Vehicle Management Unit: From Model-Based Design to Code Generation

Master of Science in Mechatronic Engineering

Instructor: Professor Stefano Carabelli

Fargham Ahmad

ABSTRACT

The aim is to determine the feasibility and implementation of a vehicle management unit on a RCP or an embedded ECU, following a model-based design and code generation for implementation. With an embedded system there arises the problem of I/O interfaces as they are always adapted from system to system according to signal dynamics. The feasibility study is conducted by following the V-model scheme. Since control strategies are already well known, the thesis presents the problem of simulating the delays introduced at the I/O interfaces of an embedded system. In addition, digital filters introduce delays, which may cause the system to be unstable, mitigation of this issue is also addressed to have a well-tuned system.

To understand the effects of the I/O interfaces and the signal conditioning components used to adapt the signal, a model-based scheme is used to form a base model. The base model represents the concept. In the next cycle of the development an enhanced model is produced containing the signal conditioning and conversion components, such as ADC, S/H units, to consider the delays due to I/O interfaces. Upon completion the enhanced model is validated by comparing the results with the base model. Using target hardware support package from Matlab/Simulink, ADC and DAC blocks are used as means of I/O for code generation model. The core part of enhanced model that performs the core functionality is reused to complete the code generation model and code is generated automatically. Following it, experiments are conducted to determine the exact the delays for signal traversal. The enhanced model is calibrated by adding the delay to the system model, this way the model is valid as the delays at the I/O interfaces are also taken into consideration.

For tuning the digital signal at input and output, the input signal is averaged periodically and at the output the signal is smoothened. The signal is acquired at a higher frequency than the computation frequency, this way the input signal is an average value for every computation period. Furthermore, at the output the signal is linearly smoothened to have smooth signal. Also, the input signal is automatically scaled to it's native values and units upon acquisition and then rescaled to bits after core functions are performed. All the strategies are implemented using model-based design and are compatible for automatic code generation for any target hardware, that is supported by Matlab/Simulink.

In conclusion, a model is produced that is parameterized to model any target hardware with it's specifications. However, this model will only have to be calibrated once for every target hardware. The calibration can be done by knowing the I/O delay of the target by experimenting once. For the input and output signal conditioning in code generation model, code can be generated and deployed automatically for any supported target. This makes the model reusable for any RCP or embedded ECU supported by Matlab/Simulink for code generation.

