Experiment 01
Familiarization of dSPACE

Aim: To study a typical data acquisition and controller system used in power electronics control implementation and electrical drives applications.

Basic Steps: In order to perform an experiment, the following basic steps must be performed:

1. Model system in Simulink,
The system can be built in Simulink using any of the available libraries and may or may not include blocks from the dSPACE library. The dSPACE library includes blocks that provide graphical representation of physical connections to the dSPACE board. Using the blocks in this library allows a physical system to be controlled by dSPACE.

2. Download system to dSPACE **board (DS 1104)**
Once the model is built in Simulink, it must be downloaded to the dSPACE board. This can be accomplished by using the Real-Time Workshop in MATLAB.

3. Build a Control Desk Layout for the system
Once the model has been downloaded to the dSPACE board, a layout must be built in Control Desk to control the system. This interface can be used to start and stop the experiment, change
parameters of the system in real time, display values of parameters in real time, and log data from the experiment.

4. Run Experiment
After successfully loading the model onto the board and building a layout, the experiment can be run.

5. Export data to MATLAB for analysis
Control Desk can log data from the experiment and save it as file that can be exported directly to MATLAB. This allows the data to be manipulated by any tool available in the entire MATLAB program.

2.1 Introduction

One of the best features of the dSPACE package is the ease of building real-time applications. The time between converting the design into digital instructions for the DSP and effectively running the application depends only on how fast your computer can compile the initial code.

Basically, a real-time application can be created by means of two methods:

1. Using MATLAB/Simulink for building the model and automatically generate the C code and download it into the DSP memory.

2. Hand-coding in C and compile the model into DSP code.

The fastest way of developing a real-time code is developing the model in Simulink and preparing a real-time model from that. Basically, once you have completed the Simulink model which you want to run in real-time, the only command required is **RTW Build** under **Tools** menu in **SIMULINK**.
Once the command is executed, dSPACE software creates the object (*.obj) file, downloads it on DS1104 board and automatically starts the hardware execution. However, there are some important settings you have to make before "transporting" your model into the real-time world. Let's start with a simple example.

2.2 Creating a model in Simulink

Our first example introduces the analog channels input-output communication with external devices. For this example, a Signal Generator is used to generate different waveforms as inputs to our signal processing algorithm. The result of this process will be directed to an analog output channel, in order to be monitored with the lab oscilloscope.
Suppose that we need to analyze the response of a second-order system at different types of input signals, with variable amplitudes. The second-order system is defined by the following parameters:

1. Damping (ξ) = 0.7, Natural frequency (ωn) = 20 Hz
2. We need its response at different types of input signals such as Sine-wave, Square-wave, and Saw-tooth wave.
3. The input has a variable gain, in the range of [0 ... 5].
4. Firstly, simulation model will be developed using a model for signal generator and model for Oscilloscope.
5. Create a folder expt-02.
6. Start MATLAB and set the Path Browser to your working folder (expt-02).
7. Type Simulink at the prompt line and create a new model from **File** menu.
8. Choose from the Simulink *Continuous library* the **Transfer Function** block and drag it into a new simulation model.

The second order system, with the parameters specified, can be described in transfer function with the following relation:

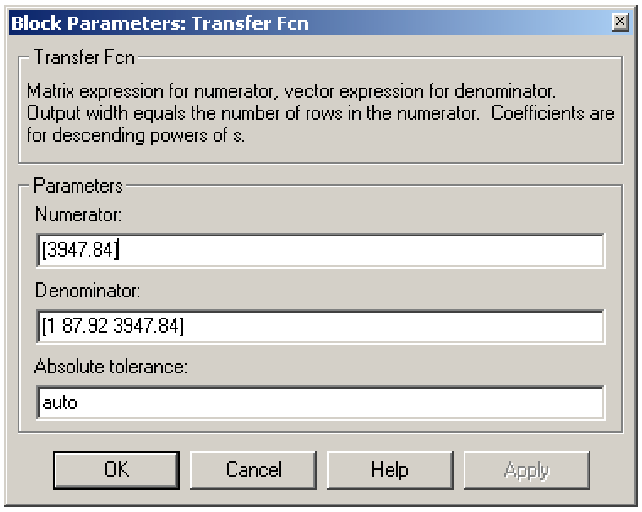
$$G(s) = \frac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} \qquad (2.1)$$

where $\omega n = 2\pi f = 2\pi.20 = 62.63$ and $\xi = 0.7$ The numerical model becomes

$$G(s) = \frac{3947.84}{s^2 + 87.962s + 3947.84} \qquad (2.2)$$

1. Set the parameters of the **Transfer Fcn** block as shown in Fig. 2.1

2. Next, drag a **Signal Generator** block from *Sources* library, a **Slider Gain** from *Math* library and a **Scope** from *Sinks* library. Connect all blocks as in Fig. 2.2.
3. Now, set the simulation parameters by pressing Ctrl-E as:


Block Parameters: Transfer Fcn

Transfer Fcn

Matrix expression for numerator, vector expression for denominator. Output width equals the number of rows in the numerator. Coefficients are for descending powers of s.

Parameters

Numerator: [3947.84]

Denominator: [1 87.92 3947.84]

Absolute tolerance: auto

Figure 2.1: Parameters for Second Order System

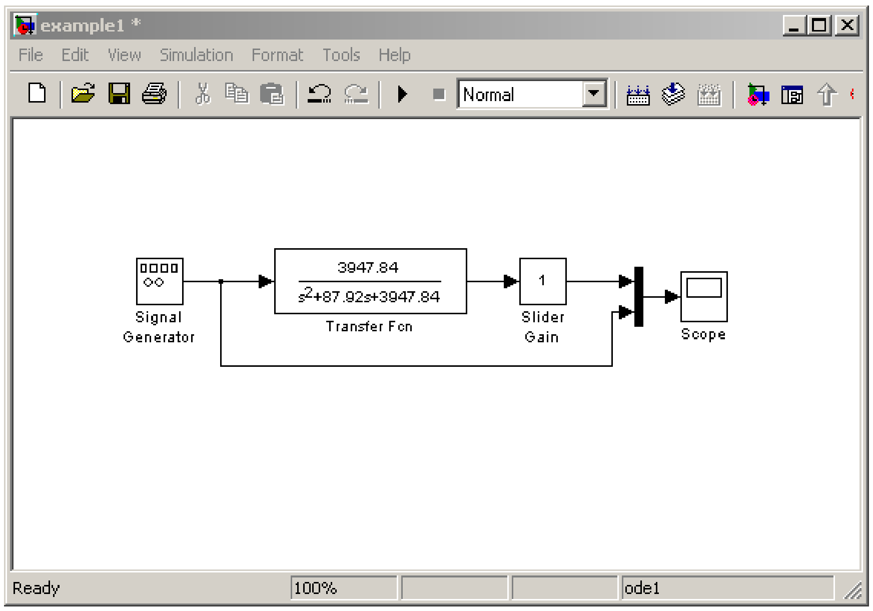


Figure 2.2: Simulink Model of 2*nd* Order System

– Set the **solver options** to *fixed step* and *ode1*.
– You need to specify a fixed step size. Let it be “0.001” i.e 1ms.
– Stop-time as “2” seconds.

- Set the Signal Generator output to *Square-wave* and frequency as “2Hz”.
- Keeping other parameters unchanged.

- Run the simulation by pushing the triangular play button and adjust the display of the Scope with the autoscale option. You will obtain a plot similar to the one shown in Fig 2.3.

