DC Motor Control[*]

Jacob Fainguelernt

Abstract

This module demonstrates the use of the c28x peripherals and DMC library blocks to control the speed of a DC motor in a closed-loop fashion. This example is based on the "DC Motor Speed Control via RTDX" SIMULINK demo.

1 Introduction

This chapter demonstrates the use of the c28x peripherals and DMC library blocks to control the speed of a DC motor in a closed-loop fashion. This example is based on the "DC Motor Speed Control via RTDX" SIMULINK demo.

The target speed of the motor is set by the user in the MATLAB GUI. This value is fed to the Controller (based on the eZDSP-F2812) to change the motor speed. The loop is closed by a tachometer. The controller constantly adjusts the value of the DC voltage applied to the motor to maintain the desired speed. The control loop is shown in the following figure:

[*]Version 1.1: Apr 20, 2009 11:26 am GMT-5
[†]http://creativecommons.org/licenses/by/2.0/

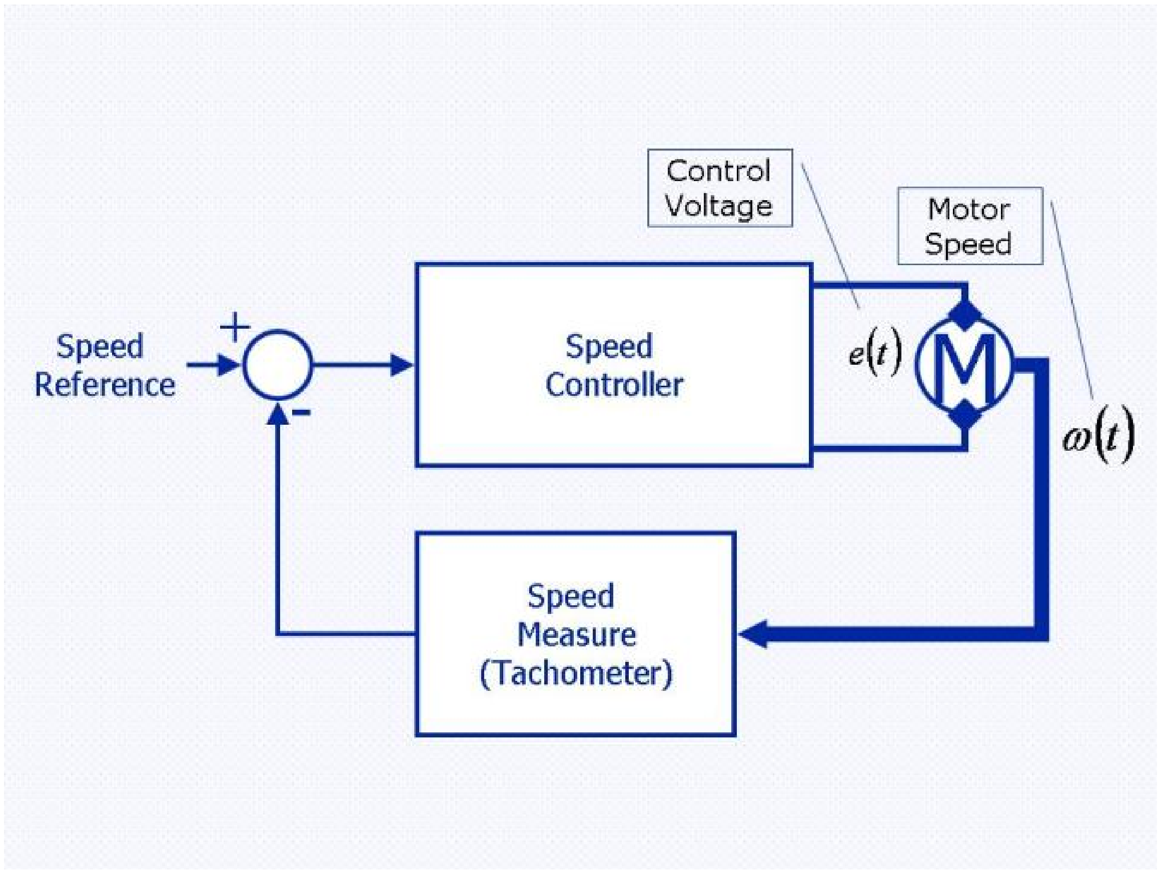


Figure 1: DC Motor Control Loop

The Speed controller comprises two blocks (please refer to Figure 2):

1. This block compares the desired speed with the measured speed and generates a digital value proportional to the DC value to be applied to the motor.

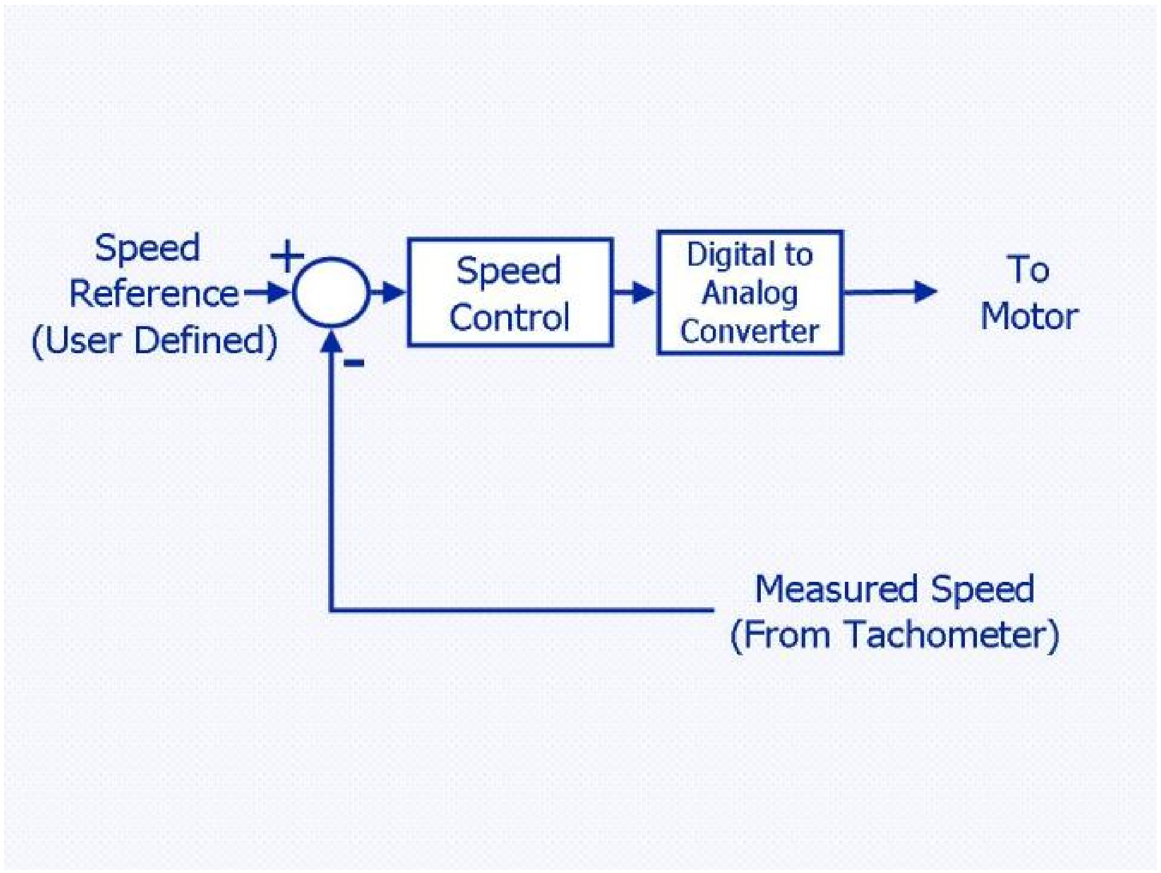

Speed Reference (User Defined)
+
-
Speed Control
Digital to Analog Converter
To Motor
Measured Speed (From Tachometer)

Figure 2: The Speed Controller

2. This block is implemented using a PWM signal generator and a Low Pass Filter as shown in Figure 3. This method is described in . The output voltage, generated at the LPF output will be:

$$V = D \cdot V_{\text{ss}} \qquad (1)$$

Where:

V	- Generated Voltage
D	- PWM Duty Cycle
V_{ss}	- Supply Voltage (3.3 V)

Table 1

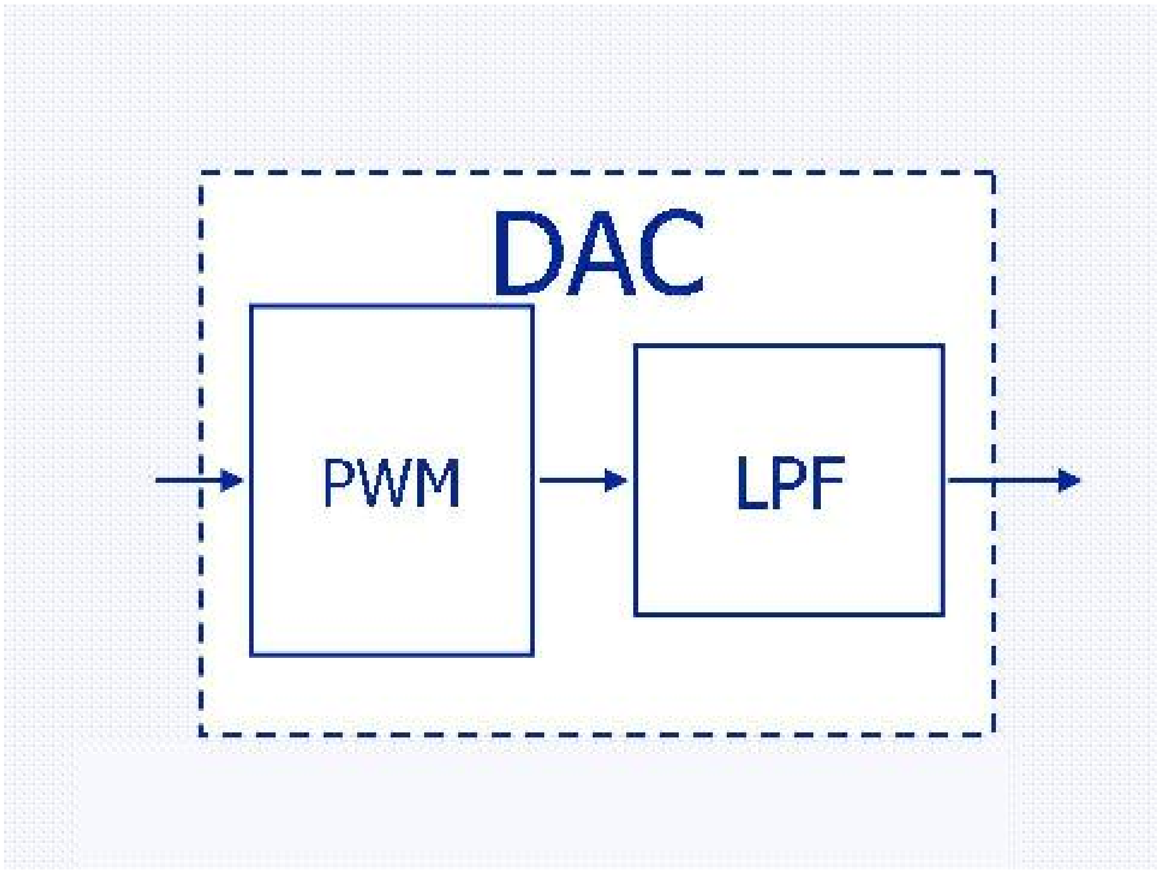


Figure 3: Digital to Analog Converter Implementation

Figure 4 shows the loop control implementation, based on three components:

1. eZDSP-F2812 (Please refer to)
2. Interface circuit (Please refer to Section)
3. DC Motor Kit (Please refer to Section)