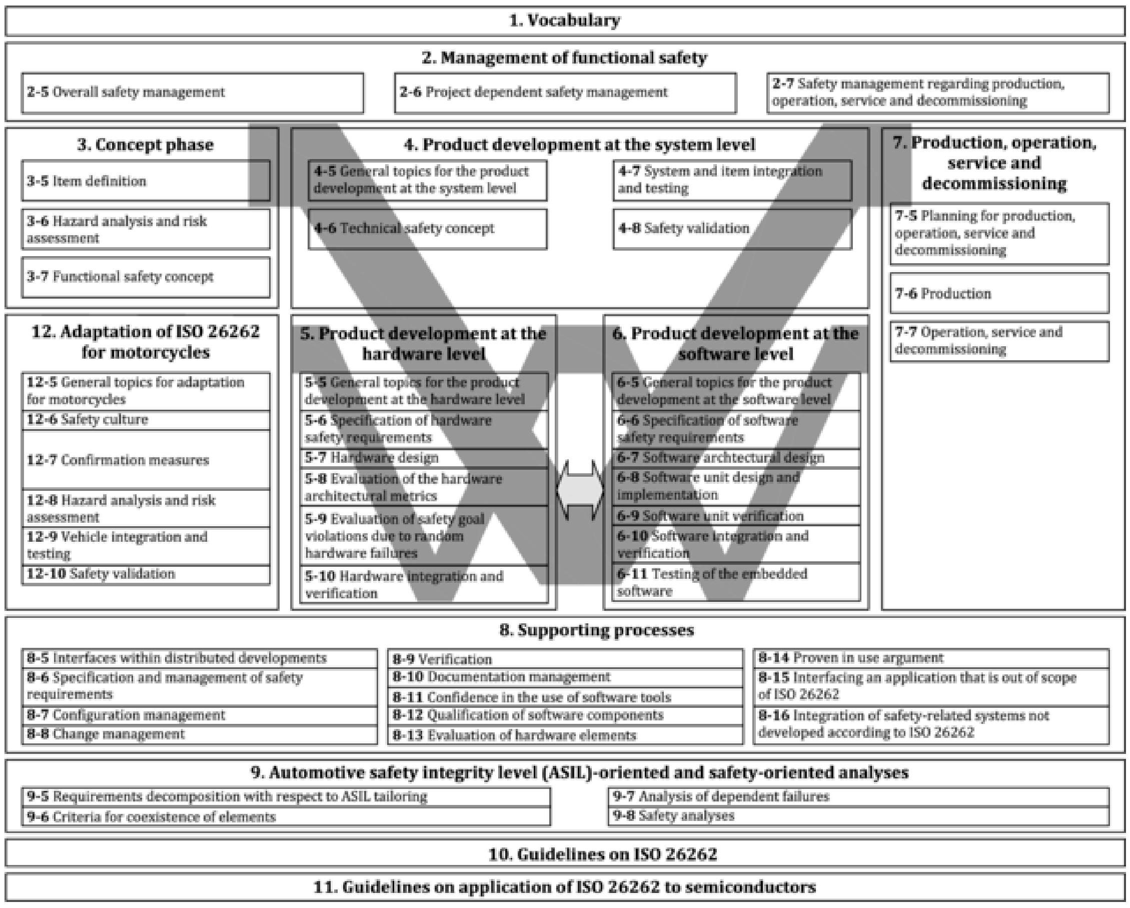
CONTENTS

Introduction

A. Objective

Vehicle management unit comprises of powertrain control and vehicle management. The thesis addresses the production of such a unit from model-based design to code generation for production. As the product is for production, it's mandatory that it meets the standard requirements for certifications, in particular vehicle functional safety. Hence, all the development phases must be carried out following the V-model scheme set by ISO-26262 Road Vehicles-Functional safety.

To go from concept to production in an industry standard approach results in a more acceptable product, which reduces cost in terms time, there is no need to map the development cycle to industry standard. The figure below outlines the steps for the product development cycle, which is ISO-26262 compliant.



1. Vocabulary
2. Management of functional safety
2-5 Overall safety management
2-6 Project dependent safety management
2-7 Safety management regarding production, operation, service and decommissioning
3. Concept phase
3-5 Item definition
3-6 Hazard analysis and risk assessment
3-7 Functional safety concept
4. Product development at the system level
4-5 General topics for the product development at the system level
4-6 Technical safety concept
4-7 System and item integration and testing
4-8 Safety validation
7. Production, operation, service and decommissioning
7-5 Planning for production, operation, service and decommissioning
7-6 Production
7-7 Operation, service and decommissioning
12. Adaptation of ISO 26262 for motorcycles
12-5 General topics for adaptation for motorcycles
12-6 Safety culture
12-7 Confirmation measures
12-8 Hazard analysis and risk assessment
12-9 Vehicle integration and testing
12-10 Safety validation
5. Product development at the hardware level
5-5 General topics for the product development at the hardware level
5-6 Specification of hardware safety requirements
5-7 Hardware design
5-8 Evaluation of the hardware architectural metrics
5-9 Evaluation of safety goal violations due to random hardware failures
5-10 Hardware integration and verification
6. Product development at the software level
6-5 General topics for the product development at the software level
6-6 Specification of software safety requirements
6-7 Software archtectural design
6-8 Software unit design and implementation
6-9 Software unit verification
6-10 Software integration and verification
6-11 Testing of the embedded software
8. Supporting processes
8-5 Interfaces within distributed developments
8-6 Specification and management of safety requirements
8-7 Configuration management
8-8 Change management
8-9 Verification
8-10 Documentation management
8-11 Confidence in the use of software tools
8-12 Qualification of software components
8-13 Evaluation of hardware elements
8-14 Proven in use argument
8-15 Interfacing an application that is out of scope of ISO 26262
8-16 Integration of safety-related systems not developed according to ISO 26262
9. Automotive safety integrity level (ASIL)-oriented and safety-oriented analyses
9-5 Requirements decomposition with respect to ASIL tailoring
9-6 Criteria for coexistence of elements
9-7 Analysis of dependent failures
9-8 Safety analyses
10. Guidelines on ISO 26262
11. Guidelines on application of ISO 26262 to semiconductors

Figure A: The V-model Scheme by ISO-26262.

B. Framework For Development Cycle

Since, the domain of thesis lies towards software, hardware design is not addressed, only hardware selection is of importance to meet the specifications. The development cycle scheme issued by ISO-26262, is the basis for vehicle management unit production. All the development is done by following a model/based design. The 5 steps in which the development cycle is iterated, as follows:

1) Concept Phase:
   It involves developing an ideal model, which simulates the basic idea for prototyping. This phase deals with feasibility of the project. This is step 3 in the scheme issued by ISO-26262.

2) Product development at the system level:
   The concept phase is realized to meet design specifications, that includes all the components of the system. The model created is to replicate the real behavior of the system. This model is developed after iterations and testing, until the model is a proper fit. This is step 4 in the scheme issued by ISO-26262.

3) Product development at the hardware level:
   The hardware for prototyping can be an embedded ECU by Texas Instruments or an ECU that is bypassed externally by a RCP by dSPACE, also RCP by dSPACE can be a standalone ECU. The later allows for more functionality, however there are some latencies in the I/O as the signals are exchanged between RCP and ECU, also ECU is still the controller that is closing the feedback loop. Bypassing ECU externally can run new control functions on the RCP, and the original control functions on the existing ECU. Furthermore, RCP can be used for control optimizations and data collection for analysis, as it's a more capable device in terms of computation. This is step 5 in the scheme issued by ISO-26262. However, no development is done at hardware level.

