Victor Garcia
Jessica Jiang

Lab 1 Report

1. Introduction

This lab served as an introduction to simulink and dSPACE. Simulink is a module tool that allows the user to program the depicted board in Figure 1, without any coding. Meanwhile, dSPACE is the controller that will take in the Simulink program and through that, control the motor it is attached to. Throughout this course, one will use this board for rapid prototyping with motors and other hardware to analyze data and system behavior. In this lab, a simple block module made in Simulink was used to measure the speed and velocity of the motor at any given time. The user would be using the dSPACE platform to display and graph the recorded values.

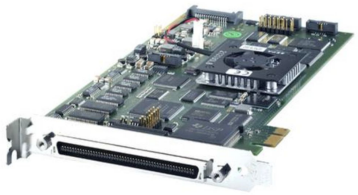


Figure 1: dSPACE DS1104 digital controller

2. Design Procedure

2.1 Hardware Configuration

Before one proceeded in providing the DS1104 R&D Controller Board with the instruction-set in simulink, one needed to set varying connections for the loadless motor to work. For this lab, one learned that the board is mounted onto the PC through PCI slot connectors, granting speedy response time since it is connected directly through the motherboard. Due to this, the motherboard is also capable of flashing the instructions from Simulink to the target. Figure 2 demonstrates how the digital controller is attached to the bottom-most PCI slot.

Figure 2: As one may see, the digital controller is mounted on the desktop. One must note that the software key is validated by physically applied by connecting a USB to the PC.

The DS1104 R&D board is then connected to the power electronics board, which is used to connect other hardware; in this case the team connected one motor to it. The remaining connections for the motor are to to the isolated DC power supply. The connections can be interpreted into a block diagram depicted below:

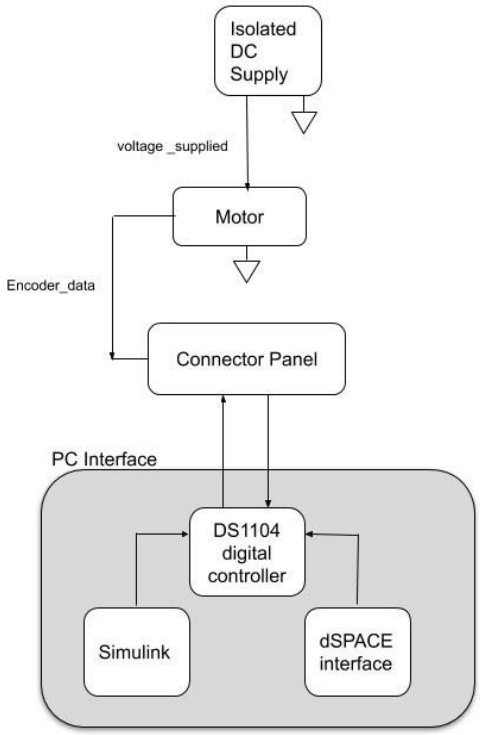


Figure 3: Block Diagram

2.2 Simulink

To program the board, setup and save a new blank model in Simulink. Find the dSPACE controller model in *Simulink Library > dSPACE RT1104 > DS1104 MASTER PPC > DS1104ENC_POS_C1* and drag it into the window. From there, use the blocks to convert the raw data into data with standard SI units. The goal of the lab is to convert the raw data into position and velocity graphs.

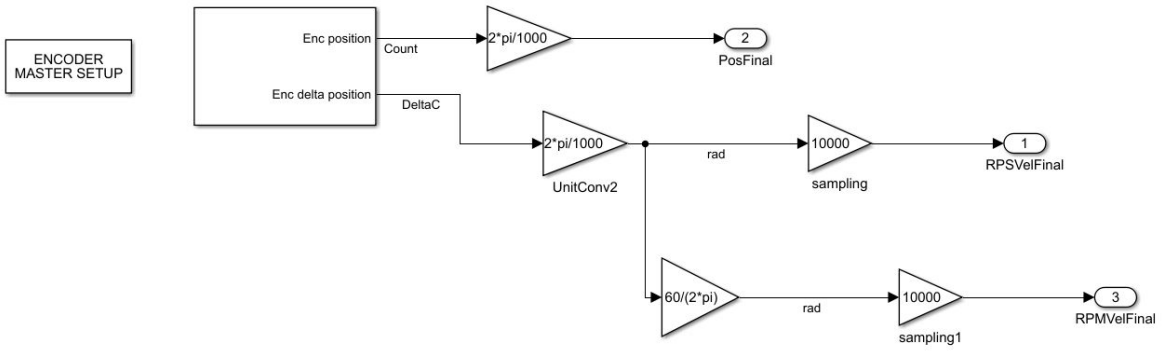


Figure 4: Model Diagram created from modeling blocks in Simulink

Be sure to label all nodes and wires so that the could be referenced and found in dSPACE's platform.