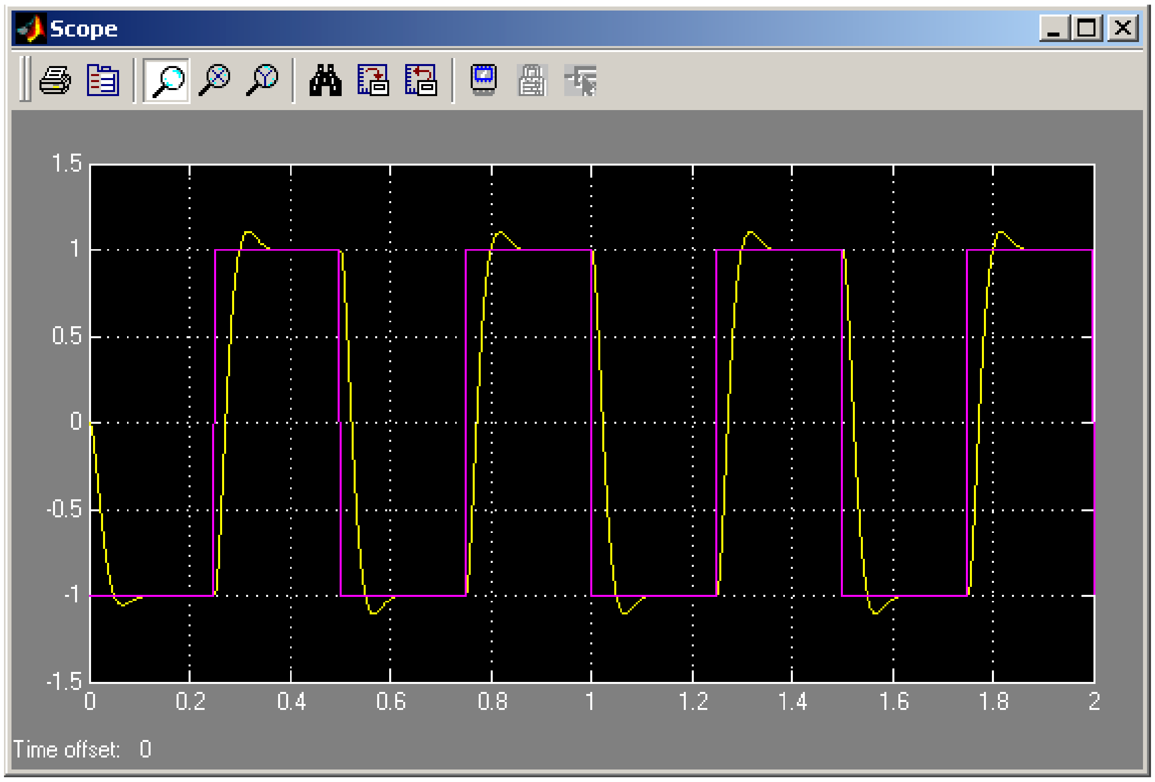


Figure 2.3: Simulation Results of 2*nd* Order System for Square Wave Input

This simulation was really fast. The generator waveform can be changed by double-clicking the signal generator block. Now, since we have the idea how the system work, we will implement the system in “real-time” and observe the results.

2.3 Build the real-time simulation model

The model developed for simulation is now to be connected to the external devices (Signal generator and Oscilloscope). Since these devices are physically generating/accepting signals going to or coming from the DSP board, we need to stream these signals via the analog Input/Output channels, located on the controller box.

Firstly, make sure that the Signal Generator and the Oscilloscope are connected via shielded BNC cables to ADC#5 and DAC#1 respectively. The Signal Generator output is set such that its signal amplitude is approximately 1V.

Communication with the input/output channels is performed via two dSPACE blocks found in the *dSPACE RTI1104* library under the sub-library *DS1104 MASTER PPC*, named **DS1104 ADC C5** and **DS1104 DAC C1**. They will replace our **Signal Generator** block and **Scope** block respectively. The analog input channel is down-scaled by the hardware with a ratio of 1:10. This means that 10 V at the input will be read as 1V in our model. The analog output channel is also down scaled in the hardware with the same ratio. Thus, a 1 V signal generated within the model will have amplitude of 10V at the connector. Thus, two **Gain** blocks, from the **Math** Library, will be required to correctly read and write the values from and to the analog channels.

- Drag the **DS1104ADC C5** and **DS1104DAC C1** blocks into the Simulink model and replace the Signal generator and Scope blocks. Place the two gains of 10 and 0.1 on the input and output signals respectively. The model should become similar to the one depicted in Fig. 2.4. You might need to save the model with another name, to preserve the simulation model.

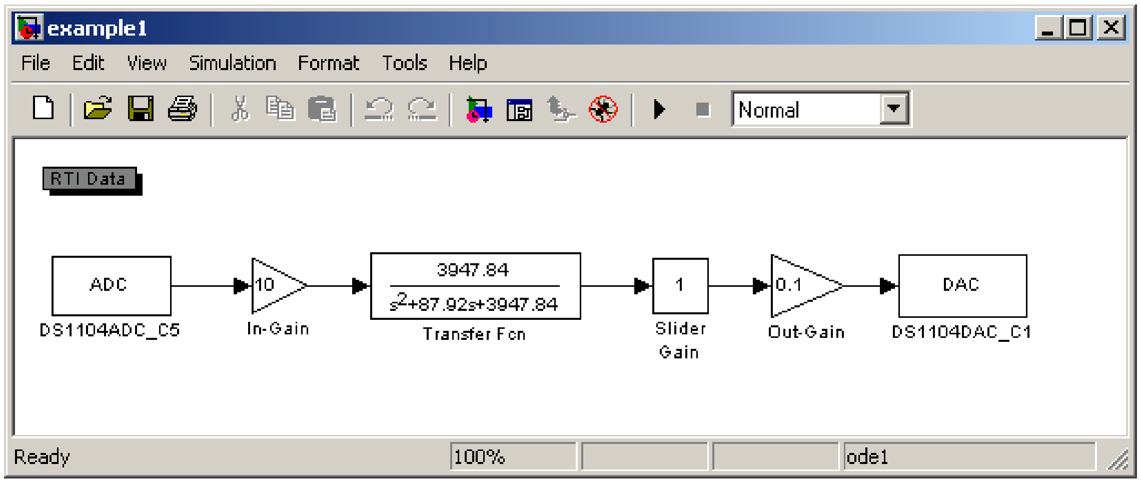


Figure 2.4: Real-Time model for dSPACE in Simulink.

Remember that we performed the simulation for only 2 seconds. In real-time, however, the system needs to run continuously. Therefore, press CTRL-E for simulation parameters and

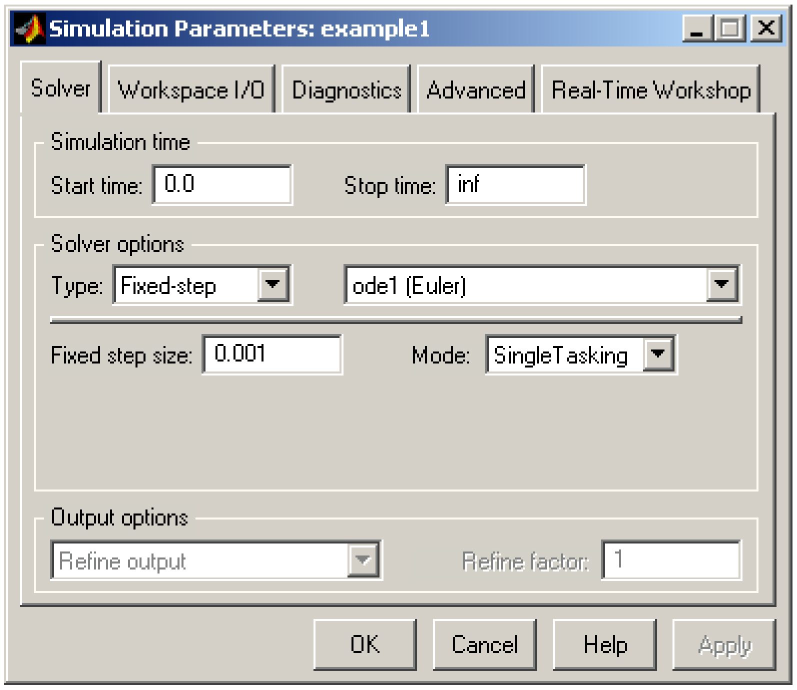
- Set the **Stop-Time** as ‘**inf** ’. The simulation parameters should look like shown in Fig. 2.5
- In Simulation Parameters, go to “Advanced” and make **Block Reduction** “**OFF**”.
- Next, choose the **Real-Time Workshop**, **Build Model** from the **Tools** menu.

