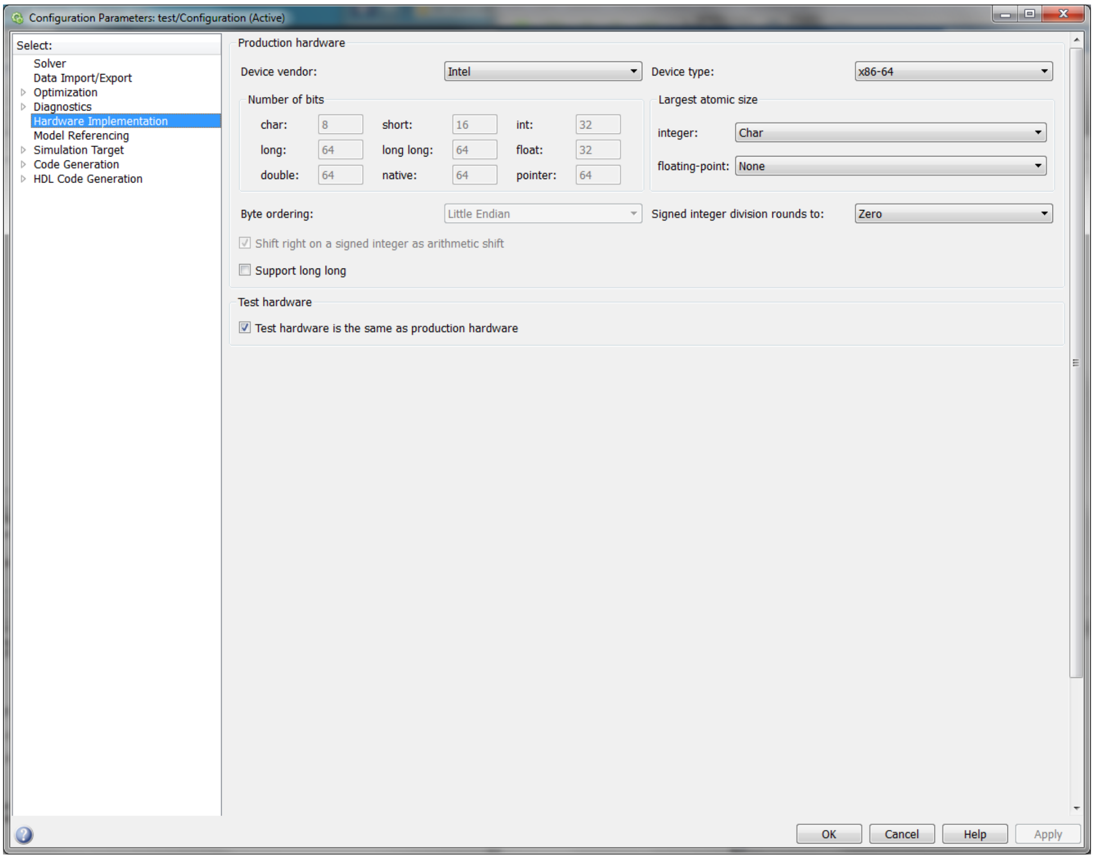

Figure 13: Configuration settings for Hardware Implementation.

- **Maximum missed ticks** : 10
- **Input channels** : [1]
- **Quadrature mode** : quadruple
- **Reset input function** : none
- **Output data type** : double
- set the connections of International Instruments PCI-6221 for this block by clicking on 'Board setup' button

Refer to Figure 15 for a screenshot of the settings. Block `Encoder Input` utilizes to output the information from encoder. The number of `Encoder Input` blocks in the Simulink model equals the number of the encoders that need to be used. Add a block for each encoder (i.e., the pendulum encoder and the motor arm encoder). Double-click on the block to open the settings dialog and set the 'Input channels' appropriately (each encoder should have a different channel number which is highlighted in red ellipse in the figure). Refer to Figure 16 to see an example with two encoder Simulink blocks added.

3. For each encoder block, add a `Gain` block (found in the 'Math Operations' library) followed by a `Terminator` block (found in the 'Sinks' library). The `Gain` block is used to convert the output of the encoder to a physical measurement (i.e., degrees or radians). The appropriate conversion factor will be