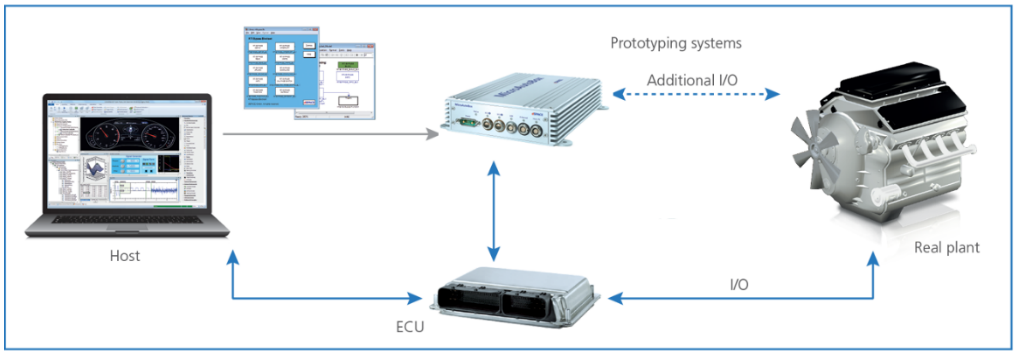


Figure B: The ECU bypass or Standalone Scheme.

4) Product development at the software level:
   The software is C code that is generated from Simulink after following a model-based design. This is step 6 in the scheme issued by ISO-26262.

5) Final-Testing:
   It consists of comparing the results obtained from simulation and the behavior of the plant(vehicle). For unsatisfactory results, the complete process is iterated after applying fixes.

Chapter 1-Model Descriptions

This chapter outlines the modelling strategy used and the motivation for considering and developing an enhanced model. As it is known that the model-based design follows a V-cycle scheme, the following is based on the framework introduced previously. The figure below mentions the different stages of the development cycle. The different stages represent different models. Each model is approached in sequence to have a model in the end that simulates the system in similar way. All the simulations are conducted on Matlab/Simulink environment, following a model-based design.

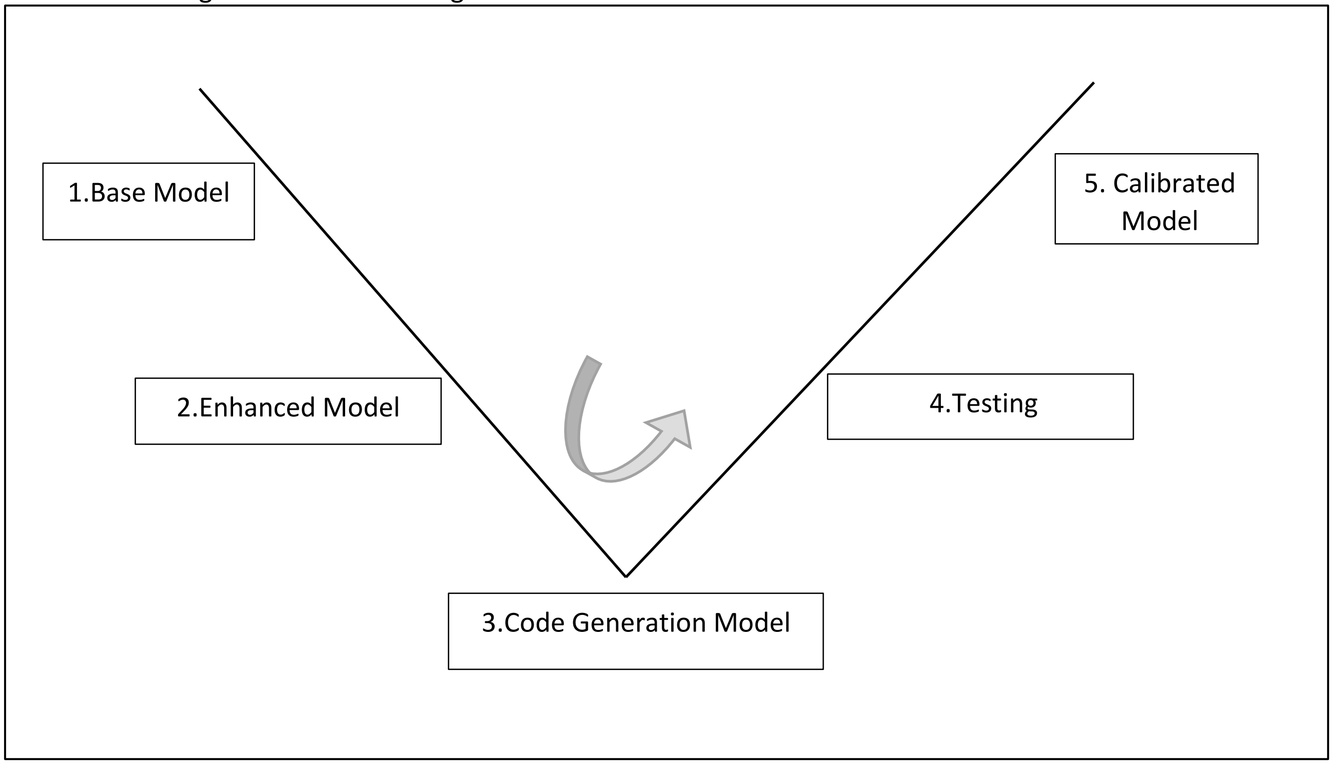


Figure 1.1: The adapted development scheme.