Once the above command is given, you will observe a list of messages displayed in Matlab Command window. These messages correspond to the different steps that the RTI Software perform in order to transform the Simulink code of the *example1.mdl* file into DSP code.

First there is a compilation stage, in which the Simulink file is transformed into a C file, then comes the link stage where all the variables and subroutines are correlated with the DSP environment, and finally the code is transformed into an object file and downloaded into the DSP memory. The MATLAB window screen will look like shown in Fig. 2.6. You can see that the file was successfully built. The result is “example1.obj” which was already loaded in the DSP memory and its execution started.

Please note that the directory in which the model was built is the same you choose for creating the Simulink model. If you look now in this directory you will find several files, generated during the build command. Due to the large number of files generated, it is advisable that each project to be located in a separate sub-directory.

Now, our simulation is running in the DSP board, in a digital, real-time format. We can predict that this is much faster than what we saw on the Scope screen, in Simulink. More important, now, seems to be the interaction with the system. We need to visualize, modify and analyze the variables. For this, dSPACE comes with its own Graphical User Interface, called **CONTROL DESK**. So, let’s learn how to use the Control Desk to interact with the real-time simulation.



Simulation time

Start time: 0.0 Stop time: inf

Solver options

Type: Fixed-step ode1 (Euler)

Fixed step size: 0.001 Mode: SingleTasking

Output options

Refine output Refine factor: 1

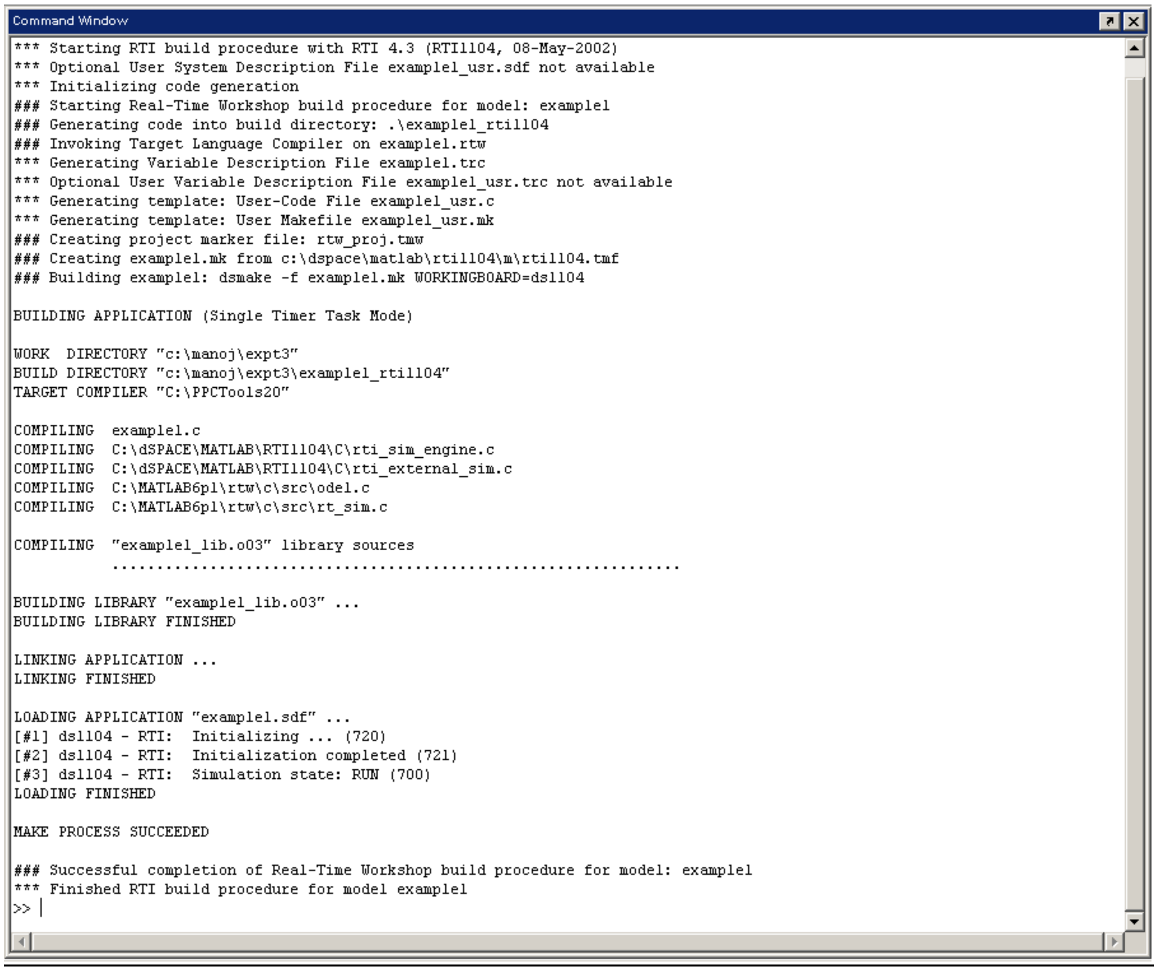
Figure 2.5: Simulation Parameter setting for real-time model

2.4 Creating a new experiment file with Control Desk

ControlDesk is a software that allows the user to look at the variables, display their behavior and modify the simulation parameters by interacting directly with the DSP board.

– Start ControlDesk and select only the toolbars checked as shown in Fig. 2.7. The Tool Window is displayed at the bottom of Control Desk screen as shown in Fig. 2.7. The Tabs display the tool currently used. In the figure only three of the working tools are available: *Log Viewer*, *Interpreter* and *File Selector*.

As we shall further describe, there is one tool, very important, to which we shall give a special attention. This tool is called Variable Browser and the Parameter Editor. It provides access to the variables of an application. These variables are stored in a file called **example1.sdf**.



```
*** Starting RTI build procedure with RTI 4.3 (RTI1104, 08-May-2002)
*** Optional User System Description File example1_usr.sdf not available
*** Initializing code generation
### Starting Real-Time Workshop build procedure for model: example1
### Generating code into build directory: .\example1_rti1104
### Invoking Target Language Compiler on example1.rtw
*** Generating Variable Description File example1.trc
*** Optional User Variable Description File example1_usr.trc not available
*** Generating template: User-Code File example1_usr.c
*** Generating template: User Makefile example1_usr.mk
### Creating project marker file: rtw_proj.tmw
### Creating example1.mk from c:\dspace\matlab\rti1104\m\rti1104.tmf
### Building example1: dsmake -f example1.mk WORKINGBOARD=ds1104

BUILDING APPLICATION (Single Timer Task Mode)

WORK  DIRECTORY "c:\manoj\expt3"
BUILD DIRECTORY "c:\manoj\expt3\example1_rti1104"
TARGET COMPILER "C:\PPCTools20"

COMPILING  example1.c
COMPILING  C:\dSPACE\MATLAB\RTI1104\C\rti_sim_engine.c
COMPILING  C:\dSPACE\MATLAB\RTI1104\C\rti_external_sim.c
COMPILING  C:\MATLAB6p1\rtw\c\src\ode1.c
COMPILING  C:\MATLAB6p1\rtw\c\src\rt_sim.c

COMPILING  "example1_lib.o03" library sources
           ................................................................

BUILDING LIBRARY "example1_lib.o03" ...
BUILDING LIBRARY FINISHED

LINKING APPLICATION ...
LINKING FINISHED

LOADING APPLICATION "example1.sdf" ...
[#1] ds1104 - RTI:  Initializing ... (720)
[#2] ds1104 - RTI:  Initialization completed (721)
[#3] ds1104 - RTI:  Simulation state: RUN (700)
LOADING FINISHED

MAKE PROCESS SUCCEEDED

### Successful completion of Real-Time Workshop build procedure for model: example1
*** Finished RTI build procedure for model example1
>> |
```