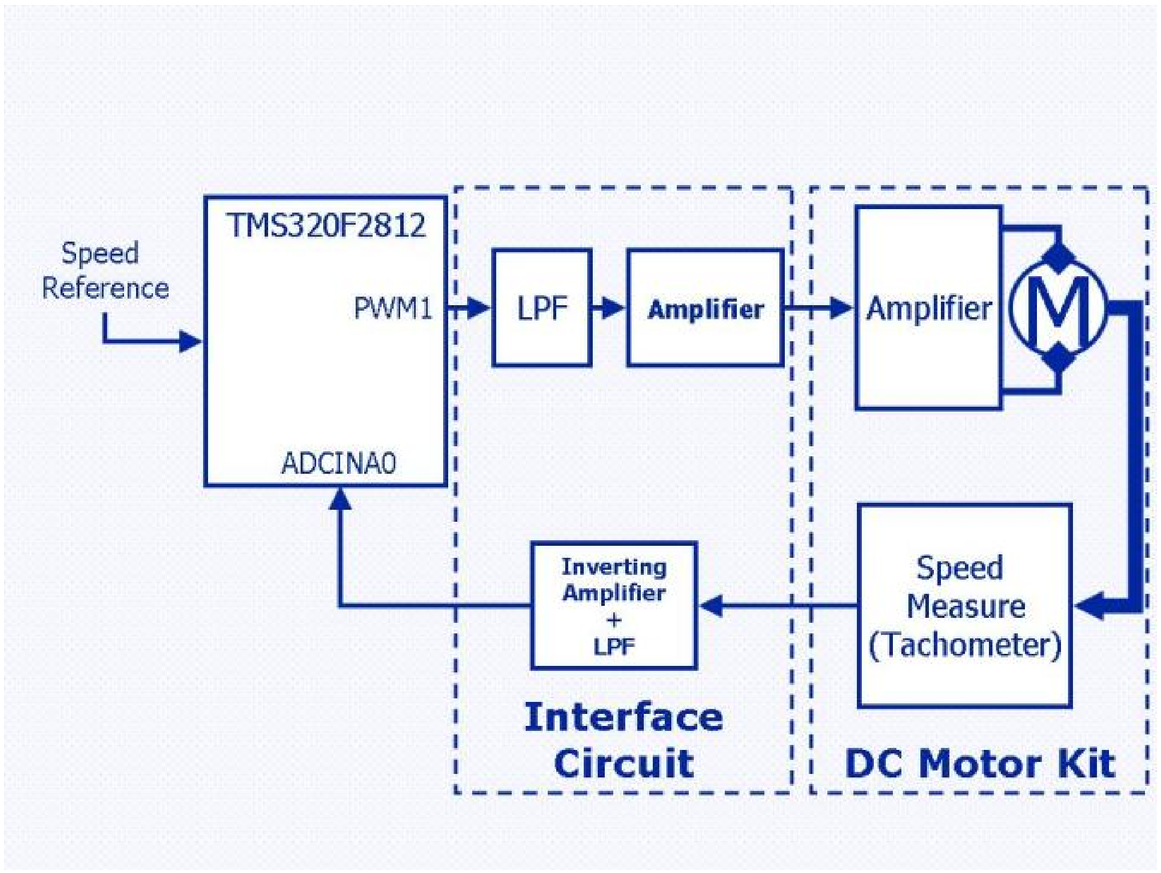


Figure 4: DC Motor Control Loop Implementation

1.1 Setup

This demo is based on the Spectrum Digital eZdsp-F2812 that is connected to a DC Motor Kit through a dedicated interface circuit. The setup is shown in Figure 5.

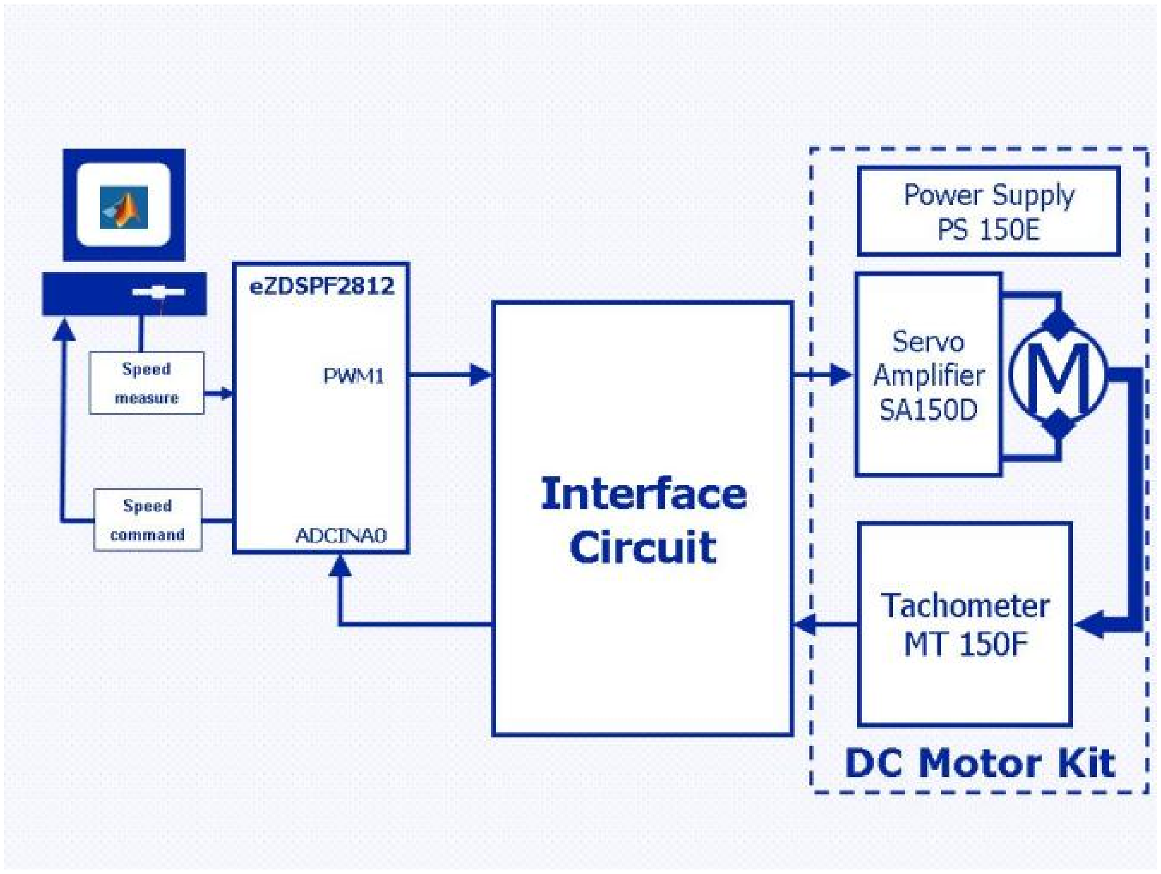


Figure 5: Setup

1.1.1 The DC Motor Kit

Servo amplifier [SA150D]:
This unit operates the motor from signals applied to the input sockets 1 or 2, enabling control of the motor speed and reversing its rotation. This example shows speed control only, and will make use input socket 1 only. The servo amplifier is connected to the servo motor by an 8-pin plug and able. Terminals for ± 15 volts and ground (common) are available on this unit.
Servo motor with Tachometer [MT150F]:
This unit is a DC motor that produces a torque of the order of 8 oz-in (600 gm-cm) at 2A input current. The inertia is about 3x10−5 Kg −m2. The output shaft may be fitted with a brake disc and/or an inertia disc to load the motor. A second shaft on the side of the motor is coupled to the main shaft by 30:1 gears (the smaller shaft rotates slower than the main shaft). The tachometer with terminals +, - and common (ground) is attached to the motor.
Power supply [PS150E]:
This unit provides the various voltages supplies required for the servo components. There are terminals for ±15 volts, and common (ground). An ammeter is also included. The maximum current is 2 A. An 8-pin

socket and cable connects this unit to the servo amplifier.

1.1.2 I/F Circuit

This circuit was designed as part of the demo that will be described in detail in chapter .

1.2 Measurement Equipment

1. Signal Generator
2. Voltmeter
3. Oscilloscope.
4. Stroboscope

1.3 Related Files

- Simulink Model for Simulation - DCMotorControl.mdl[1]
- MATLAB script for Real-Time - speedControlDCLoop.m[2]
- Simulink Model for Real-Time - DCMotorControlc2812.mdl[3]

2 System Identification

The transfer function of a DC motor can be approximated by a first order model with unknown constants. These constants can be identified experimentally.

The tachometer provides the feedback signal for speed control systems. A schematic diagram of the tachometer is given in Figure 6.

[1]See the file at <http://cnx.org/content/m22189/latest/DCMotorControl.mdl>
[2]See the file at <http://cnx.org/content/m22189/latest/speedControlDCLoop.m>
[3]See the file at <http://cnx.org/content/m22189/latest/DCMotorControlc2812.mdl>

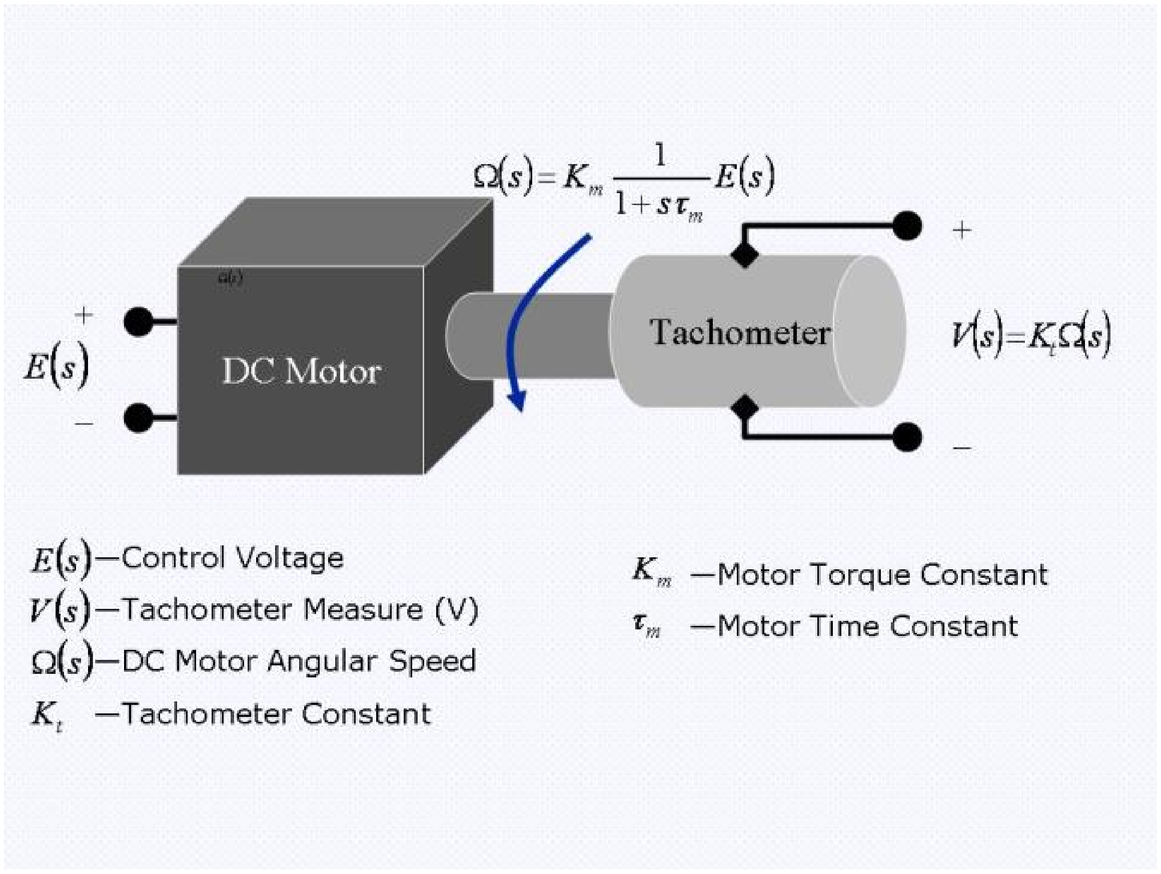

$\Omega(s)=K_m\frac{1}{1+s\tau_m}E(s)$
$\omega(t)$
+
$E(s)$
−
DC Motor
Tachometer
+
$V(s)=K_t\Omega(s)$
−
$E(s)$—Control Voltage
$V(s)$—Tachometer Measure (V)
$\Omega(s)$—DC Motor Angular Speed
K_t —Tachometer Constant
K_m —Motor Torque Constant
τ_m —Motor Time Constant

Figure 6: DC Motor with Tachometer

In this section we will identify the constants in the mathematical models of a DC motor and the tachometer experimentally. The process consists of consists of two parts:

1. Measurement of the tachometer coefficient and the motor constant
2. Measurement of the motor time constant.

Measurement Equipment

- Signal Generator
- Voltmeter
- Oscilloscope.
- Stroboscope

2.1 Measurement of Tachometer Coefficient and Motor Constant

The tachometer coefficient and motor constant can be measured by generating a control voltage, using the PWM block of the DSP, and measuring:

1. The motor speed using a stroboscope
2. The tachometer output using a voltmeter

The setup is shown in Figure 7.