1. Base Model
   The simplest model that transforms the idea to a model. This phase is the concept phase in the framework, and it is the most crucial part of the development and it contains, only the main task. As this is the simplest model, the tradeoff is in neglecting the model details, the operation conditions, etc. The results obtained from this model are not enough to have a complete behavior. The model is not valid as it doesn't simulate the system entirely. Also, base model is a reference model for validation of the enhanced model and the code generation model.

2. Enhanced Model
   The enhanced model takes into consideration the base model and adds to it the components which enhance the model in such a way that it represents the system. This phase is product development at the system level. This model is the model that is required to have in order to completely understand the system and its behavior. For example, sampling and hold unit, A/D and D/A units. All these components are modelled for model simulation, as they include the I/O interfaces

Furthermore, the figure below depicts exactly what the enhanced model takes into consideration.

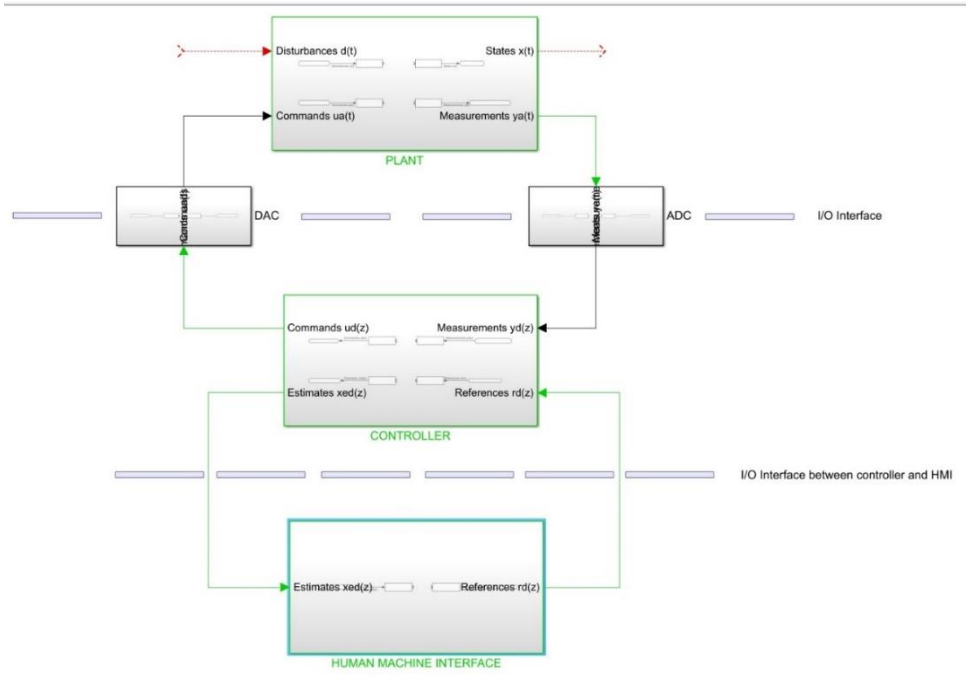


Figure 1.2: Model highlighting the I/O interface.

This model enables to analyze if the selected target hardware is the right choice. The model can replicate any embedded system's I/O interfaces. It models the latencies introduced at the I/O interfaces between the controller (target hardware) and the plant. The controller operates in the digital time domain, and the components used for the interfacing of controller and plant introduce time delays. This way all the sources of delays are considered, and the model is valid.

3. Code Generation Model
   The following model is the model that will be model-based, and the development will be carried out on Simulink. This phase is product development at the software level. In this part, the model for target deployment is carried out. This model differs the enhanced model by just I/O interface blocks, because the I/O i.e. signal acquisition and transmission is done via Simulink blocks. However, the core functionality part of model is exactly that of the enhanced model.

4. Testing
   An experiment is setup to observe the hardware behavior for which code is generated. This part is required to assess the performance of the system with the enhanced model. Experimental data is collected for model calibration.

5. Calibrated Model
   The final part is the calibration of the enhanced model to make adjustments so that the enhanced model is completely valid. Gains and delays are adjusted in the enhanced model to have a better model.

Chapter 2-Development Cycle Implementation Example For An Embedded ECU

2.1 Base Model

2.1.1 Introduction And Objective

The base model takes into consideration the concept. Since, the objective is to determine the I/O latency, it is sufficient to use a simple lowpass filter on the microcontroller and observe the delay. However, this is just base model that is not meant for target deployment, rather it is transforming the idea to the concept model. Also, worth noting is that in the simulation the hardware details are completely neglected.

The concept is to implement a lowpass filter to observe the signal behavior at the input/output boundary of interface. The concept of the modelling is to determine the output on an oscilloscope from a lowpass filter when a reference signal is applied. The reference signal and filtered output are compared to draw conclusions.

2.1.2 Description

The model is composed of a signal generator block, an analog low pass filter and a scope from the Simulink library. Beneath is the description of signals and the signal parameters.