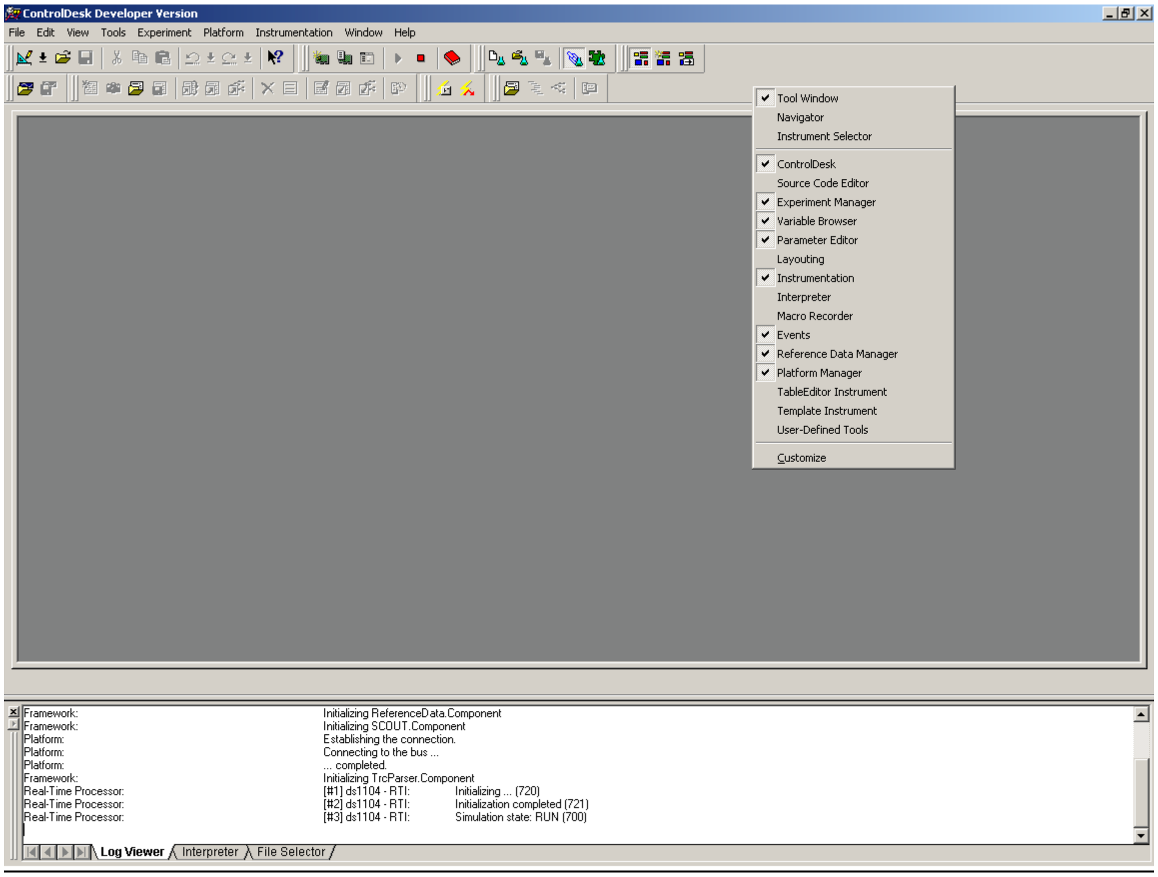
Figure 2.6: Compilations details developed by dSPACE and SIMULINK in MATLAB Screen.

Thus, for handling the variables of a simulation we must load the *.sdf file before starting the graphical design.

- First we start a new experiment. Click **File/New Experiment**. In the pop-up window
write the name of the experiment, and very important, set the path where the simulation files are stored. (see Fig. 2.8). **Note** When creating a new experiment, don't forget to set correct working directory.
- Next, load the file containing the variables of the simulation. Click **File/Open Variable**

File and select example1.sdf.

The **Variable Manager Tab** appears at the bottom of the screen (see Fig. 2.9). The window contains the structure of the simulation model. At the highest level we see the simulation control variables. Their function is described in Table1.



```
Framework:                  Initializing ReferenceData.Component
Framework:                  Initializing SCOUT.Component
Platform:                   Establishing the connection.
Platform:                   Connecting to the bus ...
Platform:                   ... completed.
Framework:                  Initializing TrcParser.Component
Real-Time Processor:        [#1] ds1104 - RTI:     Initializing ... (720)
Real-Time Processor:        [#2] ds1104 - RTI:     Initialization completed (721)
Real-Time Processor:        [#3] ds1104 - RTI:     Simulation state: RUN (700)
```

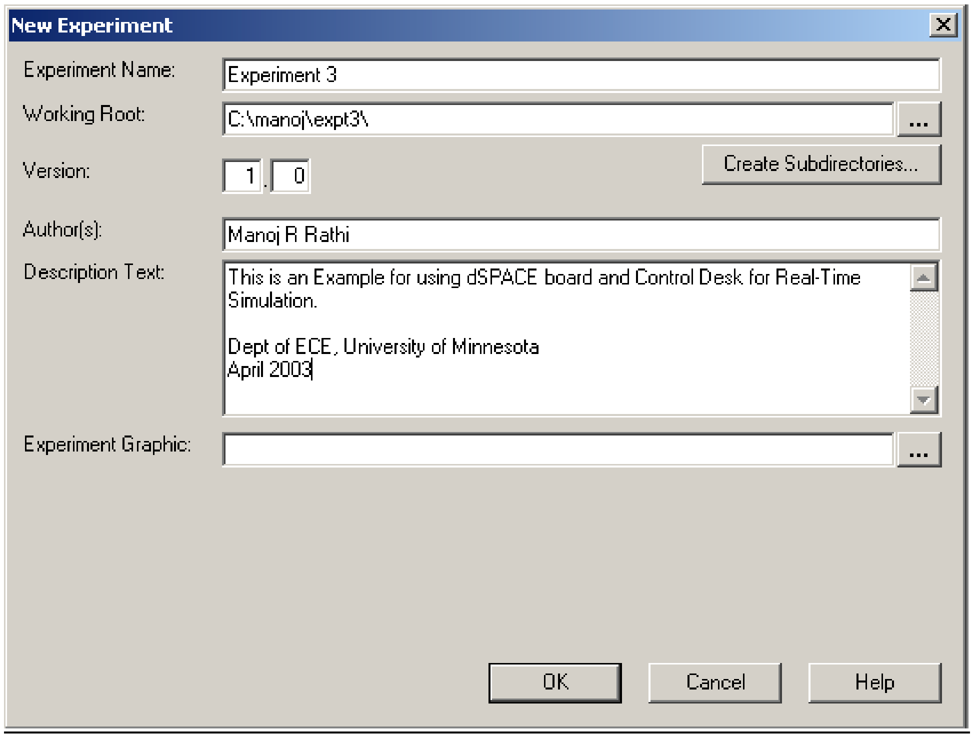
Figure 2.7: Control Desk Screen

For the purpose of our simulation, the variables of interest are contained under the **Model Root** group. This group contains the variables belonging to the top-level of the Simulink model. Variables from subsystems go into further groups in deeper hierarchical levels. The variables available have a prefix to distinguish the different variable types and are generated in the following order as detailed in Table. 2.1.