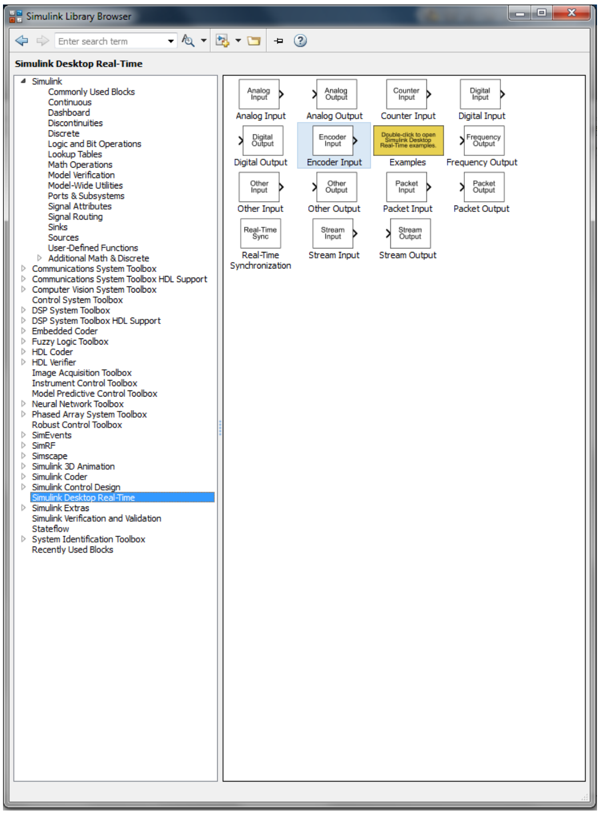

Figure 14: `Encoder Input` Simulink block.

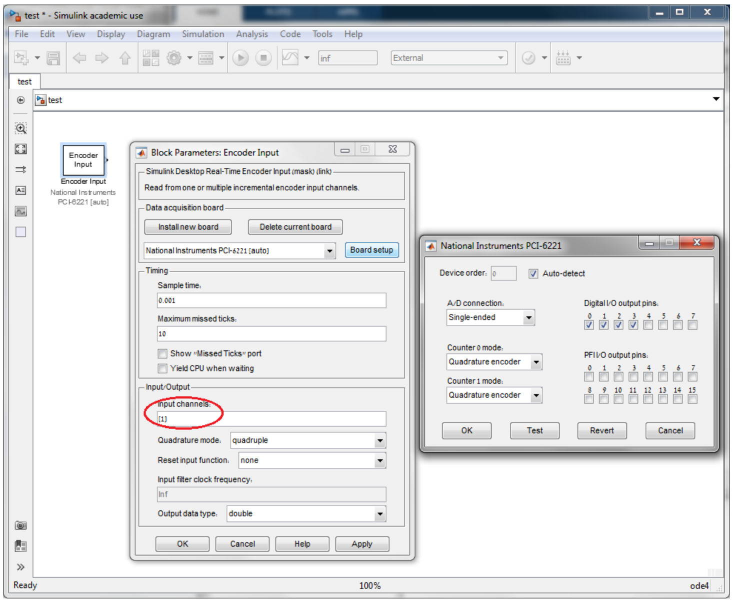

Figure 15: `Encoder Input` settings for the first encoder.

determined in the corresponding laboratory course. Refer to Figure 17 for a screenshot of this setup.

Add the D/A Converter : In the Simulink Library Browser under the `Simulink Desktop Real-Time` library is a special block for the digital-to-analog con-

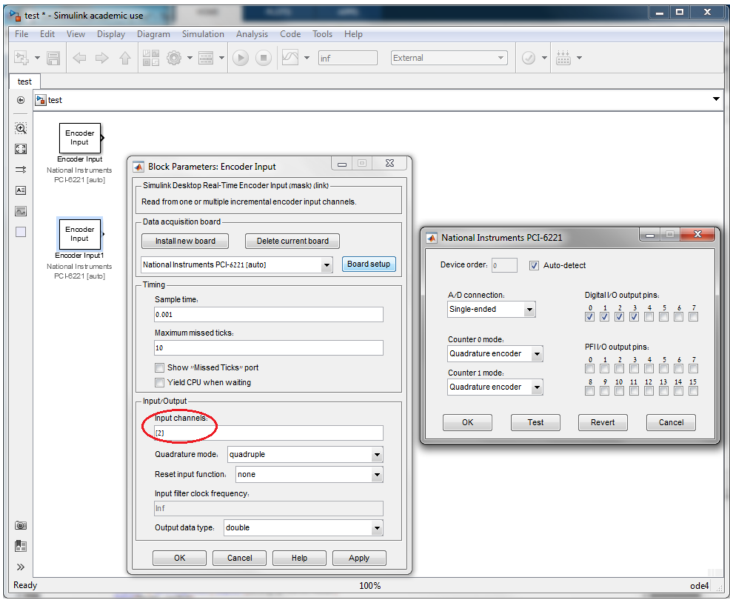

Figure 16: `Encoder Input` settings for the second encoder.