To ensure that the variables inside the diagram could be used outside the program, the user needs to go to the output wire and *right click > properties > code generation > storage class > Simulink Global*. This allows the data from the wire to be used globally. After this step, the code is ready to be generated. Go to  *> Solver > Solver Options Type:Fixed-Step > ode4* and  *> Solver > Additional Parameters > gain sampling* to fix the gain sampling for the data. Lastly,  *> Code Generation > Target selection > Browse > rti1104.tlc* will output the data to the controller.

2.3 Common Simulink Errors

Block reduction error:  *> Optimization > "..." > un-tick block reduction*

Master Encoder error: Add an *Encoder Master Setup* block from the *DS1104 MASTER PPC* library.

2.4 dSPACE

Once the code has been generated, open up the dSPACE controller platform and open up a new project. Make sure to select the same directory as the Simulink file and link the proper ".sdf"

simulink file after finding the correct device(DS1104). From there, go to *Instrument selection > time plotter* to drag and drop an empty graph into the windows. After the graphs have been dragged to their desired positions, find the correct variables that were named as shown in *Figure 4* and drag it out to the graph. If desired, the graphs can be labeled with *Instrument selection > Static Text > Change Value*.

Once everything has been set up in dSPACE, the motor can be powered on to see the results graphed onto time plotter. Use the cursor to see the exact values on the graph at the time taken. Once cursor can be used to navigate both graphs as seen in the analysis.

3. Analysis

3.1 Interpreting Raw Data to Position

The Encoder position will have raw data presented in counts. When converting, it is important to note that 1000 counts refers to one cycle. Meaning that 1000 occurs at $2\pi\ rad$; here we interpreted a full revolution to be in terms of radians. To convert counts into radians, the following formula would have to be applied:

$$1000\ count\ =\ 2\pi\ rad\ =\ 360^{\circ} \qquad \text{Eqn. 1}$$

The conversion can be viewed as the following $x\ counts\ (\frac{2\pi\ rad}{1000\ counts})$, where the units "counts" cancels. This team chose to select the gain symbol because the encoder position pin is simply multiplied with the ratio $(\frac{2\pi\ rad}{1000\ counts})$. It is important to note that this is a linear encoder, meaning that as the motor spins, the position in the linear direction is calculated. The graph for this would show that it keeps growing while the motor spins.

3.2 Velocity in Radians per Second ($\frac{rad}{sec}$)

To calculate the velocity of the motor, the "enc delta position" pin is used, along with the following formula to convert the change in position into velocity in radians per second

$$\omega = \frac{\Delta\theta}{T_S} = \frac{1}{T_S}\left(\frac{2\pi\ rad}{count}\right) \qquad \text{Eqn. 2}$$

Here, the data output by "enc delta position" is read as $\Delta\theta$. To get the desired units, Simulink reflects that the first step is to convert to radians as before (Eqn. 1); this takes place in the first gain symbol. To obtain the velocity, a sampling period was chosen to give us units of velocity, in this case radians, per second. Here we chose the sampling period to be 0.0001, where $\frac{1}{T_S} = 10,000$. This value is reflected by the second gain operator named "sampling".

3.3 Velocity in Revolutions per Minute (RPM)

Using the previous formula (Eqn 2), the answer is then inserted into the following formula to convert radians per second to revolutions per minute