Prefix : Example	**Variable Type**
L: Output	labeled signals
B:Integrator	block outputs
S:Scope	inputs of signal sinks
P:Signal Generator Amplitude	block parameters

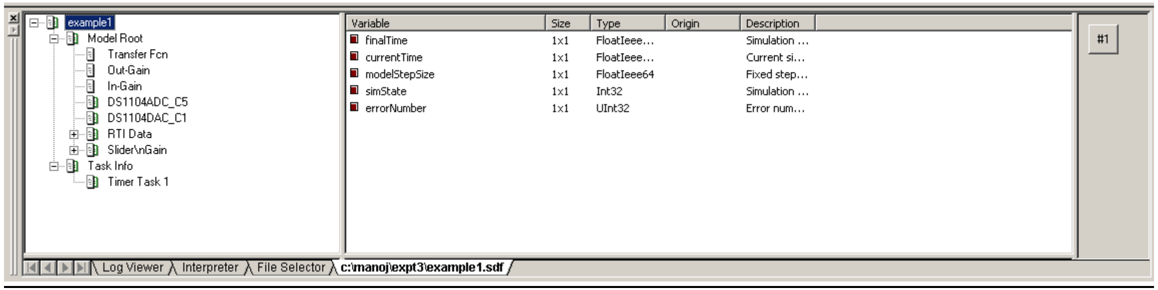
Table 2.1: Variable Types and Signal Correlation

The name following the prefix is the block name, except for labeled signals where the label itself is used. Here are all the variables corresponding to the **Model Root** for our example are detailed in Table. 2.2.



Experiment Name:	Experiment 3
Working Root:	C:\manoj\expt3\
Version:	1 . 0
Author(s):	Manoj R Rathi
Description Text:	This is an Example for using dSPACE board and Control Desk for Real-Time Simulation. Dept of ECE, University of Minnesota April 2003
Experiment Graphic:	

Figure 2.8: Experiment Settings



Variable	Size	Type	Origin	Description
finalTime	1x1	FloatIeee...		Simulation ...
currentTime	1x1	FloatIeee...		Current si...
modelStepSize	1x1	FloatIeee64		Fixed step...
simState	1x1	Int32		Simulation ...
errorNumber	1x1	UInt32		Error num...

Figure 2.9: Variable Tab Manager

2.5 Display Controls and Scopes with Instrumentation Management Tools

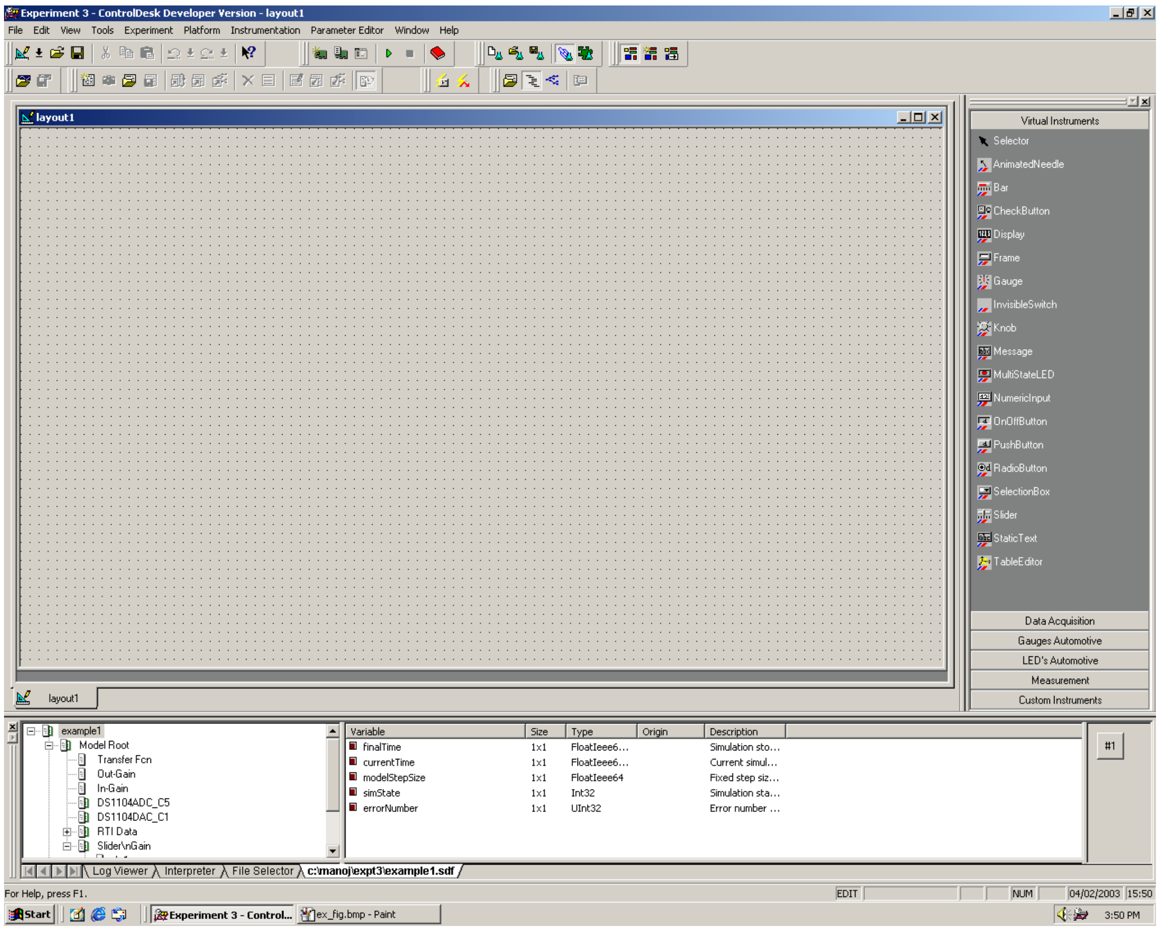
In order to see the behavior of each variable and modify the parameters in real-time, while the system is running, we need a series of buttons, knobs, slider-gains, plotter etc, that can handle these variables. Therefore, we need to start a new **LAYOUT** screen in which all those instruments can be added.

- Click File/New/Layout, from the menu.

Two new windows appear in the ControlDesk workspace as shown in Fig. 2.10. The one called Layout1, will contain the instruments used for managing the experiment. The second window is actually a toolbar which let us drag and drop the necessary controls for the experiment.

'Model Root/B:DS1104ADC_C5->ADC#5'	Display the variable received on A/D channel #5.
'Model Root/B:Slider Gain>Out1'	Display the output of Slider Gain.
'Model Root/B:In-Gain'	Display the output of the Input Gain block.
'Model Root/B:Out-Gain'	Display the output of the Output Gain block.
'Model Root/B:Transfer Fcn'	Display the output of the Second order Transfer Function
'Model Root/Slider Gain/P:Slider Gain.Gain'	Contains the value of Slider Gain.
'Model Root/P:In-Gain.Gain'	Contains the value of Input Gain.
'Model Root/P:Out-Gain.Gain'	Contains the value of Output Gain.
'Model Root/P:Transfer Fcn.A(1)'	Handle the transfer function parameters (numerator and denominator)
'Model Root/P:Transfer Fcn.A(2)'	
'Model Root/P:Transfer Fcn.C(1)'	
'Model Root/P:Transfer Fcn.C(2)'	

Table 2.2: Variables handled by dSPACE in the *example1* experiment.



Variable	Size	Type	Origin	Description
finalTime	1x1	FloatIeee6...		Simulation sto...
currentTime	1x1	FloatIeee6...		Current simul...
modelStepSize	1x1	FloatIeee64		Fixed step siz...
simState	1x1	Int32		Simulation sta...
errorNumber	1x1	UInt32		Error number ...

Figure 2.10: New Layout Window for Instrumentation and Control

The controls displayed in the Virtual Instruments toolbar let us handle only the variables that can be modified on-line, i.e. the **P**:-type variables. Our example has 5 such variables. Four of them were listed in Table. 2.2 and there is one more under the **Slider Gain** hierarchical level, which controls the amplitude of the Gain block.

1. Select the **SLIDER button** from the right toolbar.
2. The cursor changes into a square target. Click and hold the mouse while dragging a rectangular shape in the Layout1 window (see Fig. 2.11).