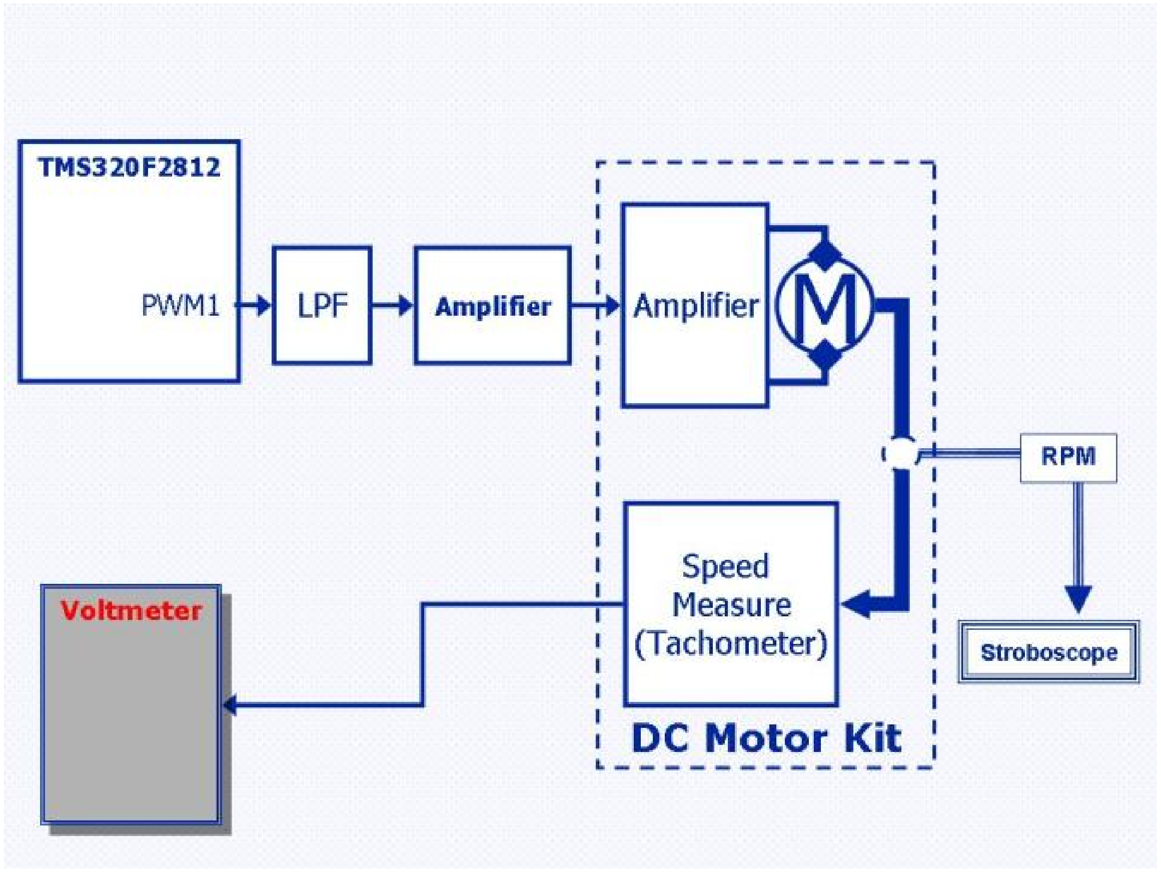


Figure 7: Calibration of the Motor and Tachometer

Figure 8 shows the model that will be used to generate the control voltage.

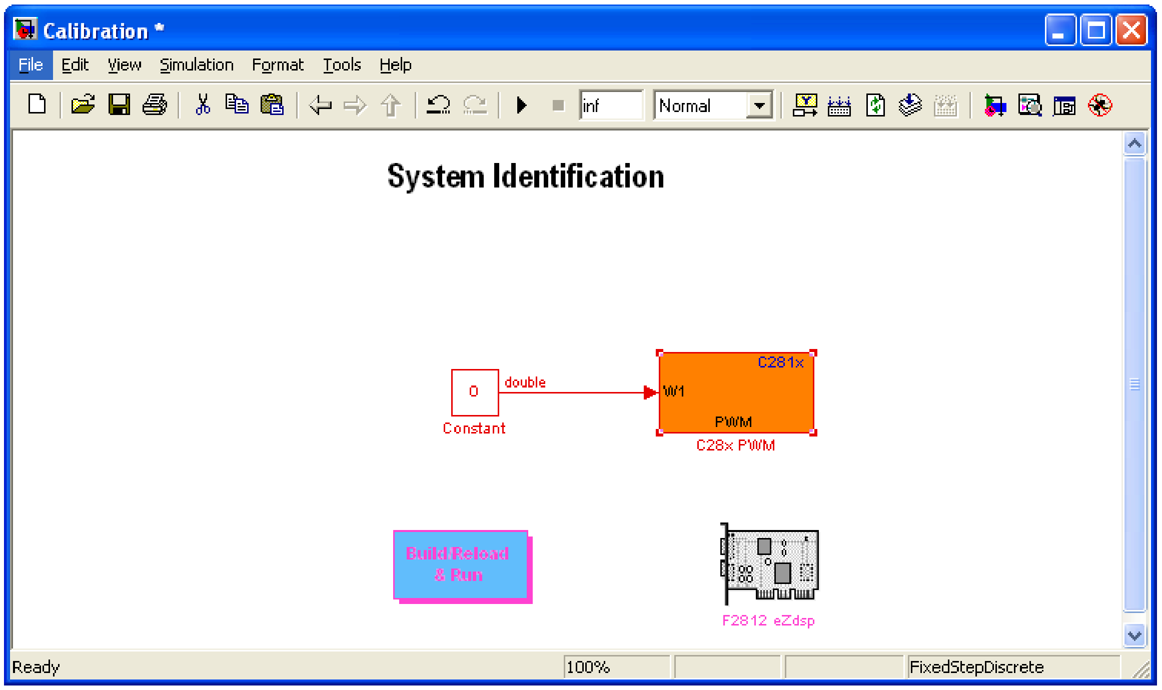
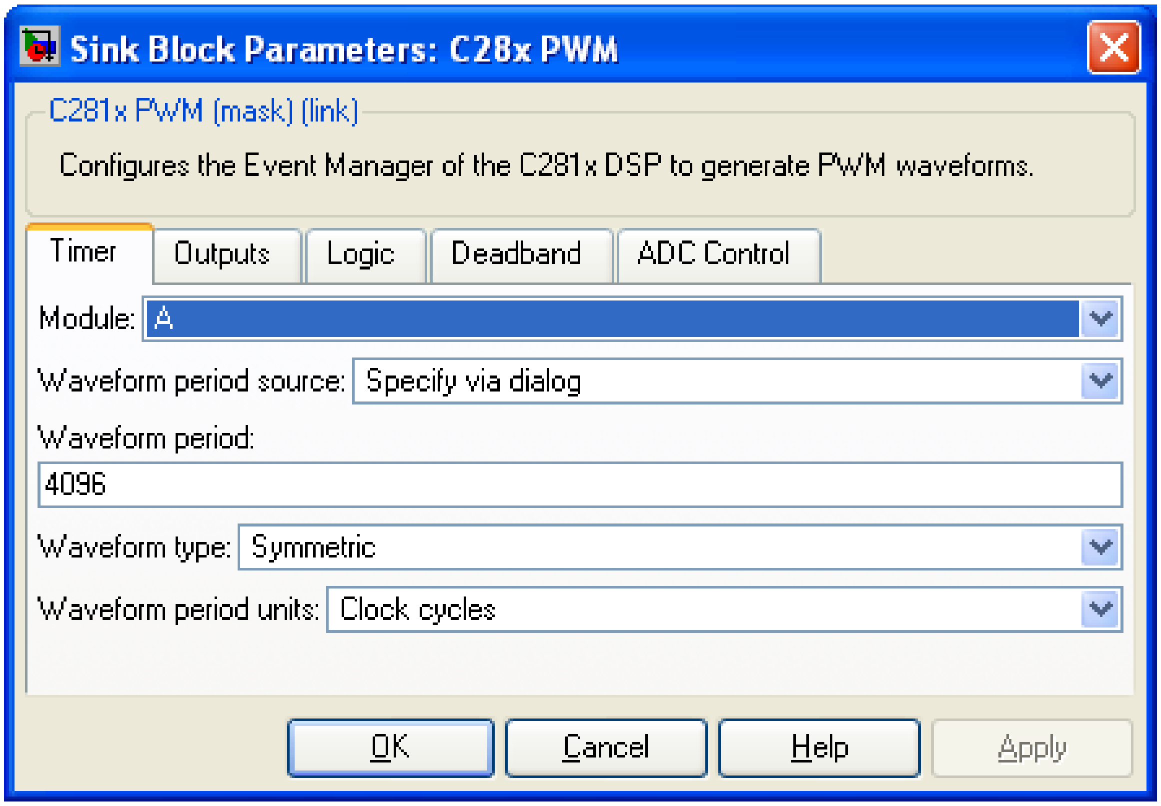


Figure 8: System Identification Model

The PWM block will be configured to generate a PWM signal with a period of 4096 clock cycles of 75 MHz (please refer to Figure 9).


Sink Block Parameters: C28x PWM

C281x PWM (mask) (link)

Configures the Event Manager of the C281x DSP to generate PWM waveforms.

Module: A

Waveform period source: Specify via dialog

Waveform period:
4096

Waveform type: Symmetric

Waveform period units: Clock cycles

Figure 9: PWM Configuration

Measurement Procedure:

1. Connect the circuit shown in Figure 7.
2. Generate a 0 V input to the DC Motor Kit (Compile Build and Run the model with Duty Cycle=0)
3. Measure the motor speed using the stroboscope and the tachometer voltage using a voltmeter.
4. Repeat step 3 for input voltages in the range 0-15 V. Change the Duty Cycle from 0 to 100% in steps of 10 %.
5. Obtain a plot of tachometer voltage vs. servo amplifier input voltage.

The ADC input voltage of the DSP should be in the range 0-3V, the therefore the tachometer output voltage must be inverted and scaled down. This is done with the circuit shown in Figure 19. The gain of this circuit is G = - 100/270 = - 0.37, i.e., inverting and attenuating approximately 1/3. The results obtained are shown in Figure 10.