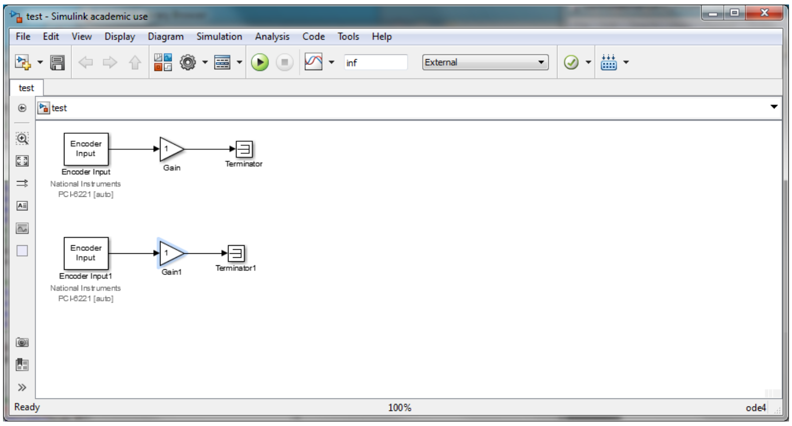

Figure 17: Setting up the encoders.

verter. This block will be used to convert a digital signal to an analog signal that will then be sent to the DC motor as voltage.

1. Open the Simulink Library Browser, and click on the `Simulink Desktop Real-Time` library. Under the `Simulink Desktop Real-Time` library, select the `Analog Output` block (this block is located above the `Encoder Input` block in Figure 14).

2. Add the block `Analog Output` to the Simulink model. Double-click on the block to open the settings dialog and refer to Figure 18 for a screenshot (The settings dialog for this block is same as the one for `Encoder Input` block except 'Input/Output' part).

The input to the `Analog Output` block can come from a variety of sources within the Simulink model. For example, it might come via a feedback path from an encoder output, or directly from a `Constant` block from the 'Sources' library. Figure 19 shows one possible setup operating the DC motor. A `Constant` block is con-

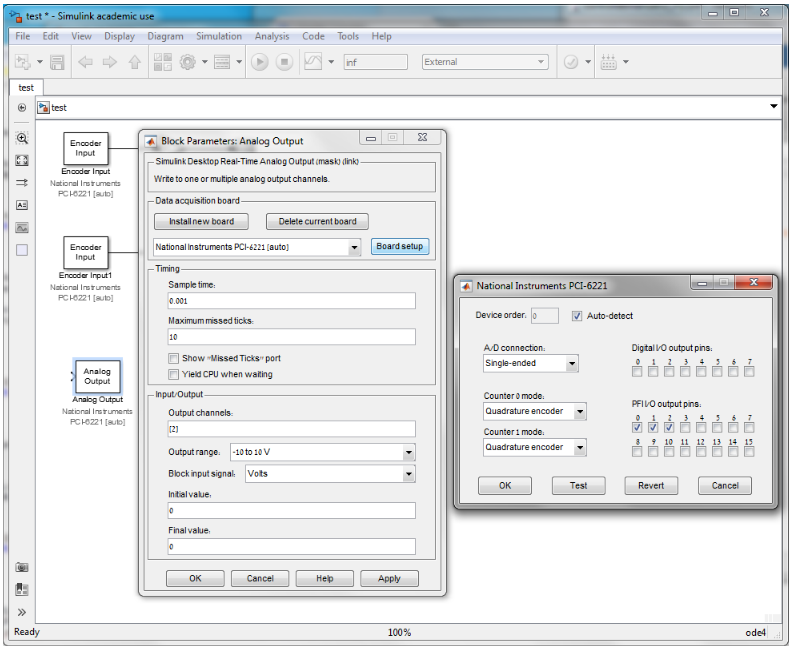

Figure 18: Setting up `Analog Output` block for the DC motor voltage.

nected to a `Transfer Function` block that acts as a pre-filter to the DC motor voltage. Then the `Transfer Function` block transmits the signal to the `Saturation` block with ± 1 limits. It is a common assumption that the control signal, which can be the output from the `Transfer Function` block, is normalized to ± 1 range, and the output from the `Constant` block belongs to the interval $[0, 1]$. On the other hand, the output range of the `Analog Output` block is ± 10 Volts (see Figure 18). So between the `Saturation` and `Analog Output` blocks should be a `Gain` block with the gain value of 10.

Add the Scope : In the Simulink Library Browser under the `Sinks` library (see Figure 8) is the `Scope` block, which displays inputs signals with respect to simulation time and (if this option is selected) saves data to the MATLAB workspace.

1. Open the Simulink Library Browser, and click on the `Simulink` library. Under the `Sinks` library, select the `Scope` block.

2. Add the block `Scope` to the Simulink model. Double-click on the block to open the settings dialog. On the `Scope` window toolbar is a button labeled 'Try Time Scope' (refer to Figure 20 for a screenshot; the button is circled in red).