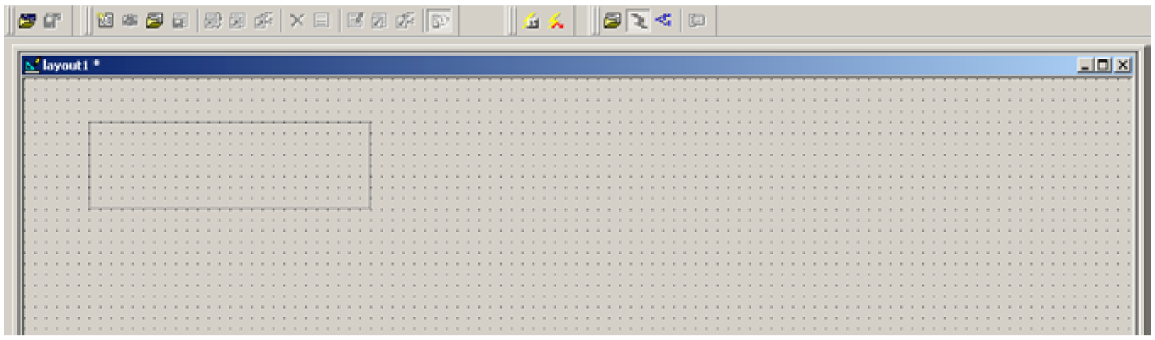
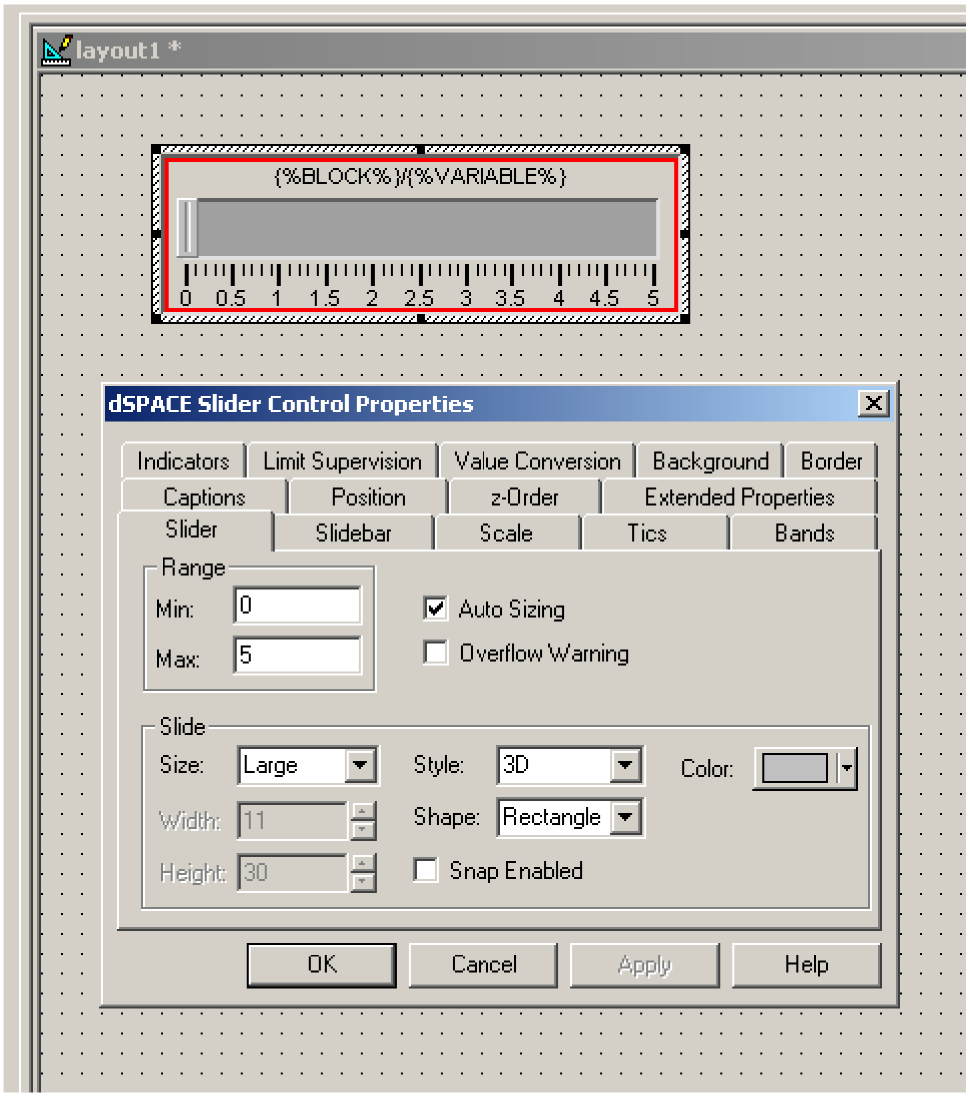


Figure 2.11: Selecting Slider Bar from Visual Instruments and drawing

• When the mouse is released, you will obtain a slider control, which let you continuously change any **P**:variable between the limits selected.
• As long as we decided to use amplifications between 0 and 5, double-click the Slider control, select the Slider Tab, and set the Range Min and Range Max as shown in Fig. 2.12.
• Now the limits where the cursor can be adjusted are 0 to 5. But the slider is still bordered with a red line. This means that it hasn't been assigned a variable to control yet.
• In the Variable Manager window, at the bottom of the screen, select the Slider Gain.
• Click the **P:Slider Gain.Gain** variable and drag it to the rectangle drawn in the Layout window.
• The new Slider control will display the handled variable and will no longer be bordered with a red line as shown in Fig. 2.13.
Now, we can add some visualization equipment. In our example, only two signals are worth monitoring: the system input, coming from the real signal generator connected to ADC#1 and the output of the second-order system. Both should be displayed on the same scope. The output signal will also be monitored with the Lab Oscilloscope connected to the DAC#1 channel, on the dSPACE Interface Box. Remember that the real values



dSPACE Slider Control Properties

Range
Min: 0
Max: 5

[✓] Auto Sizing
[ ] Overflow Warning

Slide
Size: Large
Style: 3D
Color:
Width: 11
Shape: Rectangle
Height: 30
[ ] Snap Enabled

Figure 2.12: Slider Control Setting Window

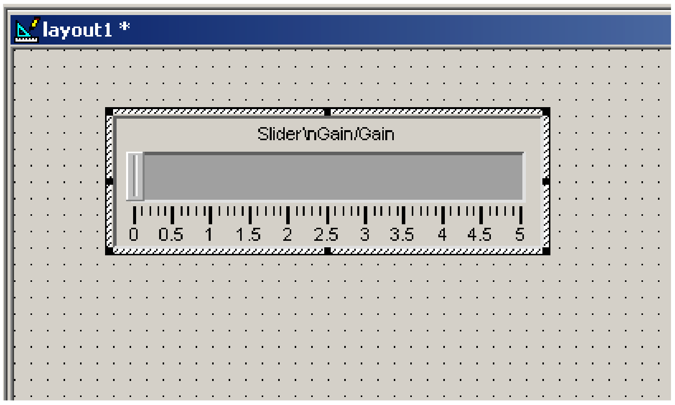


Figure 2.13: Slider Control with Variable handled and its control limits

of the input signals will be obtained after the **In-Gain** block, while the real values of the output signals will be obtained before the **Out-Gain** block.

– Click on the Data Acquisition tab, in the Instrument toolbar at the right.
  – Select the Plotter icon and draw a larger rectangle in the Layout window.