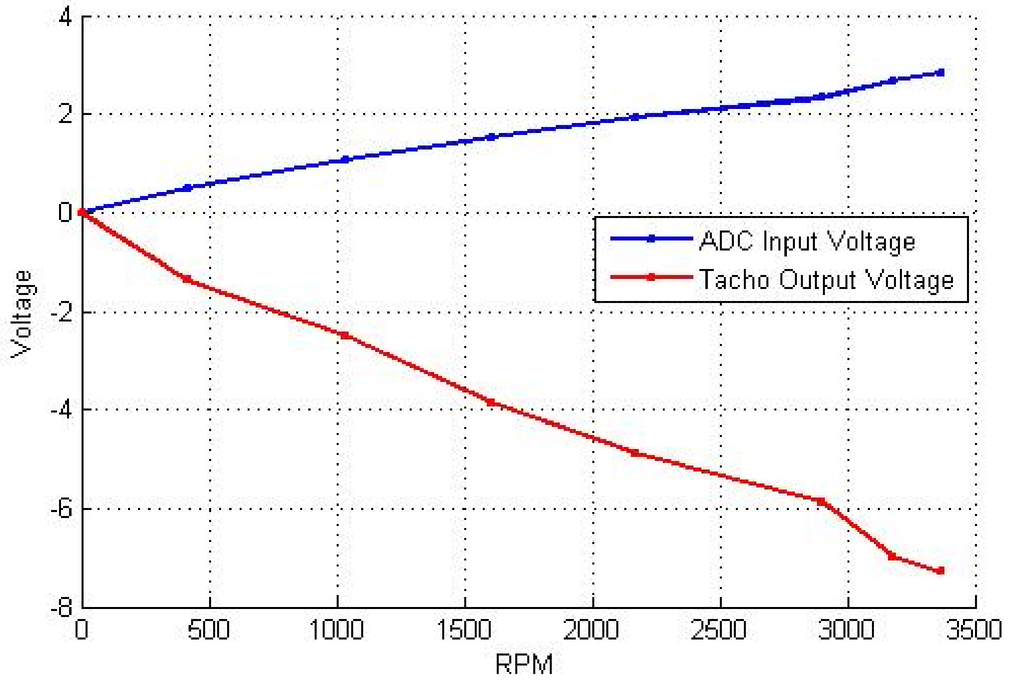


Figure 10: Speed Measurement

2.2 Measurement of the motor time constant

The measurement of the time constant will be obtained by generating a square pulse and measuring the motor response to this stimulus. The principle of operation is shown in Figure 11; the experimental setup is shown in Figure 12.

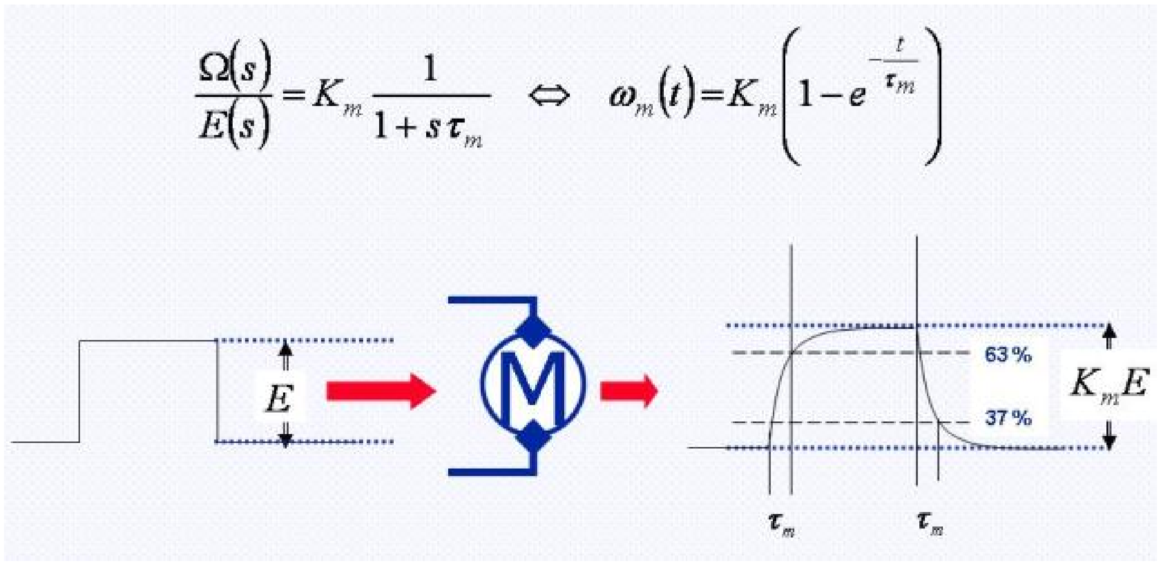

$\frac{\Omega(s)}{E(s)} = K_m \frac{1}{1+s\tau_m} \Leftrightarrow \omega_m(t) = K_m\left(1-e^{-\frac{t}{\tau_m}}\right)$
E
63 %
37 %
$K_m E$
τ_m
τ_m

Figure 11: Measurement of the motor time constant

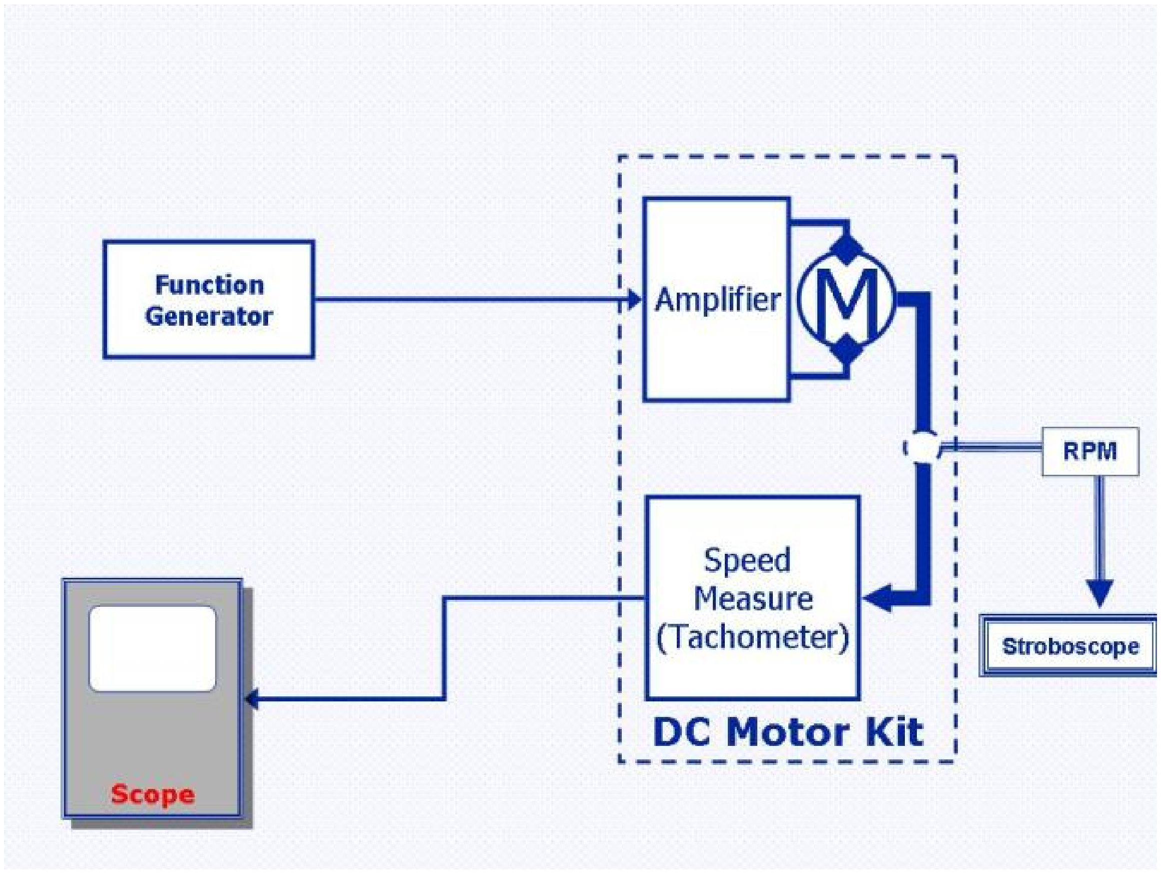


Figure 12: Measurement of the motor time constantMeasurement

The Function Generator generates pulses, as shown in Figure 13.

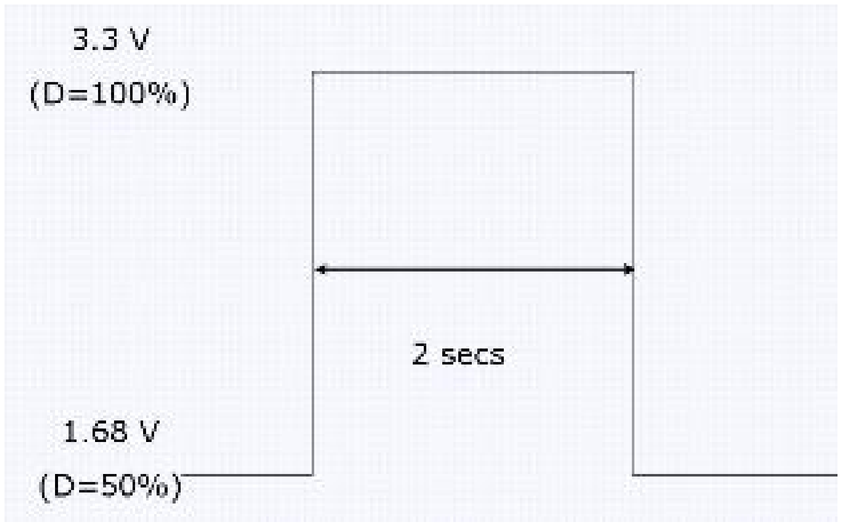


Figure 13: Pulse Parameters

Measurement Procedure

1. Connect the circuit shown in Figure 12.
2. Generate the pulse shown in Figure 13.
3. Measure the rising and falling times of the motor response.

The DC motor was modeled as a first-order linear system where $\tau_{\text{rise}} = \tau_{\text{fall}}$; however, the actual motor is not linear (due to friction, for example), therefore the measured values were:

τ_{rise}	$= .13$ secs
τ_{fall}	$= .23$ secs

Table 2

We used the average of these readings, i.e., $(0.13+0.23)/2 = 0.18$ and rounded to 0.2 sec.

$$\frac{\Omega(s)}{E(s)} = K_m \frac{1}{1+s\tau_m} \quad \Leftrightarrow \quad \frac{\Omega(s)}{E(s)} = K_m \frac{1}{1+.2s} \qquad (2)$$

3 Real-Time System Design

The system design comprises two parts:

1. Simulation, to determine the parameters of the Speed Control Transfer Function
2. Hardware Design of the interface circuit.

3.1 Simulation

The speed control is a PID controller. A DC motor model (Figure 14) was created. The complete control loop model is shown in Figure 15. Figure 16 shows the parameters chosen for the PID controller.