Table 2.1: The description of Signals

Name	Symbol	Unit	Range
Reference Signal	u(t)	V	[-5 to 5]
Filtered Output Signal	yf(t)	V	-

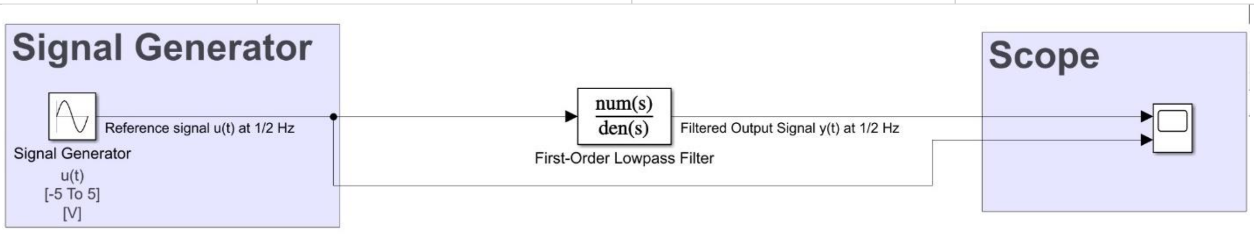

Signal Generator
Reference signal u(t) at 1/2 Hz
Signal Generator
u(t)
[-5 To 5]
[V]
num(s)
den(s)
First-Order Lowpass Filter
Filtered Output Signal y(t) at 1/2 Hz
Scope

Figure 2.1: The Simulink Scheme for the base model.

The scheme shown above comprises of a signal generator, a lowpass filter and a scope. The table below provides the model parameters.

Table 2.2: The description of Parameters

Parameter	Name	Unit	Value
Amplitude_Signal	Signal Amplitude (Generator)	V	5
Bias_Signal	Signal Offset (Generator)	V	0
Frequency_Signal	Signal Frequency (Generator)	Hz	1/2
CutOff_Freq	Cutoff Frequency (Filter)	Hz	8 × ½ = 4

The low pass transfer function is given by $\frac{1}{0.03979s+1}$. Where the cutoff frequency is 4Hz.

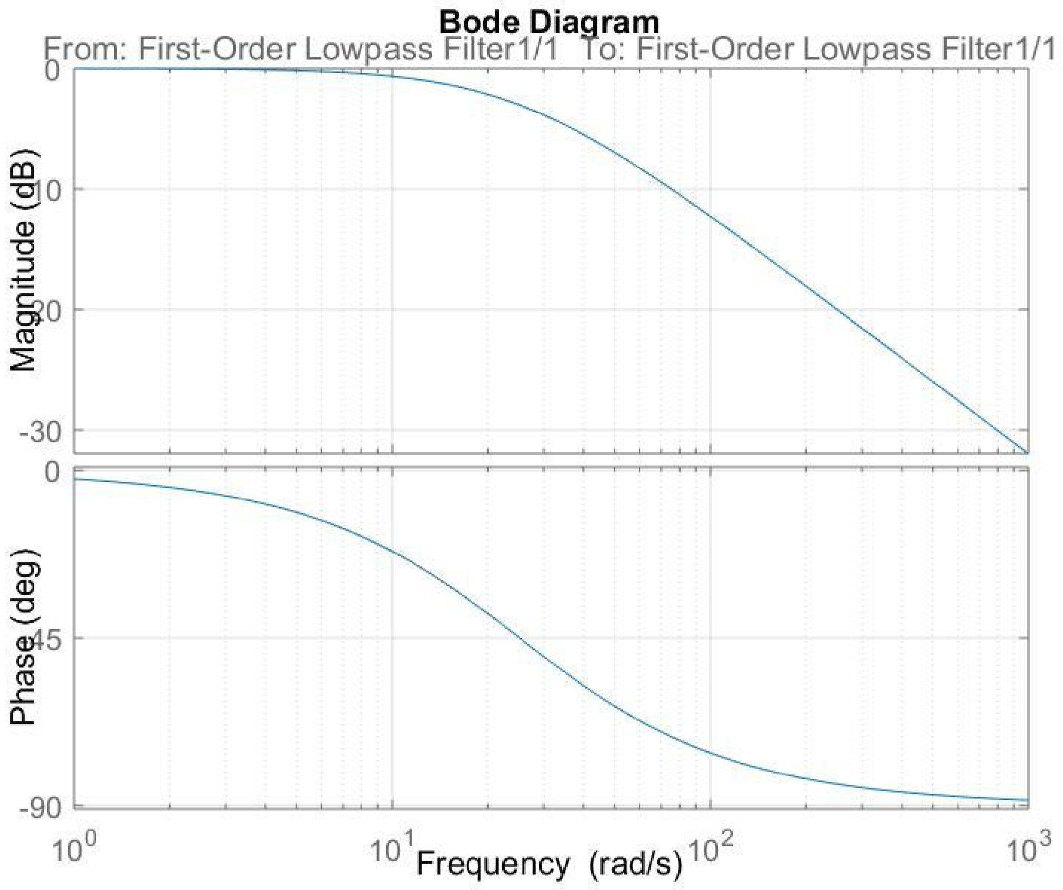


Figure 2.2: The frequency response of the analog lowpass filter.

2.1.3 Theoretical Behavior

The Filtered Output signal y(t) is expected to be different from u(t) in two aspects:

1. The phase shift is due to the time it takes to charge the plates of the capacitor, as the input increases. Thus, resulting in the output voltage lagging behind that of the input signal. The higher the input frequency applied to the filter the more the capacitor lags and the circuit becomes more and more out of phase.
2. The amplitude attenuation is because a first-order filter has a gain of -3 dB at the cut-off frequency, but it's ideally approximated to 0 db (flat) in the passband region.