   - press the 'Try Time Scope' button in order to migrate `Scope` to `Time Scope` block which is more effective than `Scope` block (refer to Figure 21 for a screenshot)

   - press the 'Configuration Properties...' button (this button is circled in red ellipse in Figure 21)

     – the options in the 'Time' pane can be selected as in Figure 22

     – in order to save data to the MATLAB workspace 'Logging' pane can be selected as in Figure 23

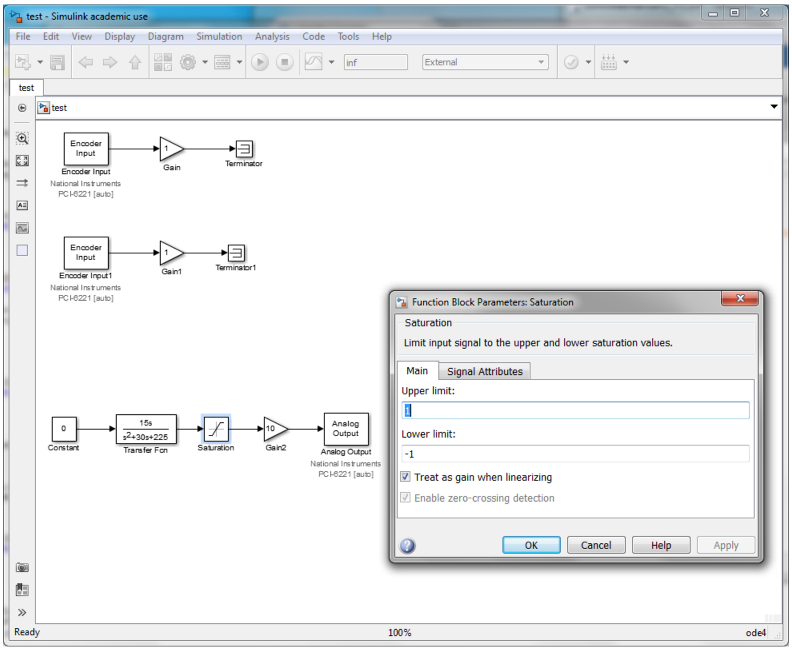

Figure 19: Simulink model with encoder and DC motor voltage blocks.

– 'Display' and 'Main' panes are described for our purposes without referring to screenshots: 'Display' pane sets up the range of the Y-axis, and 'Main' pane ensures that the `Scope` block opens when the simulation starts (check box `Open at simulation start` should be selected).

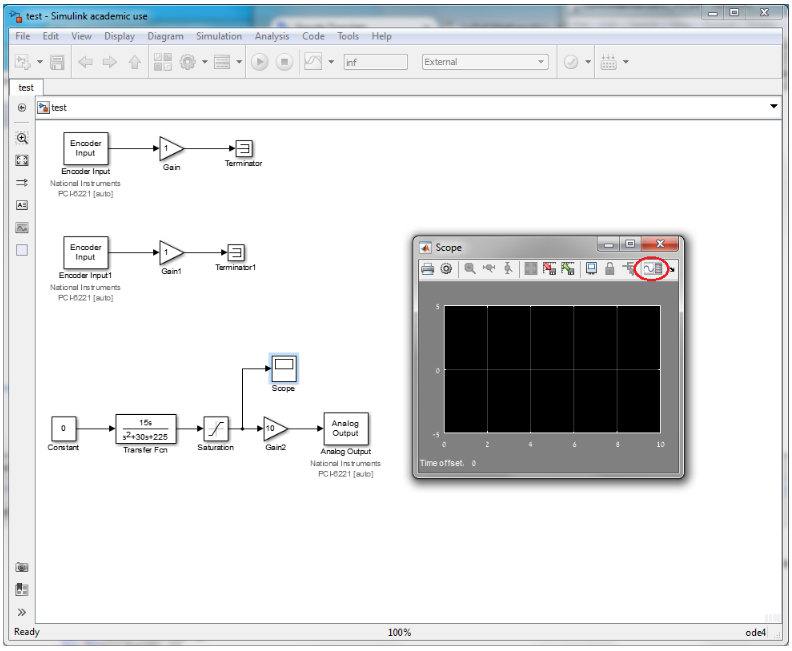

Figure 20: `Scope` block.

Real-Time Simulation : After the configuration settings are changed, and the appropriate encoder and DC motor voltage blocks have been added to