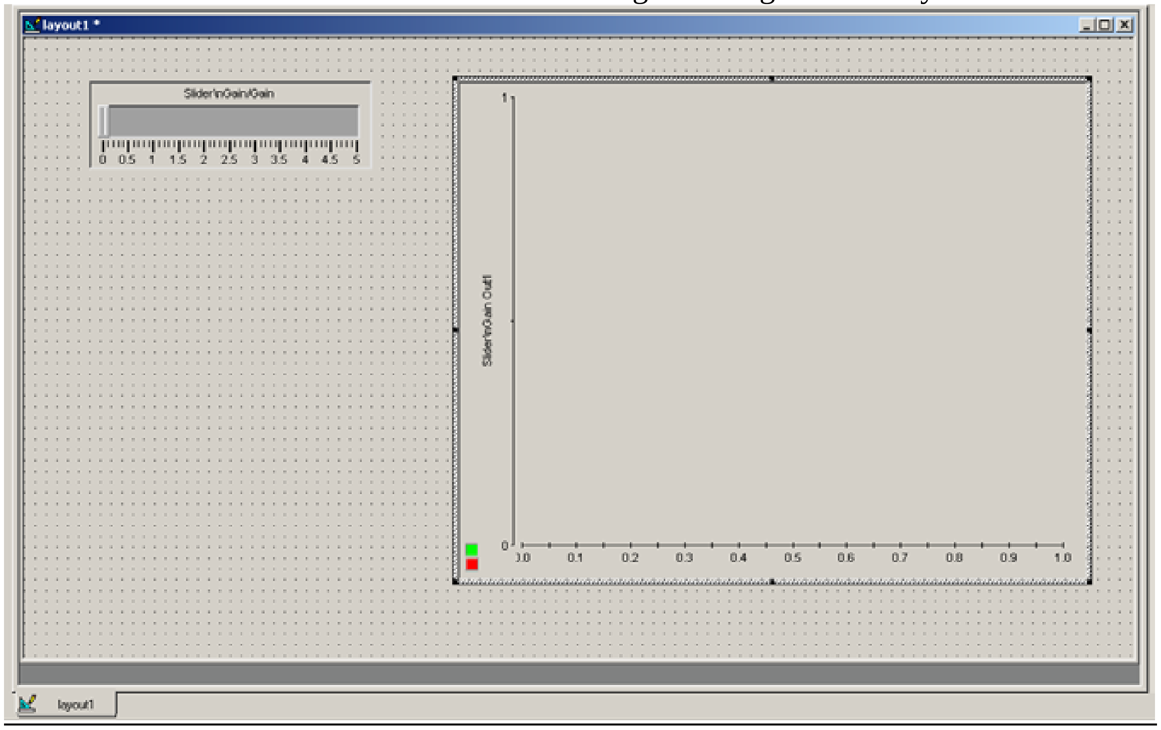
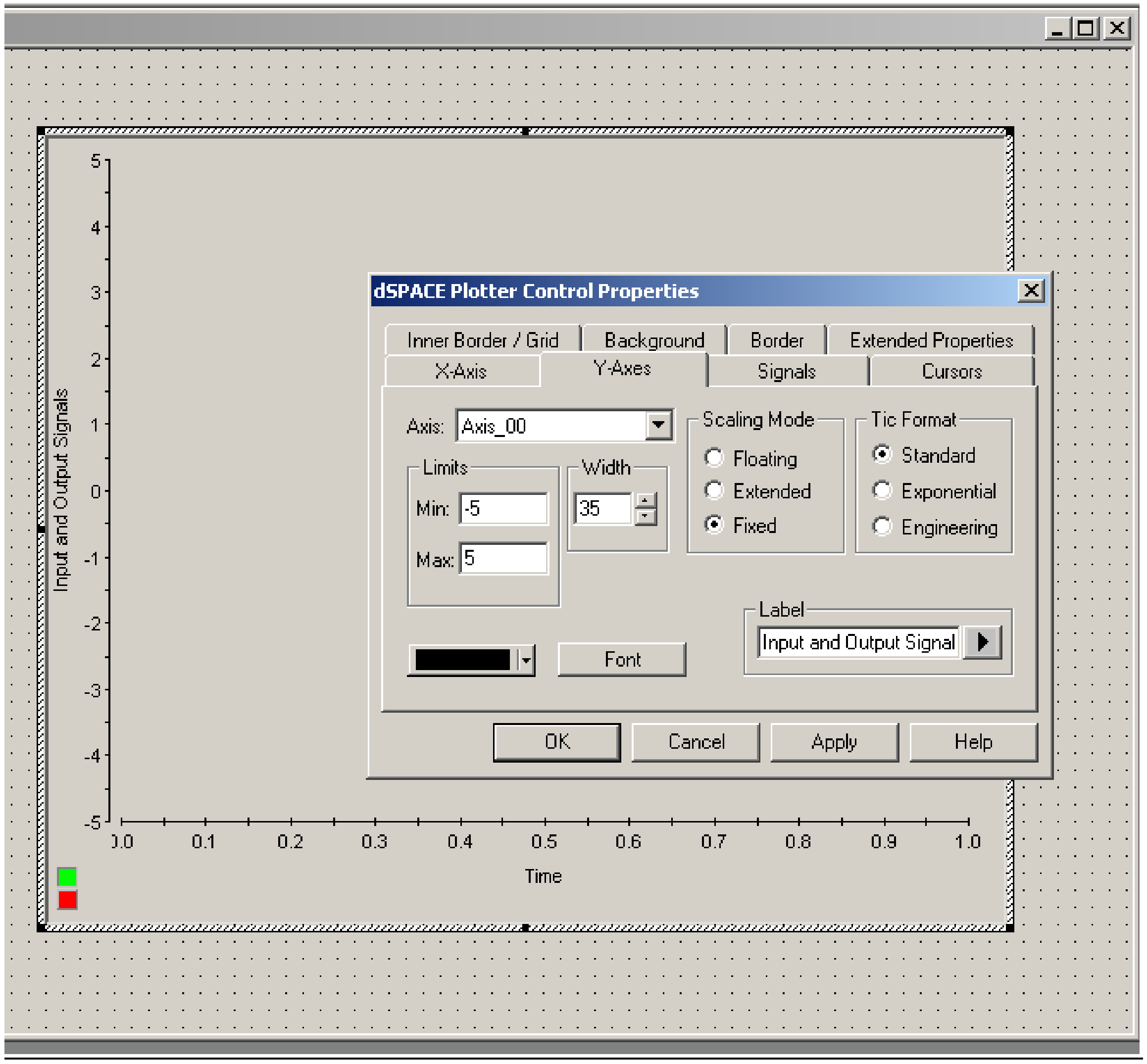


Figure 2.14: Drawing a Plotter for Signal Monitoring

1. Drag the signals which you need to monitor, in the new plotter: **'Model Root/In-Gain->Out1' and 'Model Root/Slider Gain->Out1'**.
2. When dragging the second signal, make sure that you release the mouse button above the first one, on the vertical axis. Otherwise, a new vertical axis will be drawn, and you will have less available space for visualize the waveforms. Now both signals are assigned to the plotter and will be displayed with different colors. The label on the vertical axis will show only the last signal dragged to the scope. You should have your layout as shown in Fig. 2.14
3. Any time you wish to make a correction, or see what signal was added to the plot, right-click in the scope area and select **Edit Data Connections** command. You can delete any signal by selecting it and press Delete key. You can even delete or modify your signals or changing the colour of the signals, by double clicking Y-axis and going in to the "*Signals*".
4. To format the scope, double-click in the rectangle and select **Y-axis** tab. When the signals are displayed on the same y-axis, the graph settings will be identical for both. Otherwise, you will be able to select different y-axis limits for each of the signals, independently.
5. Set Y-limits, Width, Scaling Mode as shown in Fig. 2.15
6. At the X-label you can write "Time" and at Y-label write "Input and output signals".

7. One more step, before you actually can start the simulation: setting the visualization parameters. The process parameters can be set in the **Capture Settings Window** under View/Controlbars menu.
8. Open the Capture Settings Window and set the Length of the simulation as 2, while leaving unchanged the Downsampling number. Your capture settings should look like as shown in Fig. 2.16. For more complex systems this number has to be increased when the Length is larger than 20 times the sampling time.

Now, we are ready to start the simulation.