2.1.4 Simulated Behavior

The simulation is done at 3 different frequencies to know the outcome at different frequencies. It can be seen that there is a phase delay in all the cases, however there is a significant attenuation in the region that's not passband.

Signal Frequency Value	Symbol	Unit	Delay	Attenuation	Figure
½ × 2pi	W1	rad/s	41.4 ms	0.8%	1.1
¼ × 2pi	W2	rad/s	60.4 ms	0.23%	1.2
2 × 2pi	W3	rad/s	49.2 ms	12.5%	1.3

Table 2.3: Frequencies at which simulation performed

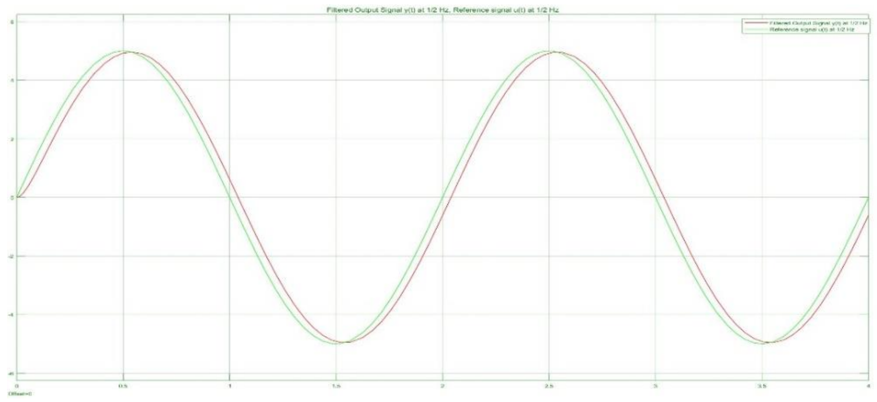


Figure 2.3: The input and output signals of the low pass filter at ½ Hz.

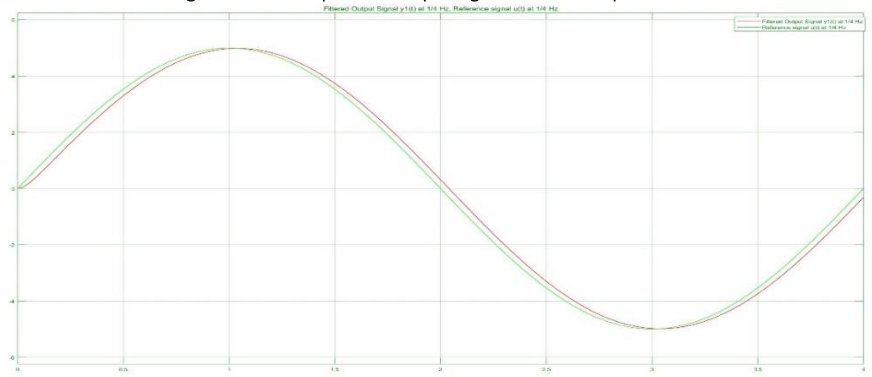


Figure 2.4: The input and output signals of the low pass filter at ¼ Hz.

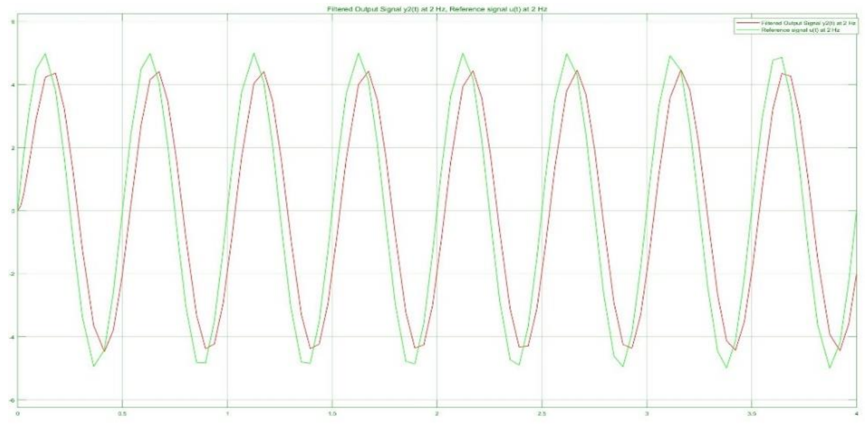


Figure 2.5: The input and output signals of the low pass filter at 2 Hz.

2.2 Enhanced Model

2.2.1 Introduction And Objective

The enhanced model is enhanced version of the base model as explained before. This model accounts for the components of the microcontroller or any RCP by considering acquisition and transmission specifications, which are added to the base model to enhance it. Hence, the enhanced model, models the ADC and DAC components of the hardware. This is used to study delays introduced that are risen by these components. All the components will be simulated in detail to have a reliable enhanced model. Also, the model is not hardware specific. The hardware specifications are parameters in the model, that can be changed to have a model of any target hardware. A lowpass filter is implemented, with same specifications that are in base model, but also considering models of the components of the target hardware.This model is based on Texas Instruments TMS320F28335 microcontroller with 12-bit ADC.