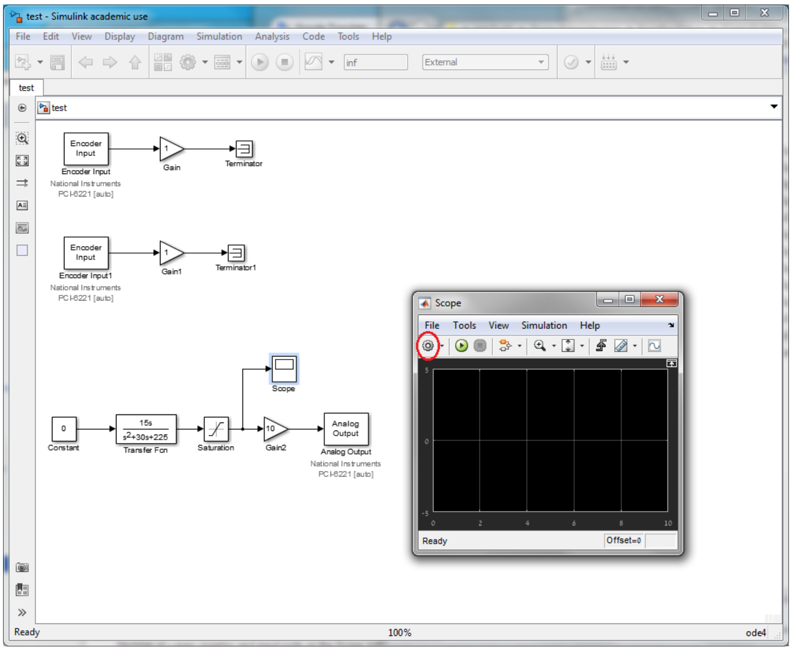

Figure 21: Migrating `Scope` to `Time Scope` block.

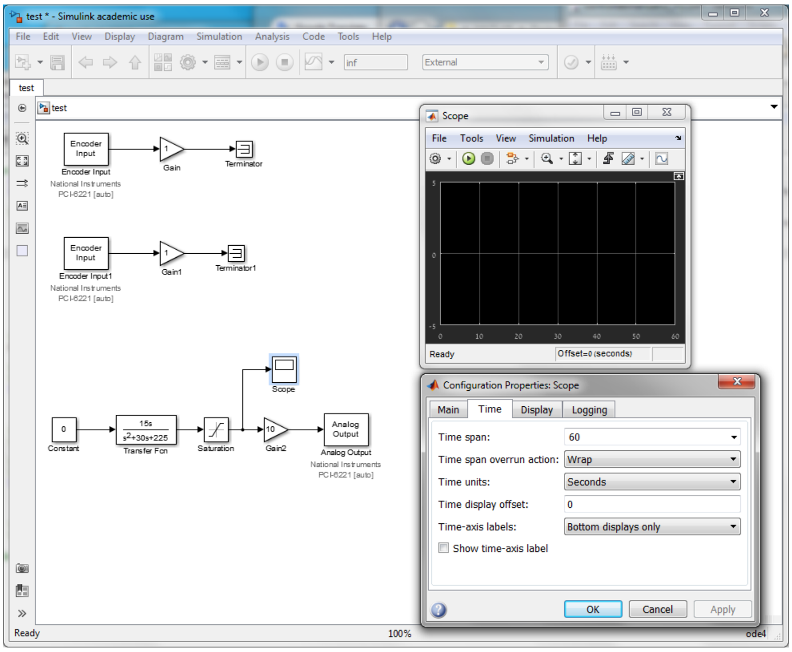

Figure 22: ‘Time’ pane of the ‘Configuration Properties...’ button in the `Time Scope` block.

the Simulink model, it is ready for the real-time simulation. There are two basic options for running real-time simulation: *Normal Mode* and *External Mode*. Details on the differences between normal mode and external mode can be found on the Mathworks website (`http://www.mathworks.com/help/rtwin/getting-started-with-real-time-windows-target.html`).

The main points to consider is in normal mode, the simulation

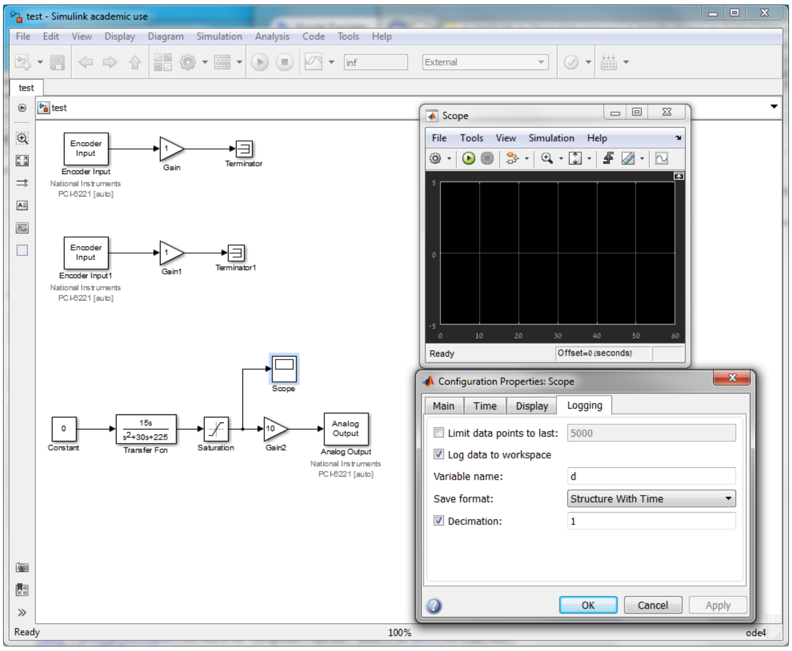

Figure 23: 'Logging' pane of the 'Configuration Properties...' button in the `Time Scope` block.