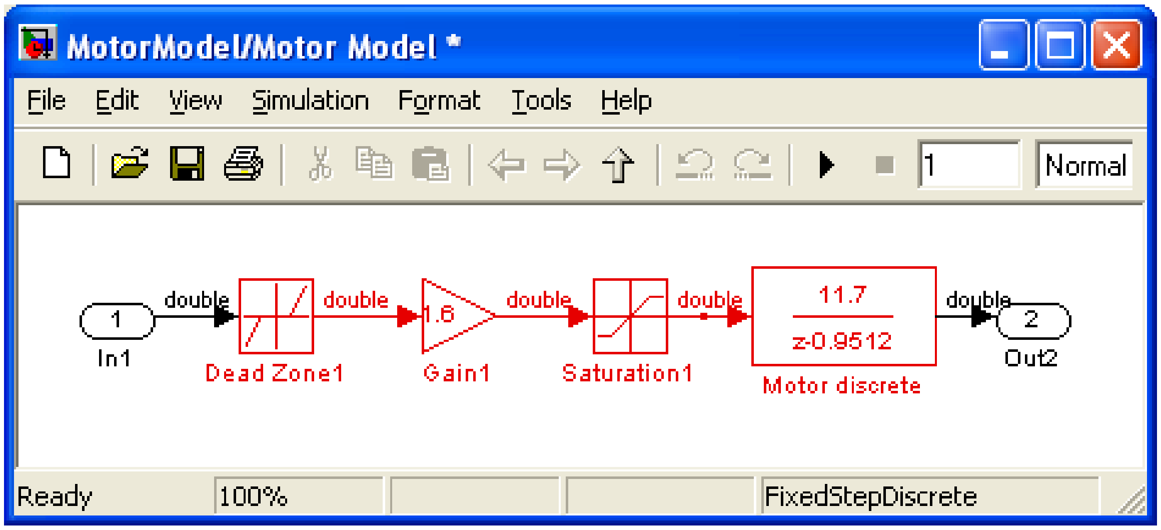


Figure 14: DC Motor Model

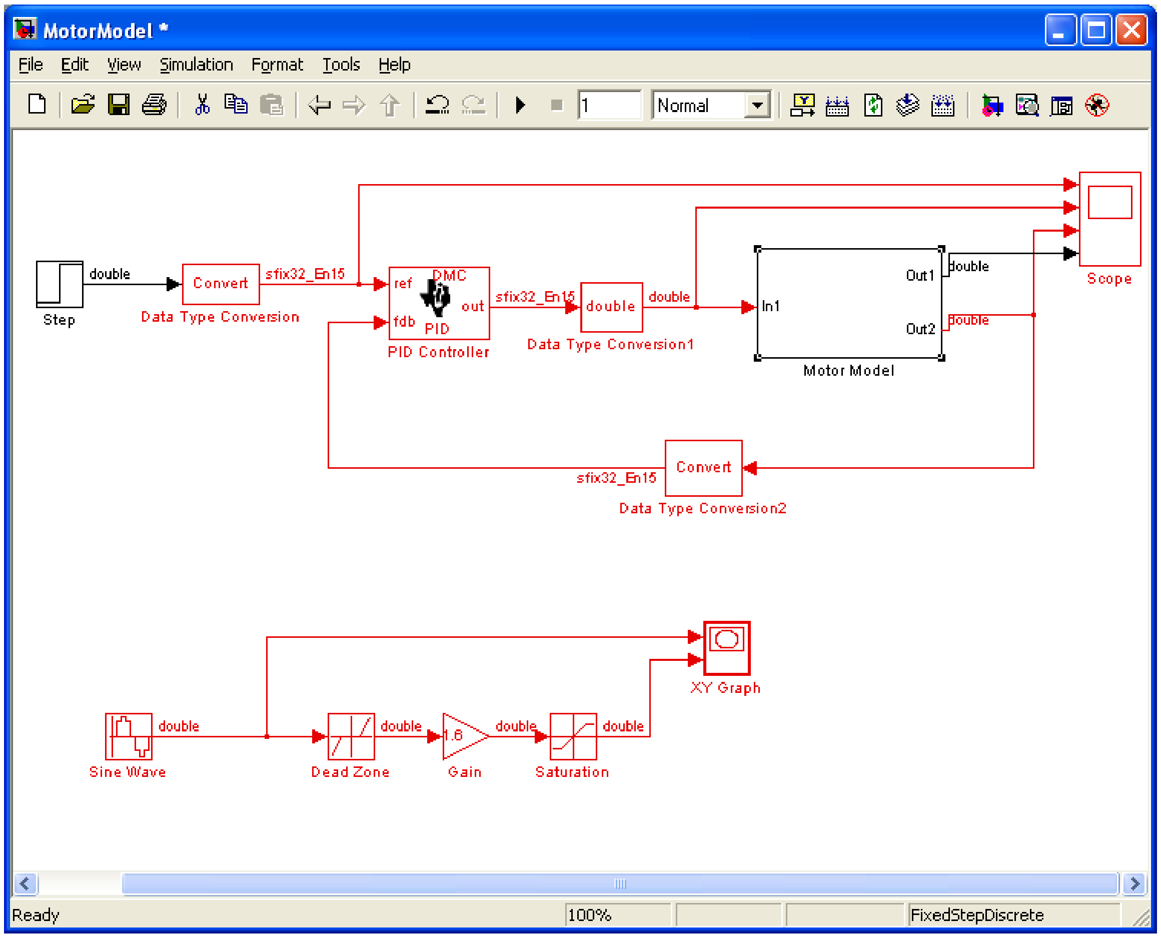
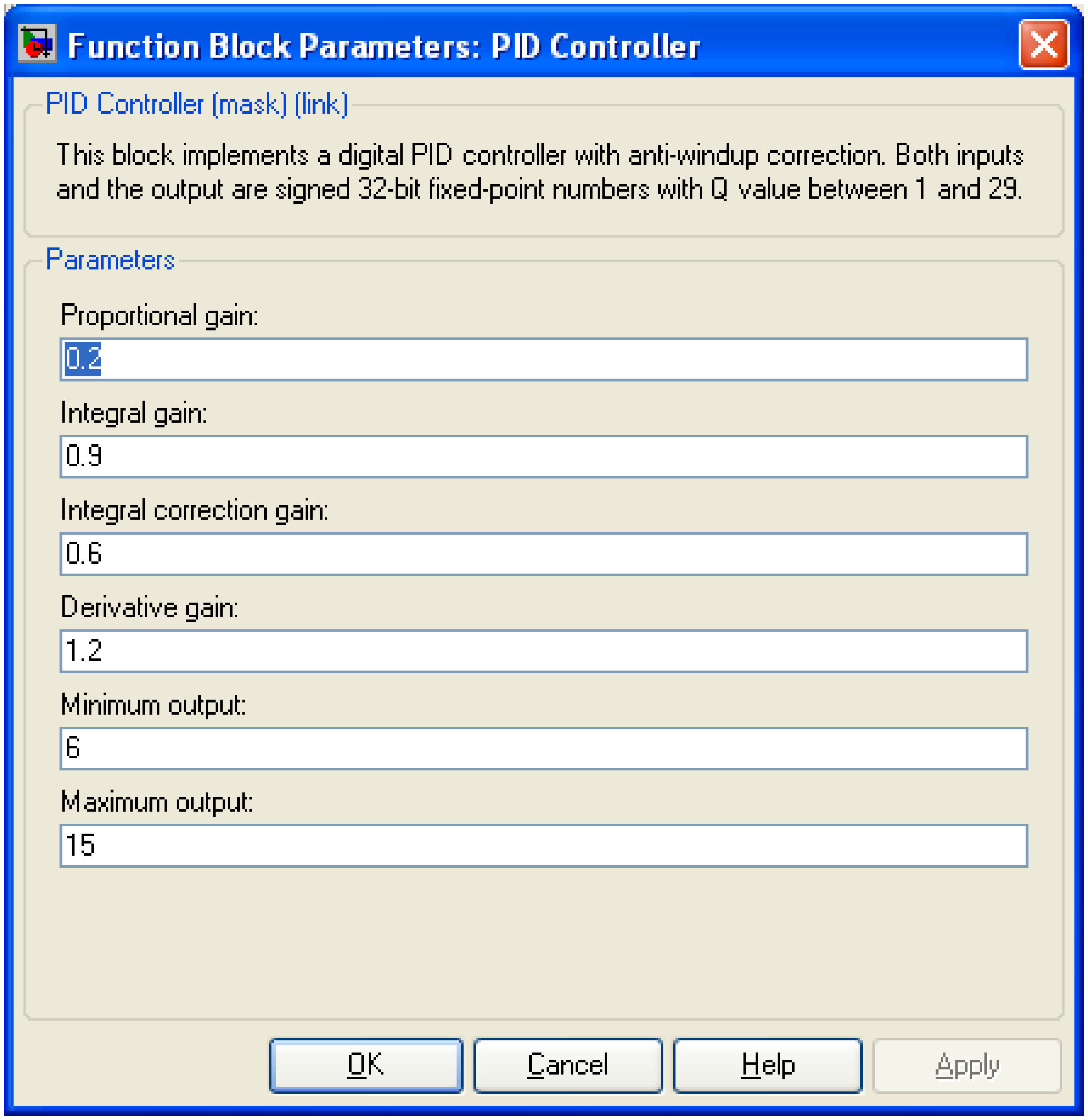


Figure 15: Loop Control Model


Function Block Parameters: PID Controller

PID Controller (mask) (link)

This block implements a digital PID controller with anti-windup correction. Both inputs and the output are signed 32-bit fixed-point numbers with Q value between 1 and 29.

Parameters

Proportional gain: 0.2

Integral gain: 0.9

Integral correction gain: 0.6

Derivative gain: 1.2

Minimum output: 6

Maximum output: 15

Figure 16: PID Controller Parameters

3.2 Hardware Design

The connection between the eZDSP and the Motor Kit requires a dedicated interface circuit, implementing the Low Pass Filter (used for the Digital to Analog conversion) and adapting the voltage levels at the eZDSP (0 to 3V) to those of the DC Motor Kit (0-15V). The various signal types and their voltage range are shown in .

The interface circuit contains two blocks. The first is LPF with an amplifier (shown in Figure 18), connected between the PWM output and the servo amplifier input. The second block is an attenuator (shown in Figure 19) connected between the tachometer output and the Analog to Digital Input of the eZDSP.