Furthermore, in the enhanced model the following components are to be modelled in addition to the base model:

1. Signal Conditioning block to adapt the input dynamics of the microcontroller.
2. Analog to digital converter block.
3. Digital filter, which can be implemented on any target by code generation.
4. Digital to analog converter block (A PWM based, with a lowpass filter).
5. Signal Conditioning block to adapt the reference input dynamics for the filtered output.

2.2.2 Description

2.2.2.1 Reference Signal Conditioning block

The signal conditioning block can be realized by 2 terms.

1. Scaling coefficient times the input signal.
2. Offset term.

Table 2.4: The description of Signal Conditioning block parameters used in Simulink

Name	Description	Symbol	Unit	Range
Reference Signal	Input signal from signal generator	u(t)	V	[-5 to 5]
Scaled Reference Signal	Input signal scaled to adapt microcontroller dynamics	$u_s(t)$	V	[0 to 5]
Scaling Coefficient	Scaling term, scales reference signal	m	-	-
Offset Term	Offset term	c	V	-
Analog_Input_Max	Microcontroller max input voltage (Parameter)	AM	V	3
Analog_Input_Min	Microcontroller min input voltage (Parameter)	Am	V	0
Max_Signal	Reference Signal maximum value	MS	V	+5
Min_Signal	Reference Signal minimum value	Ms	V	-5
Digital_Output	Max digital output voltage from microcontroller (Parameter)	DO	V	3.3

The equation will be a simultaneous equation of 2 variables 'm' and 'c' ,which can be solved by having at least 2 equations. The approached used in determining the coefficients is by finding a solution by substitution method.

The equation to determine Scaling coefficient is given by the following equation:

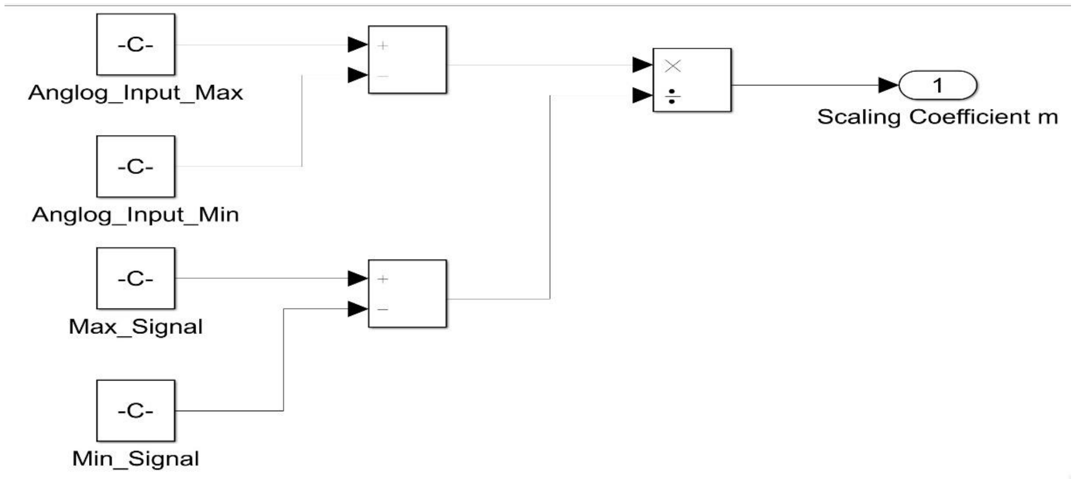


Figure 2.6: The scaling coefficient Simulink scheme.

The equation to determine Offset term is given by the following equation:

$$c = Am - Ms \times m$$

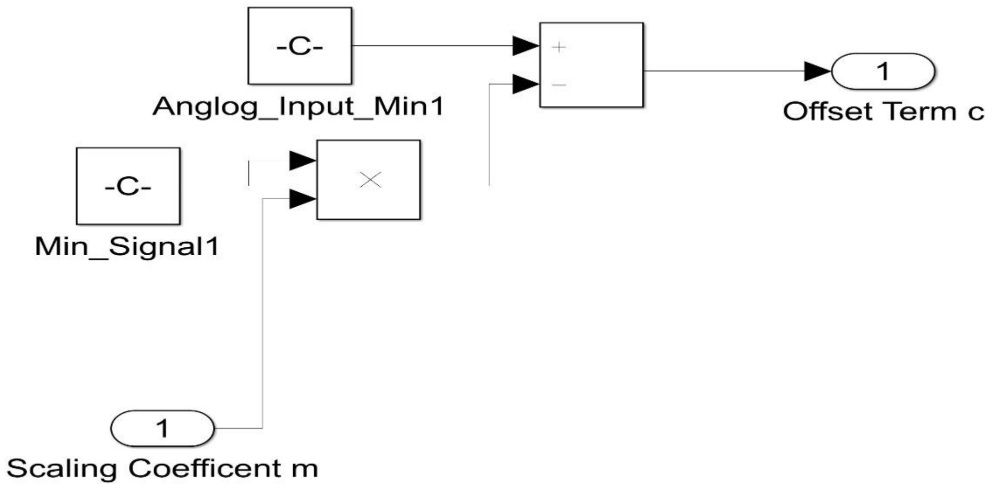


Figure 2.7: The offset term Simulink scheme.