runs entirely in Simulink and is not synchronized with a real-time clock. The simulation can be synchronized to a real-time clock using Simulink Desktop Real-Time I/O blocks, but this can lead to 'missed' real-time clock ticks (depending on the available resources of the computer, complexity of the model, etc.). This might cause problems for high-performance systems.

In External mode, the Simulink Coder generates C code from the simulation algorithm and dynamically links it with I/O driver code generated from the I/O blocks. The resulting executable runs in the operating system kernel mode on the host computer and exchanges parameter data with Simulink via a shared memory interface. The External mode executable is fully synchronized with the real-time clock. The main role of Simulink is to read and display simulation results returned from the executable.

Now the Normal and External modes are described.

1. **Normal mode** (see Figure 24).
   The basic setup for Normal mode is minimal. In the simulation model dialog box:

   - select 'Simulation -> Mode -> Normal'
   - set the **Maximum Missed Ticks** value to 100 for block parameters in the `Encoder Input` and `Analog Output` blocks
   - run the simulation by clicking the 'Run' button (in order to stop simulation, click the 'Stop' button)

! → *The 'Run' and 'Stop' buttons can be found in the `Time Scope` block too. After you push on the 'Stop' button, all data from the start until the stop in the simulation transfers to the MAT-*

LAB workspace with the variable name 'd' (see the `Time Scope` *block settings for 'Logging' pane in Figure 23). In order to plot the data use the following MATLAB code:*
t=d.time;
y=d.signals.values;
plot(t,y); grid

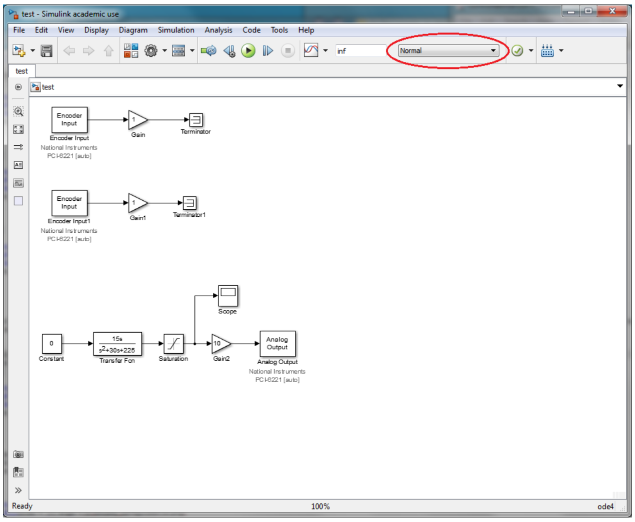

Figure 24: The Simulink model in Normal mode.

2. **External mode** (see Figure 25).
   The basic setup for External mode is:

   (a) set Code Generation parameters for creating C code and building a real-time application by Simulink Coder (see Figure 12 and Figure 13)

   (b) create an executable target application by clicking the 'Deploy to Hardware' icon in the toolbar (see Figure 26)

   (c) enter scope parameters for signal tracing (Simulink in External mode connects the Simulink model to the corresponding real-time application, and this connection allows to use the Simulink block diagram as a graphical user interface as in Normal mode):

   - open the configuration parameters dialog window (Simulation -> Model Configuration Parameters).
   - select the 'Code Generation -> Simulink Desktop Real-Time' node, and 'Simulink Desktop Real-Time' pane opens
   - enter the parameters in the 'Simulink Desktop Real-Time' pane according to Figure 27, and click 'OK' button
   - in the Simulink model from Figure 25 click Code -> External Mode Control Panel (see Figure 28) and the 'External Mode Control Panel' dialog box opens (see Figure 29):
   - click the 'Signal & Triggering...' button (highlighted

by the left red ellipse) and the 'External Signal & Triggering' dialog box opens

- enter the parameters in the 'External Signal & Triggering' dialog box according to Figure 30 (the value of 60000 in the 'Duration' field indicates the number of sample points in a data buffer), and after clicking on 'OK' button we see Figure 29 with the 'External Mode Control Panel' dialog box again
- click the 'Data Archiving...' button (highlighted by the right red ellipse) and the 'Data Archiving' dialog box opens
- enter the parameters in the 'Data Archiving' dialog box according to Figure 31, and after clicking on 'OK' button we see Figure 29 with the 'External Mode Control Panel' dialog box again
- in order to close the 'External Mode Control Panel' dialog box click 'OK' again