Axis: Axis_00

Limits: Min: -5, Max: 5

Width: 35

Scaling Mode: ( ) Floating ( ) Extended [✓ Fixed]

Tic Format: [✓ Standard] ( ) Exponential ( ) Engineering

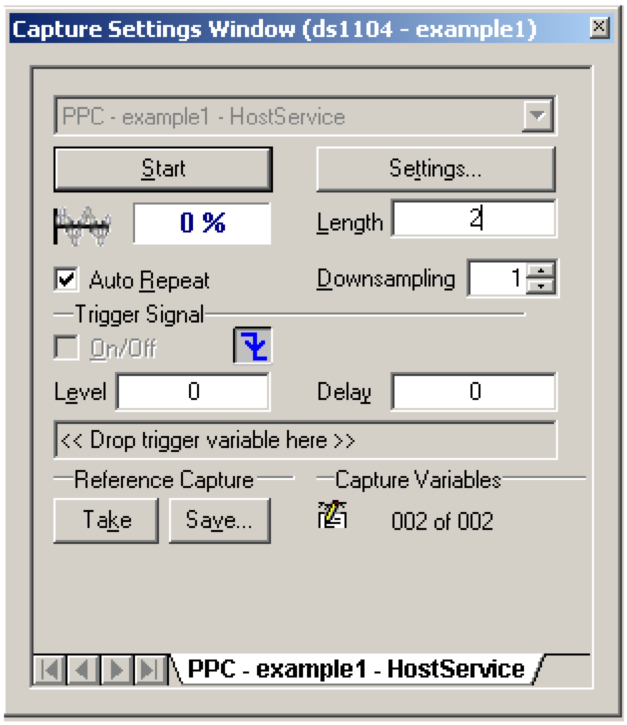
Label: Input and Output Signal

Figure 2.15: Plotter Setting Window

2.6 Running the Experiment

There are two operations that have to be done to run/stop the experiment. First, the DSP execution has to be started. Second, the animation and data acquisition/printing needs to be initiated. When stopping an experiment, the operations have to be done in the reverse order.

To start and stop the DSP you can use the icons in the Platform Management toolbar (see Fig. 2.17.
Edit mode toolbar contains edit, test or animation icons. Refer Fig. 2.17 for edit, test and animation tool icons.
Edit mode is used to edit or modify your layout, start and stop your real-time simulation. Once you start the real-time simulation, click the animation icon, this will let you modify parameters in real-time and observe the waveform. Unless real-time simulation starts, you cannot go into this mode. Clicking on Test icon lets you go into the Test mode. In this mode, you can observe the snap-shot of your real-time simulation.


Capture Settings Window (ds1104 - example1)

PPC - example1 - HostService

Start | Settings...

0 % | Length 2

✓ Auto Repeat | Downsampling 1

Trigger Signal

On/Off

Level 0 | Delay 0

<< Drop trigger variable here >>

Reference Capture: Take | Save...

Capture Variables: 002 of 002

PPC - example1 - HostService

Figure 2.16: Capture Setting Window

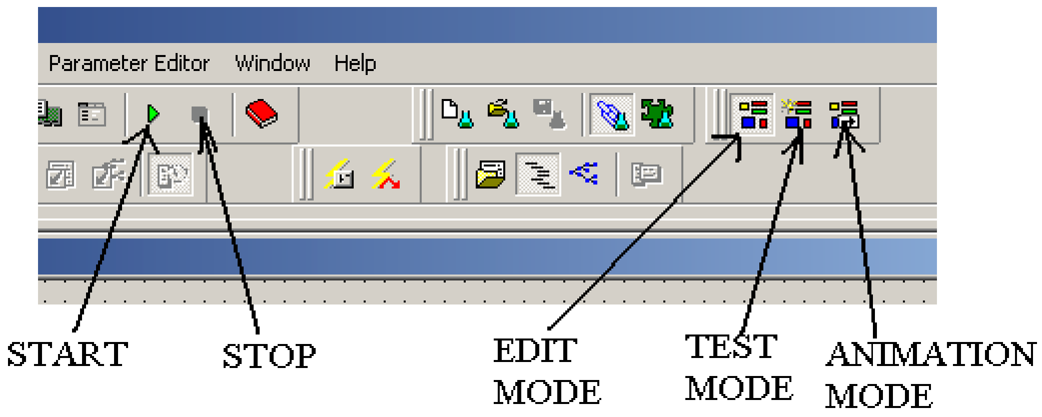


Figure 2.17: Execution and Animation Control Toolbar

1. Start the DSP by pressing the green triangle button. If the example1.obj is loaded in the DSP memory then it will start running and the Stop, red-rectangle shaped button will become active.

2. Start the Animation by pressing the animation icon in the Edit Mode toolbar.

3. You will see both the signal generator and the system output signals on the plotter. Every two seconds the scope will clear and a new set of data are displayed.

4. Change any of the parameters and watch the modifications in the signals displayed on the scope. Also observe the waveforms in the plotter window and they should like shown in Fig. 2.18 for the slider gain value displayed in Fig. 2.18. Now the experiment is running and our design job is done.

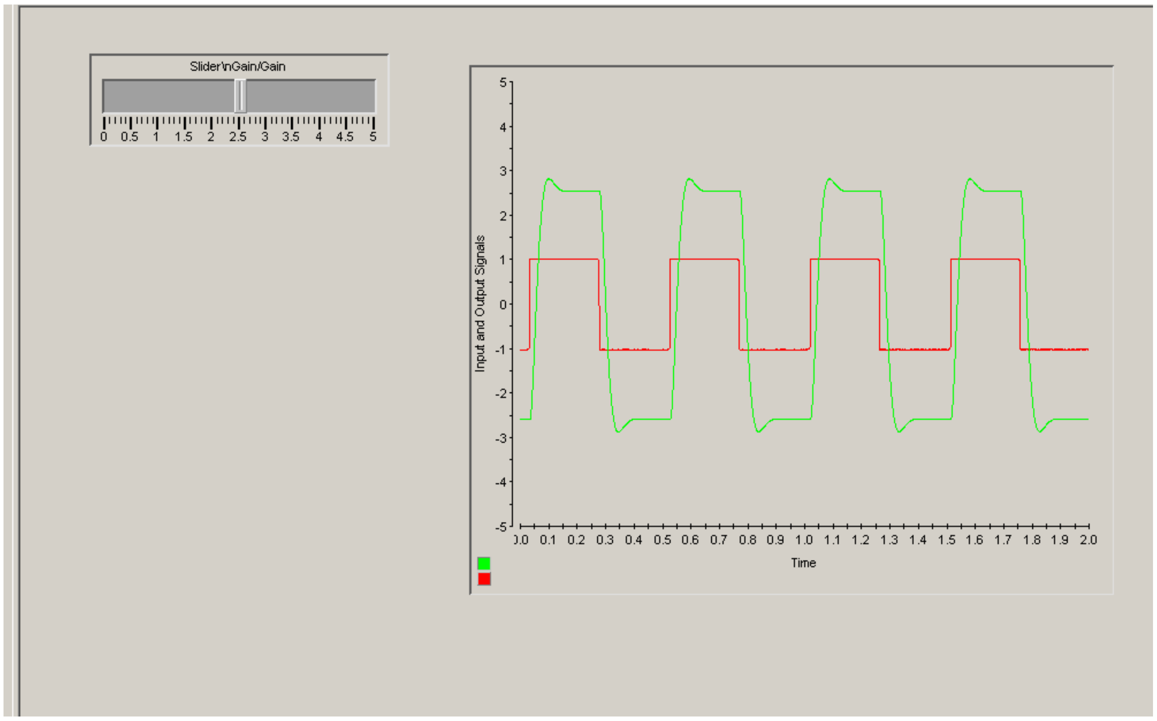
5. For saving an experiment, you must follow these steps, otherwise you might loose some precious design in the layout, or some simulation settings.

   – Click File/Add All Opened Files. This implies that the experiment will remember the *.sdf file, containing the variables to handle, will know the path for all other files and will open the connections between the layout and the variables.

   – Save the Layout in a file with the *.lay extension.

   – Save experiment in a *.cdx type file.

For loading an experiment, simply click File/Open Experiment and, if saved as previously instructed, the layout and the variables will appear on the screen.


Slider\nGain/Gain
0 0.5 1 1.5 2 2.5 3 3.5 4 4.5 5
Input and Output Signals
5 4 3 2 1 0 -1 -2 -3 -4 -5
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0
Time
🟩
🟥