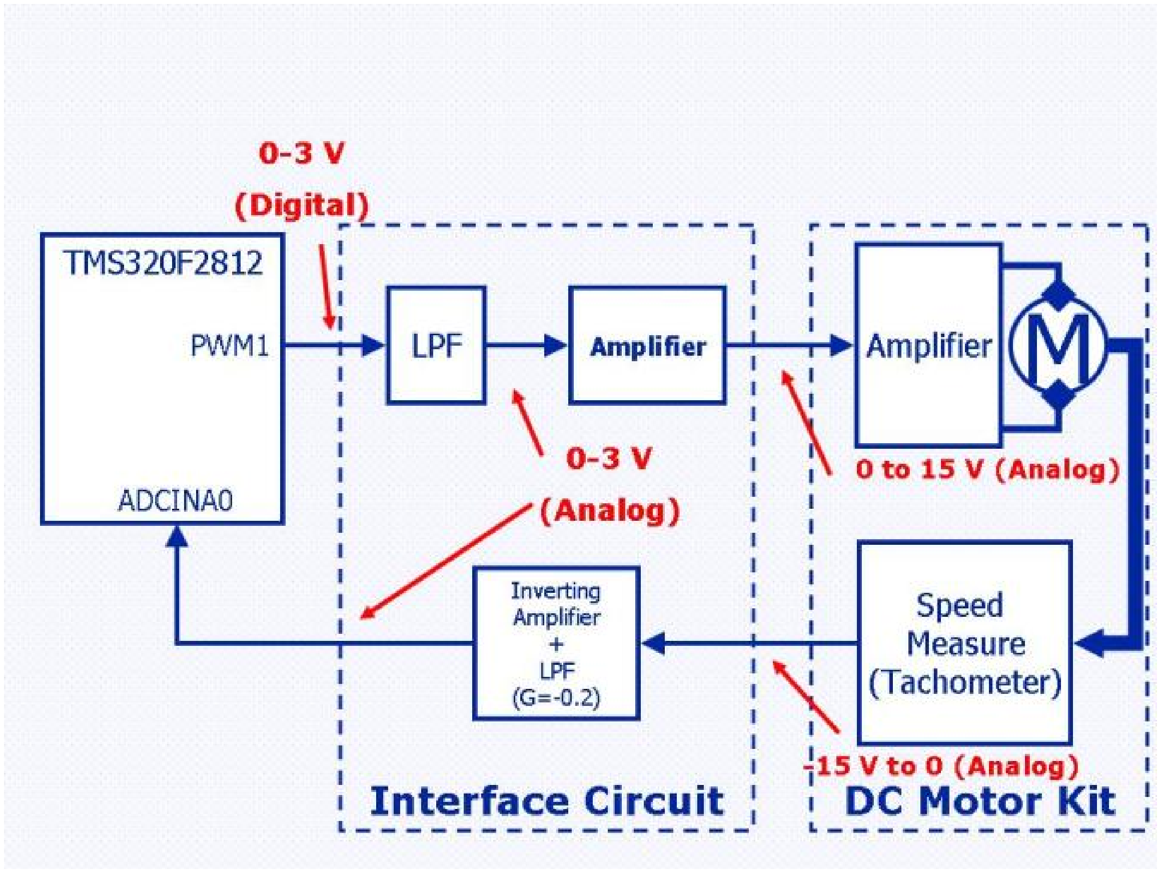


Figure 17: Signal Types and Voltage Levels

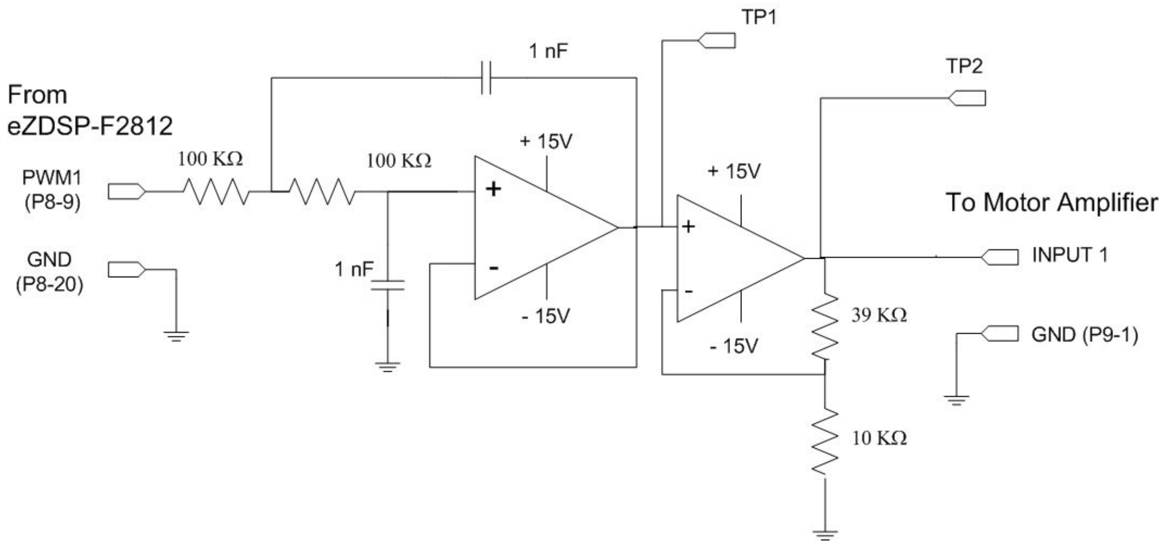


Figure 18: LPF + Amplifier Circuit

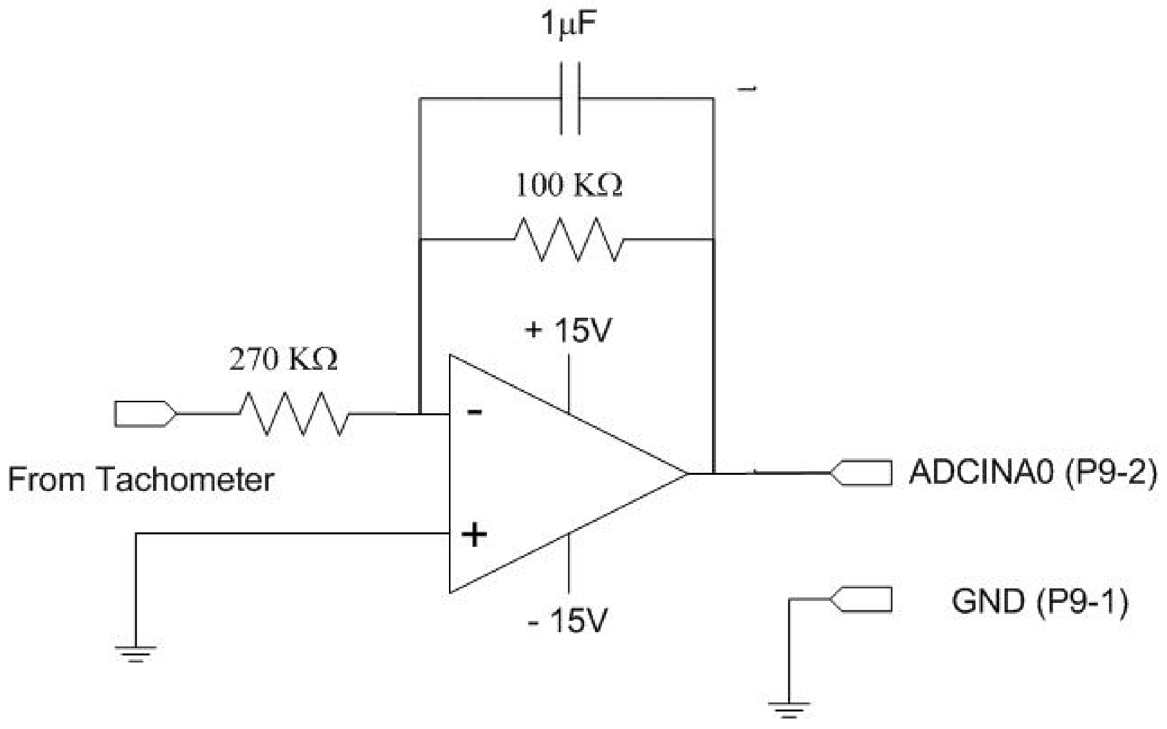


Figure 19: The Inverting Amplifier with LPF

4 Real Time Implementation

The control loop was implemented in the eZDSP F2812 as shown in Figure 20. The PID controller obtained in the previous chapter was implemented in the DSP. The environment is shown in Figure 21.

The real-time implementation model will be created from the "DC Motor Speed Control via RTDX" SIMULINK demo. In the original demo model the loop is closed by QEP block, we will use a tachometer connected to the Analog to Digital Converter module for speed measurement.

The Model and its subsystems are shown in Figures Figure 22, Figure 23 and Figure 24.

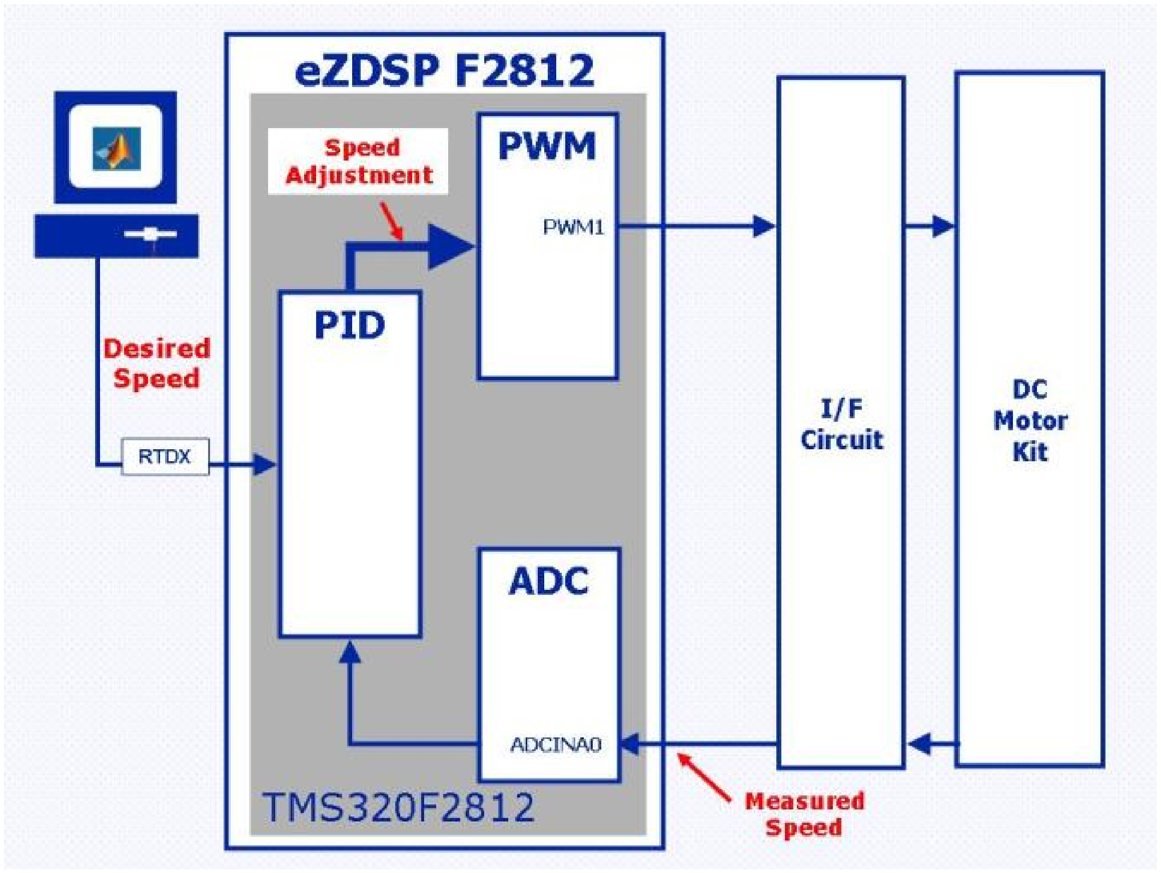


Figure 20: LPF + Amplifier Circuit>

Procedure:

1. Navigate into the following directory:

..\MATLAB\R2006b\toolbox\rtw\targets\tic2000\tic2000demos

1. Copy the following files into your working directory:
   - c2812speedcontrolDC.mdl
   - runc2812speedcontrolDC.m
   - runc2812speedcontrolDC.m
   - Open the c2812speedcontrolDC.mdl model and save it as "DCMotorControlc2812.mdl"(please refer to Figure 21[4]).

[4]The model is shown here after deleting the “Info” box.