(d) run the real-time application by clicking the 'Run' button (in order to stop simulation, click the 'Stop' button)

! → *The 'Run' and 'Stop' buttons can be found in* `Time Scope` *block too. After you run the real time application (with* **Sample Time** *= 0.001 seconds), the MATLAB workspace receives data corresponding to the 'Data Archiving' dialog box in Figure 31 and the 'External Signal & Triggering' dialog box in Figure 30. Namely, every 60 seconds, the MATLAB workspace receives a new file. For example, 'data_0.mat' with variable 'd_0', 'data_1.mat' with variable 'd_1', ... and so on. After clicking on the 'Stop' button, the MATLAB workspace receives the last file (for example, 'data_3.mat' with variable 'd_3'). The real-time application has transferred three files in total, each with 60000 samples except the last file (which contains less than 60000 samples, corresponding to the stop time). Furthermore, each file consists of the variable with name of the 'd_i' form, where the 'd' part is from the* `Time Scope` *block settings for 'Logging' pane in Figure 23, and the 'i' part coincides with the corresponding '*.mat' file number. In order to plot the data, first load all files into the MATLAB workspace and then use the following MATLAB code (assume the four files from the example above):*

```
t=[d_0.time; d_1.time; d_2.time; d_3.time];
y_0=d_0.signals.values;
y_1=d_1.signals.values;
y_2=d_2.signals.values;
y_3=d_3.signals.values;
y=[y_0; y_1; y_2; y_3];
plot(t,y); grid
```

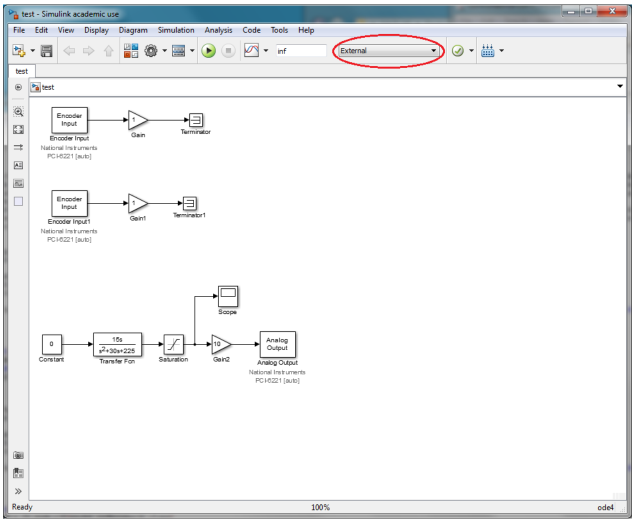

Figure 25: The Simulink model in External mode.

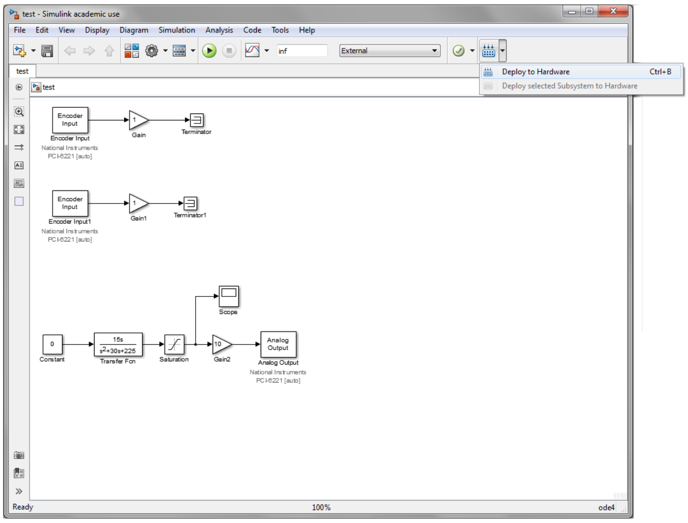

Figure 26: The Simulink model in External mode: Deploy to Hardware.

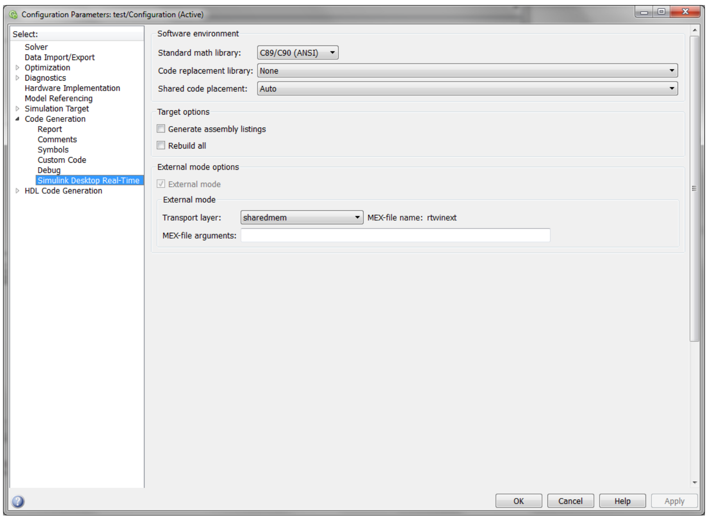

Figure 27: The 'Simulink Desktop Real-Time' pane.