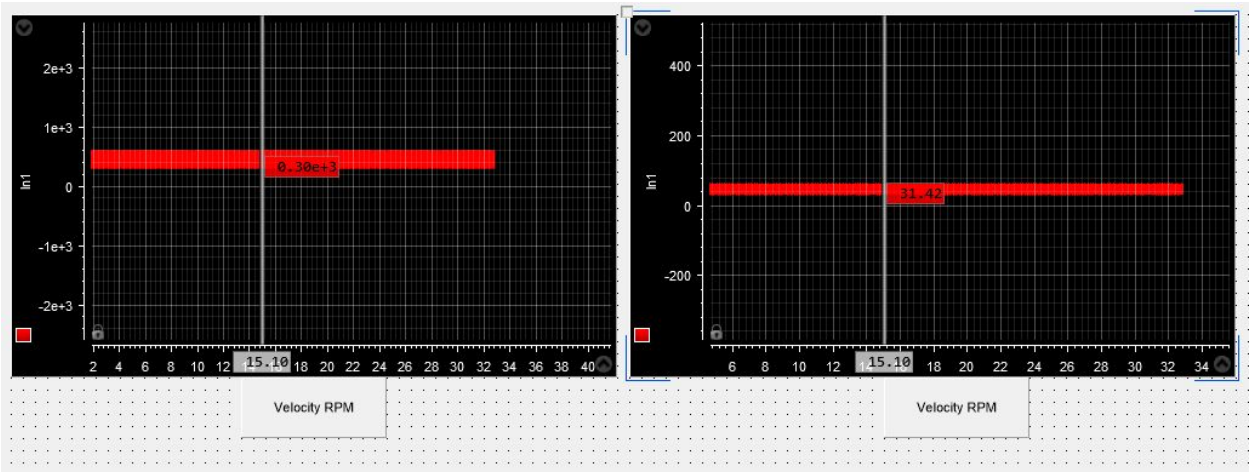
$$\omega = \frac{rad}{s} * \frac{1\ rev}{2\pi\ rad} * \frac{60\ sec}{1\ min} \quad \text{Eqn. 3}$$

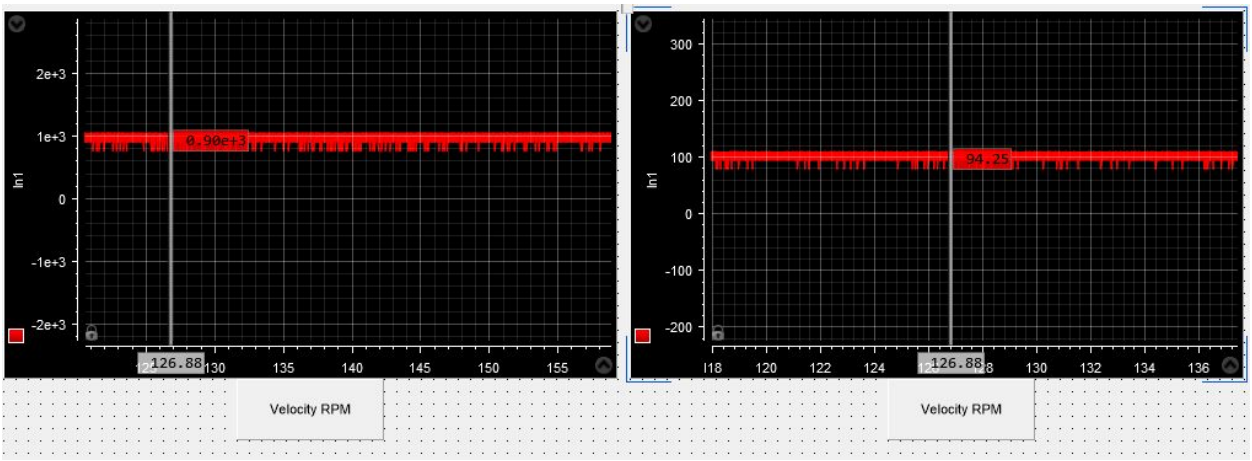
Since this operations also required data from "enc delta position", this team chose to branch off from the output of gain symbol named UnitConv2. After doing this, the operations used in the equations above are handled by a gain symbol cascaded with another gain symbol handling the sampling period. Here, the first gain block reflects the arithmetic performed in Eqn. 3.