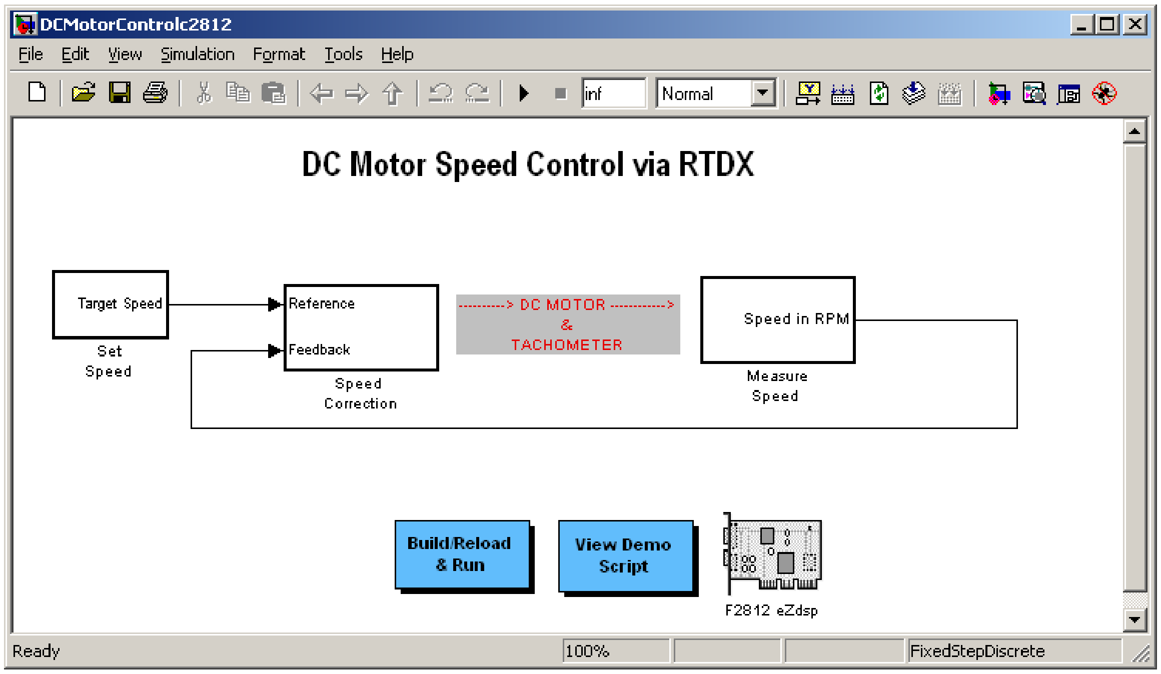


Figure 21: Real-Time Implementation Model

2. Double-click the Speed Correction Block, and you will see:

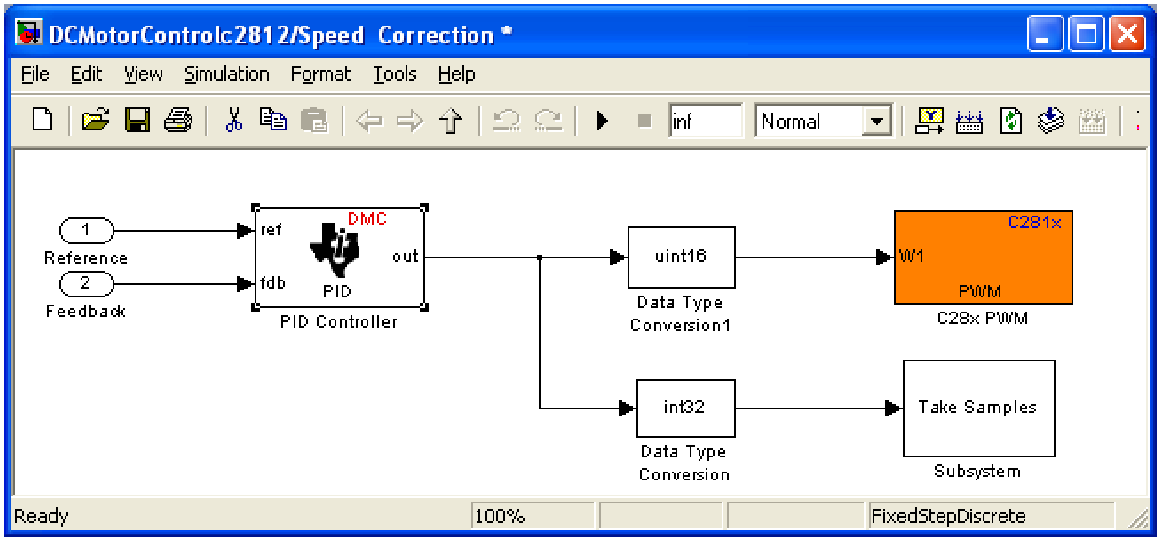
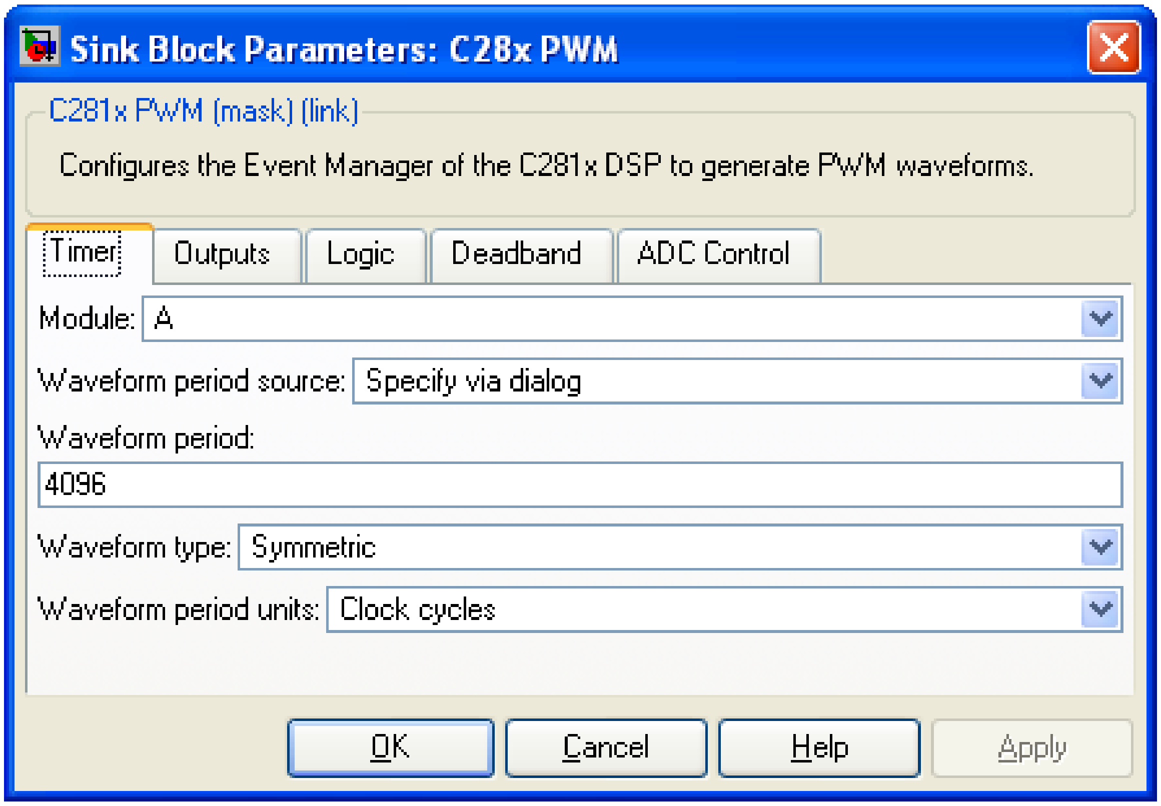


Figure 22: Figure 22: Speed Control

3. Configure the C28x PWM block as follows:



Sink Block Parameters: C28x PWM

C281x PWM (mask) (link)

Configures the Event Manager of the C281x DSP to generate PWM waveforms.

Module: A

Waveform period source: Specify via dialog

Waveform period:

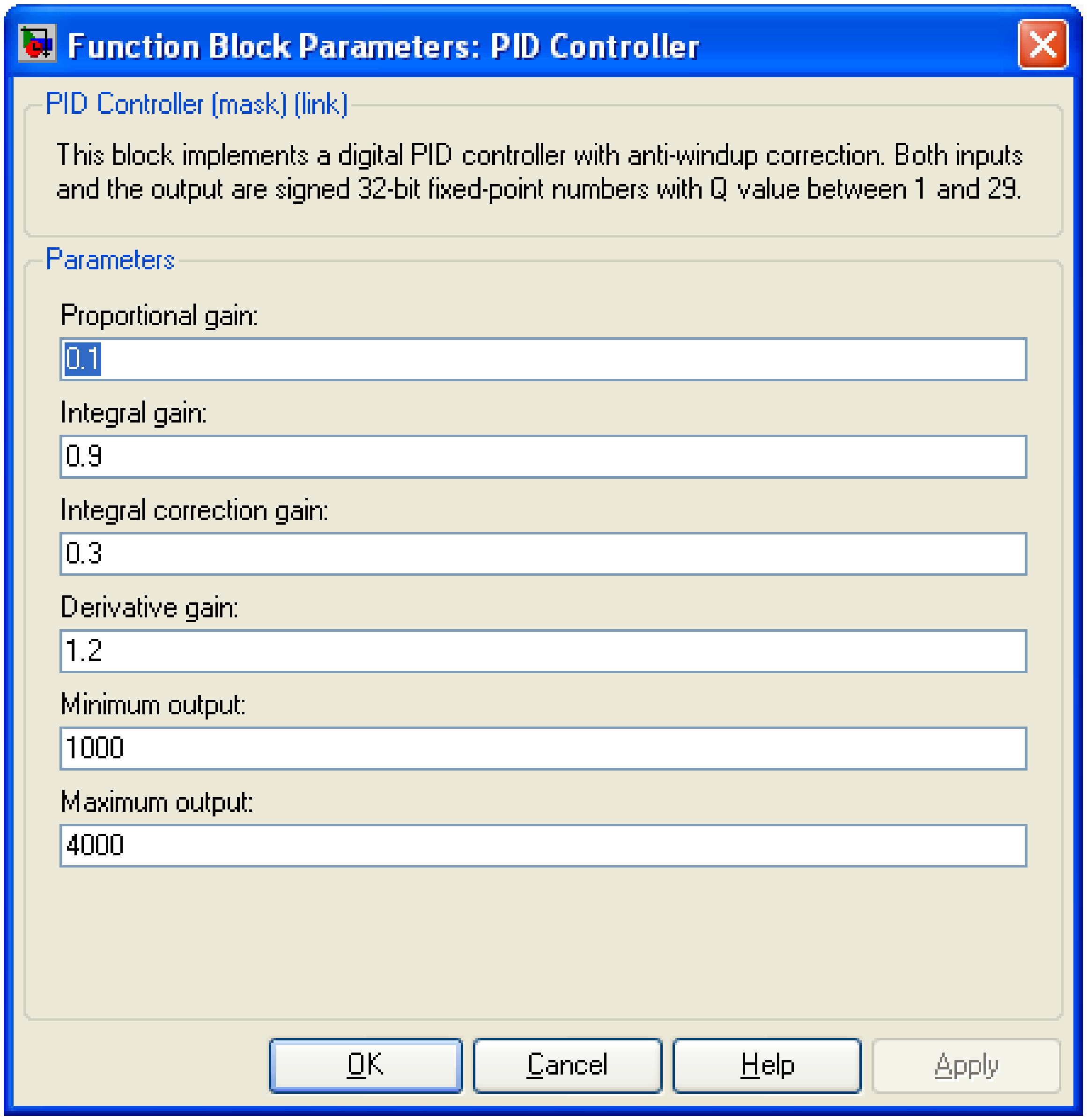
4096

Waveform type: Symmetric

Waveform period units: Clock cycles

Figure 23: Figure 23: PWM Configuration

4. Configure the "PID Controller" block as follows:



PID Controller (mask) (link)

This block implements a digital PID controller with anti-windup correction. Both inputs and the output are signed 32-bit fixed-point numbers with Q value between 1 and 29.

Parameters

Proportional gain: 0.1

Integral gain: 0.9

Integral correction gain: 0.3

Derivative gain: 1.2

Minimum output: 1000

Maximum output: 4000

Figure 24: PID Configuration

5. Double-click the Speed Correction Block, and you will see:

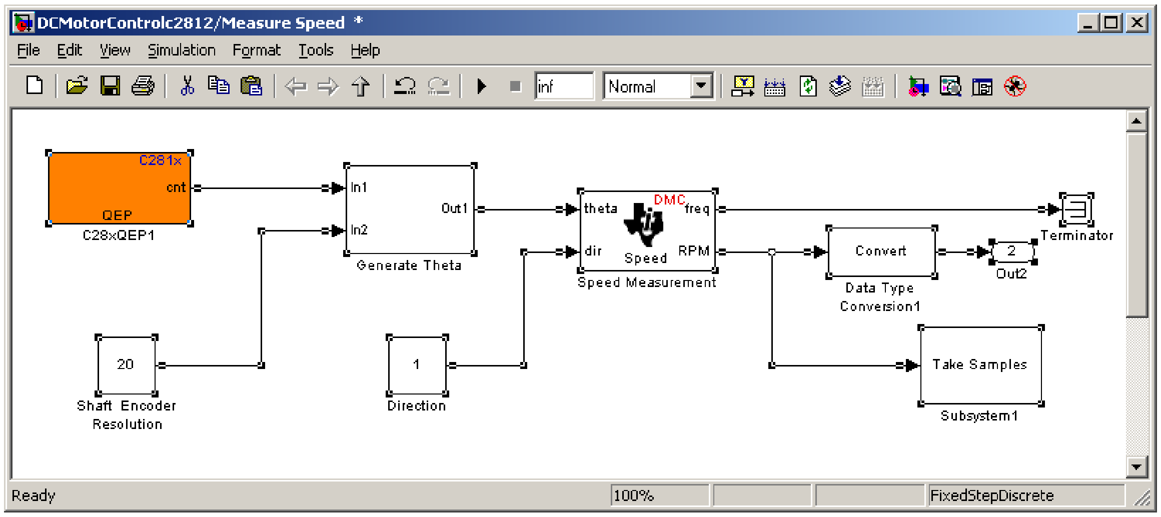


Figure 25: QEP Based Speed Measurement

6. Delete the selected blocks and replace them by the ADC block from Open the ADC block from the C281x Chip support group from the C2000 Target Preferences, and connect is as follows:

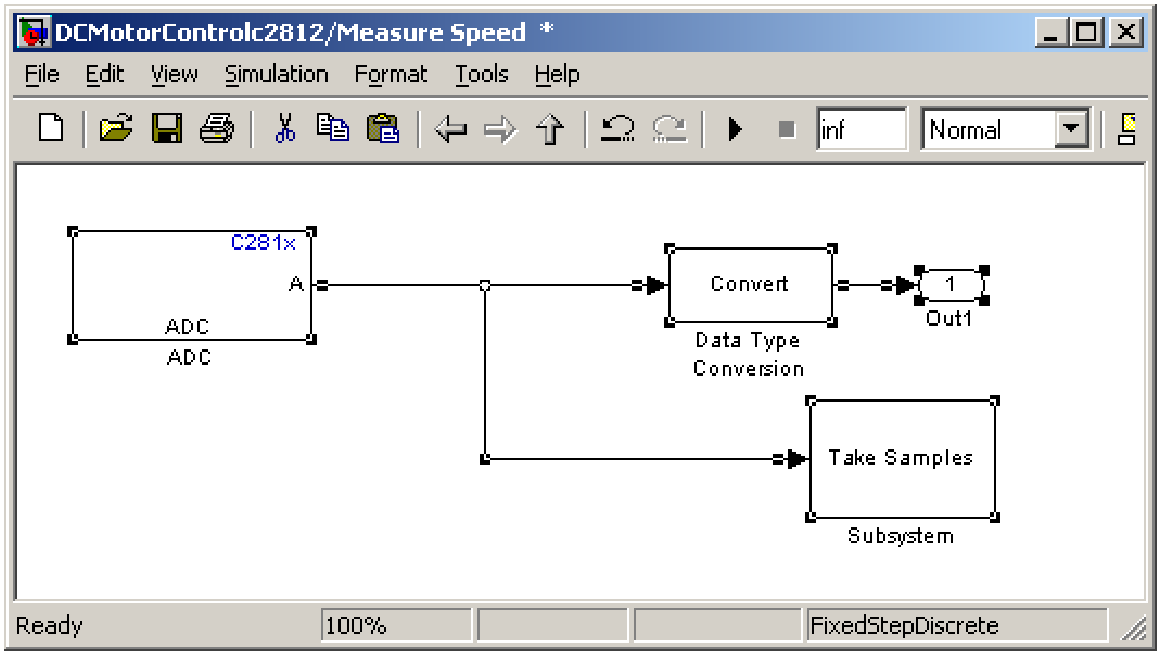
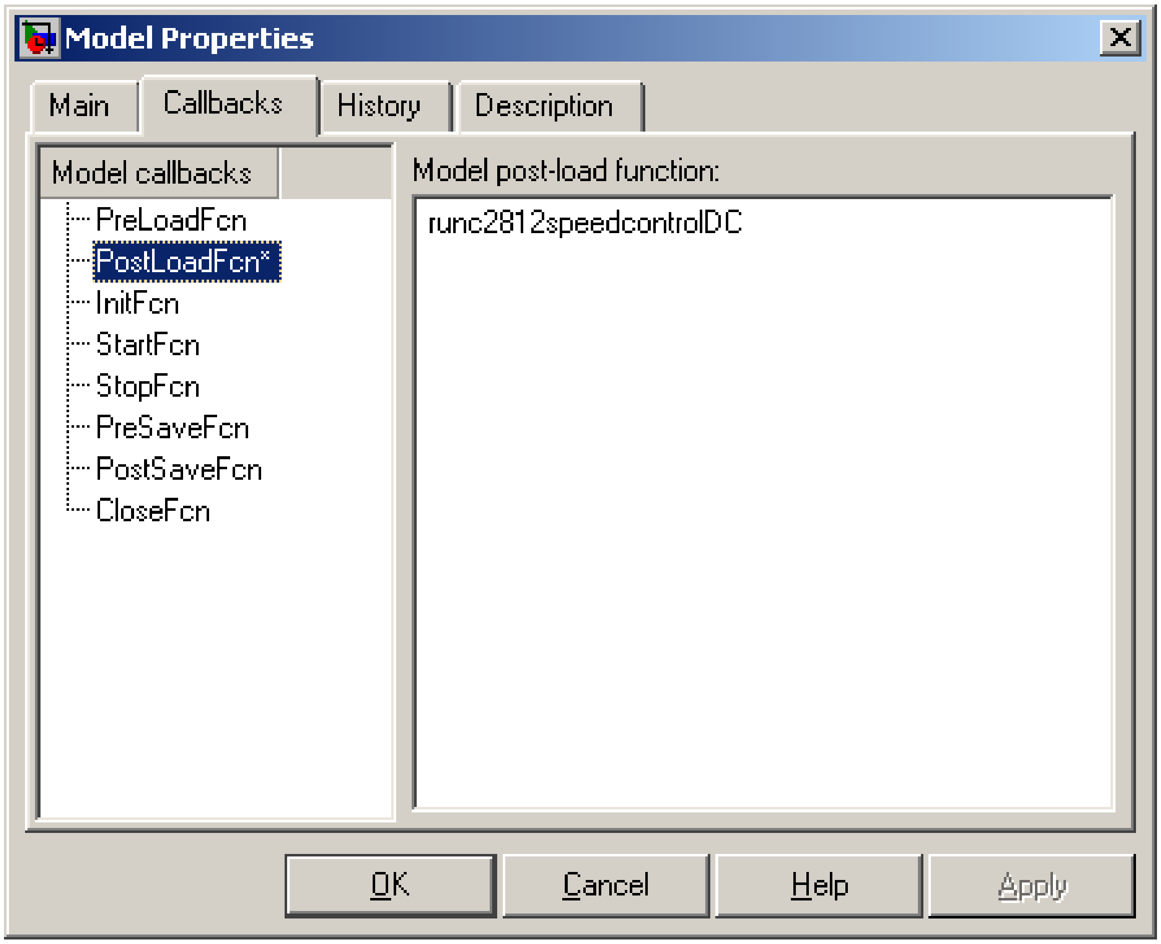


Figure 26: ADC+Tachometer Based Speed Measurement

7. Now the model is ready for real-time, we need however to update the MATLAB script file. Open the "Model Properties" from "File" menu. Change the PostLoadFcn callback to runc2812speedcontrolIDC,

as shown:



Model post-load function:

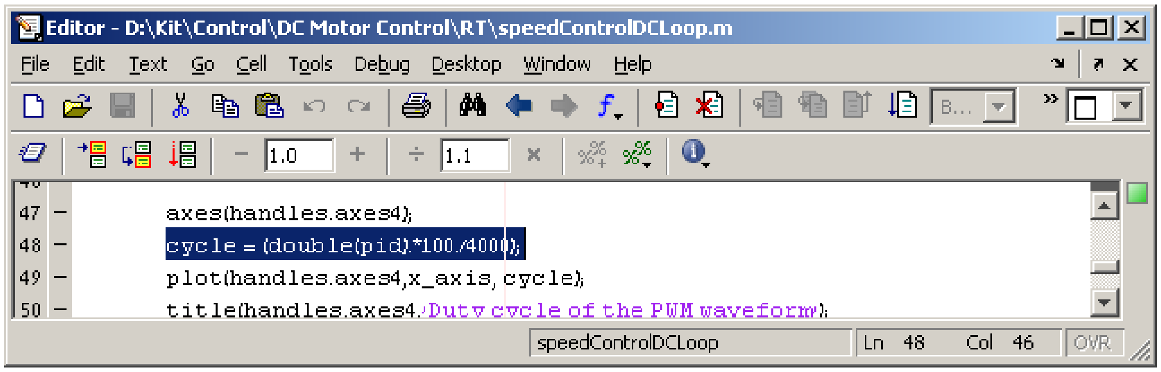
runc2812speedcontrolDC

Figure 27: Model Callback

8. The next step is to change the original PWM range (up to 64000) to the desired range (up to 4000). Open the "speddControlIDCLoop.m" file with the MATLAB editor, an change on line 48 the command:

```
cycle = (double(pid).*100./64000);
to
cycle = (double(pid).*100./4000);
```

Please refer to the following picture:


Editor - D:\Kit\Control\DC Motor Control\RT\speedControlDCLoop.m
File Edit Text Go Cell Tools Debug Desktop Window Help

```
axes(handles.axes4);
cycle = (double(pid).*100./4000);
plot(handles.axes4,x_axis, cycle);
title(handles.axes4,'Duty cycle of the PWM waveform');
```

speedControlDCLoop
Ln 48 Col 46
OVR

Figure 28: Changing the PWM range

9. Activate the motor
10. Activate CCS.
11. Click the Build/Reload&Run box, the following window should appear:

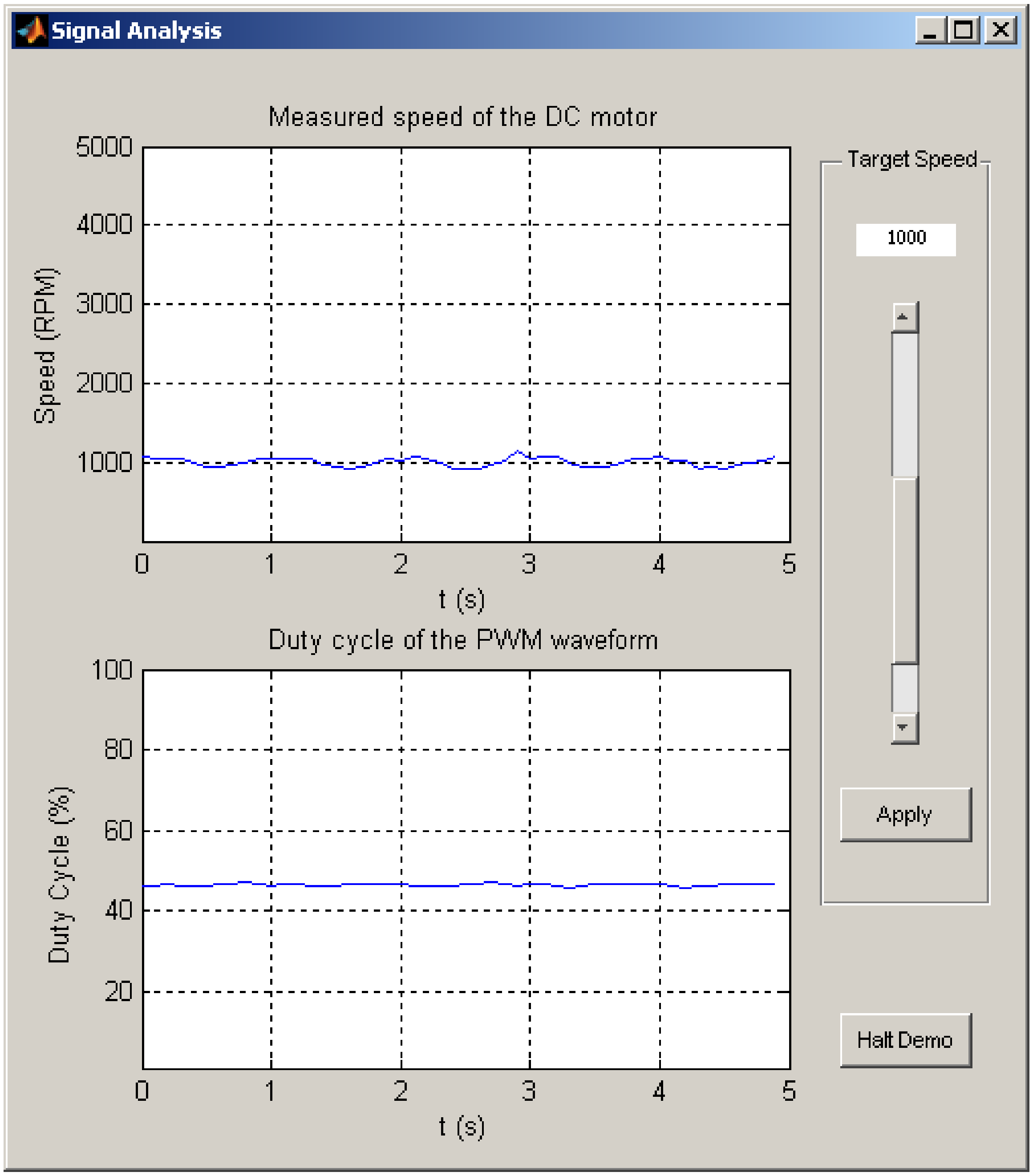


Figure 29: Speed Control GUI

12. You may change the speed of the motor using the slider in the right hand side and the "Apply" button.

5 References

1. "eZdspTM F2812 Technical Reference", Spectrum Digital, 2003 http://c2000.spectrumdigital.com/ezf2812/docs/ezf2812

2. David M. Alter, " Using PWM Output as a Digital-to-Analog Converter on a TMS320F280x", TI Application Report SPRAA88 , September 2008 http://www.ti.com/litv/pdf/spraa88a[6]

[5]http://c2000.spectrumdigital.com/ezf2812/docs/ezf2812_techref.pdf
[6]http://www.ti.com/litv/pdf/spraa88a