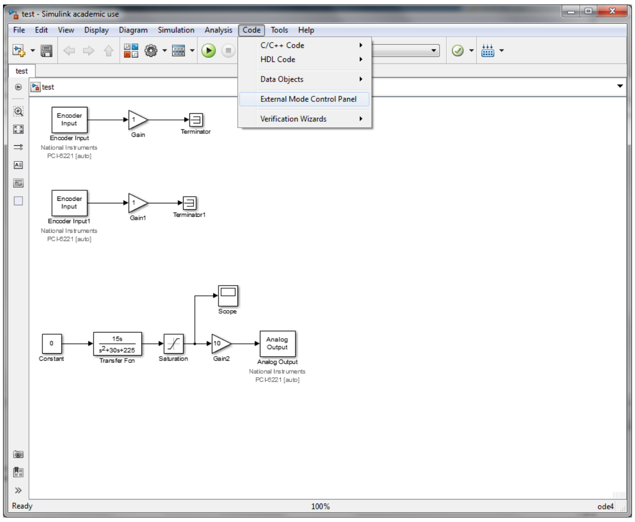

Figure 28: Opening ‘External Mode Control Panel’ dialog box.

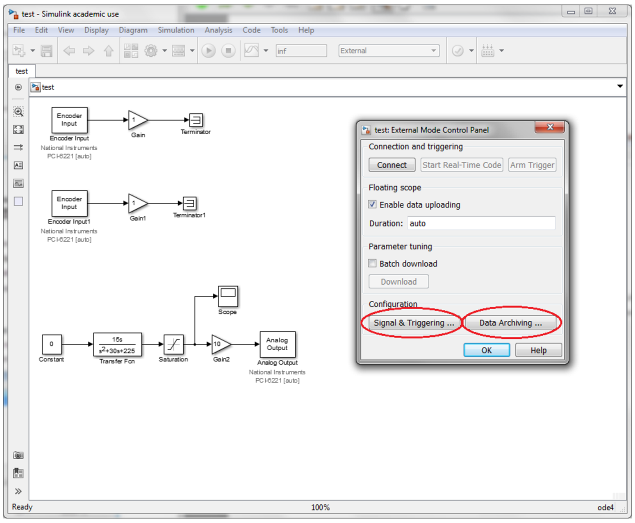

Figure 29: ‘External Mode Control Panel’ dialog box.

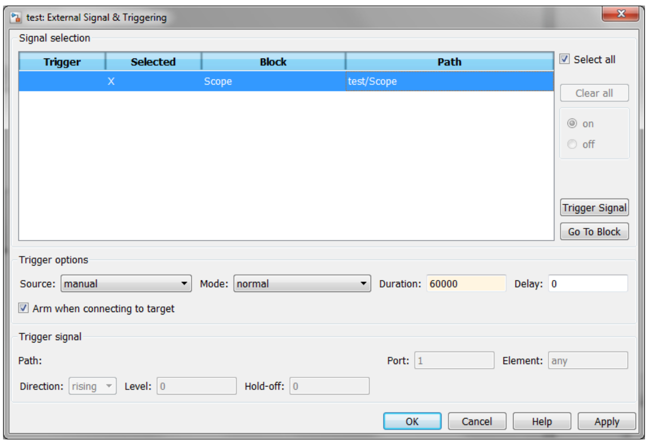

Figure 30: ‘External Signal & Triggering’ dialog box.

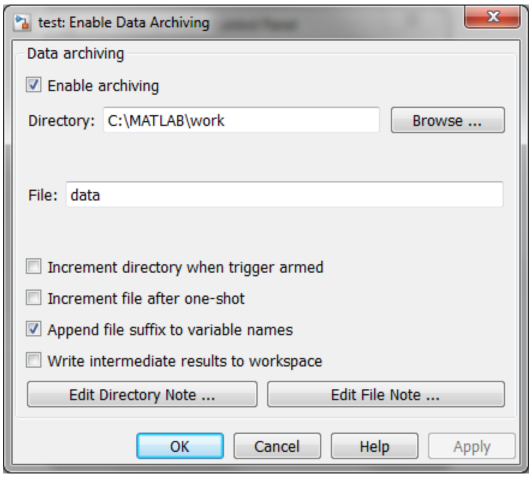

Figure 31: ‘Data Archiving’ dialog box.