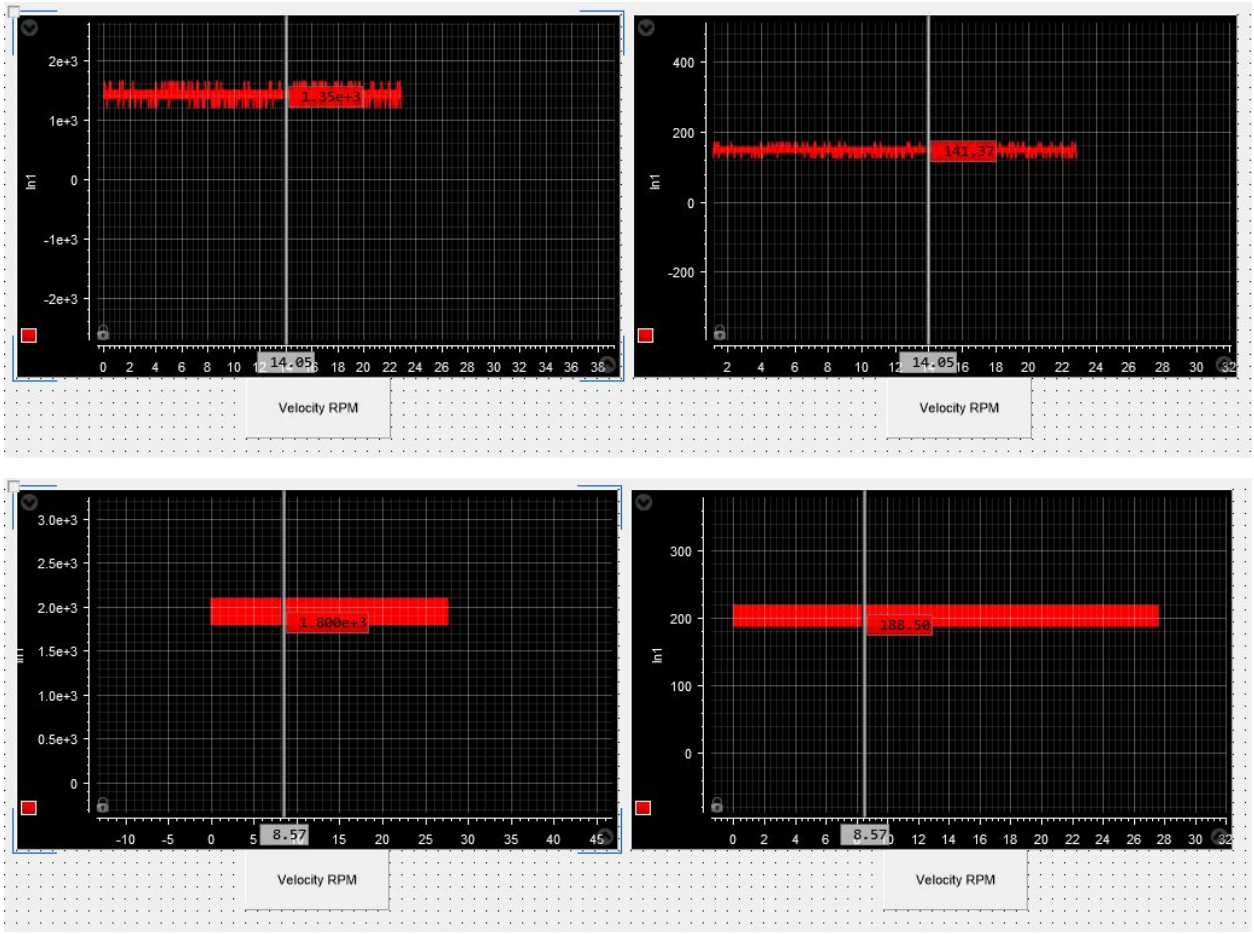
4. Simulation Results

Voltage (V)	Velocity (RPS)	Velocity (RPM)
5	0.30×10^3	31.42
10	0.90×10^3	94.25
15	1.35×10^3	141.37
20	1.80×10^3	188.5

Note: There is a problem with the labels on the left graph on each of the screen shots. Velocity RPM should be Velocity RPS.


2e+3
1e+3
0
-1e+3
-2e+3
In1
0.30e+3
15.10
Velocity RPM
400
200
0
-200
31.42
15.10
Velocity RPM


2e+3
1e+3
0
-1e+3
-2e+3
In1
0.90e+3
126.88
Velocity RPM
300
200
100
0
-100
-200
In1
94.25
126.88
Velocity RPM



The data represented above depicts a relationship between the voltage and the motor velocity. As the voltage increases, the velocity of the motor increases as well. A notable issues encountered when performing these trials was the scaling of the graphs and the noise read; almost like static. Since there was plenty of static, it was hard to get a nice looking graph. Although this was the case, the cursor tool allowed one to get a measured value for a certain point in time.

5. Conclusion

This lab was a great opportunity for one to ease into board configuration and data analysis. Although basic knowledge of Simulink is useful to finish things quickly, the user-friendly interface makes it easy for users to implement their designs. Since this was the first time one used Simulink with this board, a few issues were encountered along the way. Once of the issues this team encountered was for the real-time position representation of the motor. For some strange reason, the position would stay steadily flat rather than steadily increasing. Another issue was in regards to the data read; there seemed to be plenty of noise present. To get desired values, one had to scale axis in such a way that the data was readable through a cursor. The goal for

future labs would be to reduce this noise by implementing some sort of filter to read more precise data.