The equation of input-output relation of block is given by:

$$u_s(t) = m \times u(t) + c$$

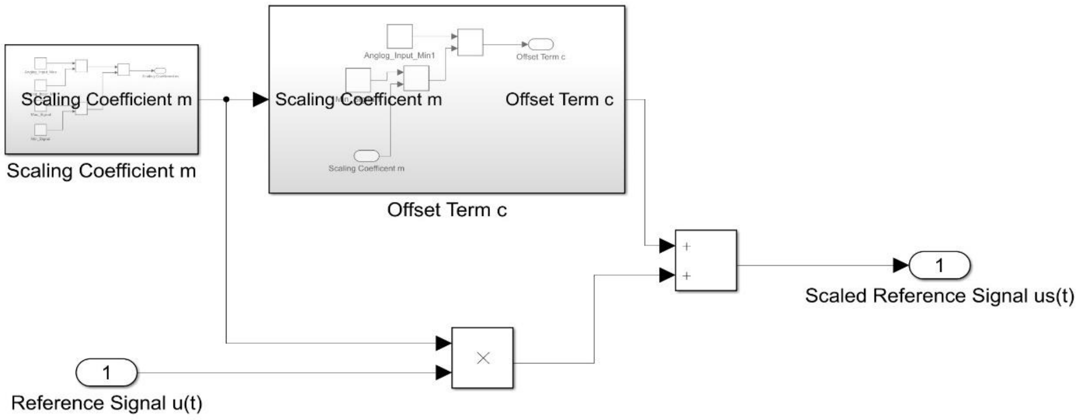


Figure 2.8: The signal conditioning between Reference signal and scaled output signal.

2.2.2.2 Analog to digital converter block

The Scaled Reference signal is converted from the analog form into the digital form to be filtered by the digital filter.

$$n = 2^{Bits} - 1$$
$$ud(z) = \frac{uspq(z) \times n}{FSR}$$

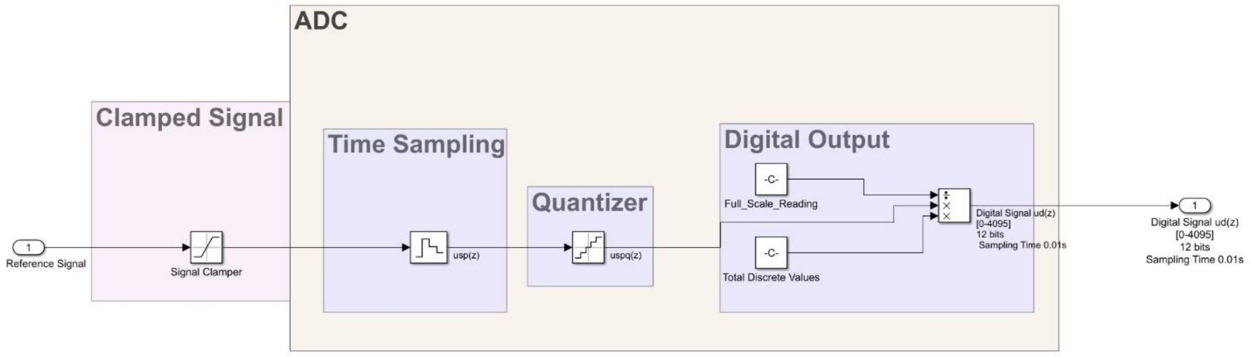


Figure 2.9: The ADC Simulink scheme.

Table 2.5: The description of ADC block

Name	Description	Symbol	Unit	Range
Total Discrete Values	2^Bits-1	n		[0 to 2^Bits-1]
Sampling Time	Sampling period of S/H block	Ts	s	80 mS
Full Scale Reading	Difference between maximum and minimum amplitude	FSR	V	-
Bits	The number of bits of the ADC	Bits	-	12
Sampled Signal	Signal sampled in time	usp(z)	V	[0 to 3]
Quantized and Discretized Signal	Signal sampled in time and quantized in values	uspq(z)	V	[0 to 3]
Digitalized Input Signal	Digital Signal	ud(z)	-	[0 to 4095]

The ADC subsystem consists of different modules and is modelled by Simulink blocks as follows:

1. Signal Clamper:
   To protect the microcontroller from any voltage that exceeds the input bounds. A saturation block is used from the Simulink library, which has the upper bound of the maximum allowed voltage, and a lower bound of the minimum allowed voltage of the microcontroller.

2. Time Sampling:
   Time sampling simulation can be achieved by using a zero-order hold block at the required sampling time.

3. Quantizer:
   Discretization of the amplitude is achieved by using a Quantizer block from Simulink library. The Quantization Interval (LSB) can be determined by the following equation.

$$Quantization\ Interval = \frac{FSR}{2^{Bits} - 1}$$

4. Digital Output:
   The digital signal can acquire discrete values from [0 to 2^#Bits-1]. The following equation is implemented by using math blocks from Simulink library.
5. Zero-order Hold:
   This block has to implement to make the signal digital for the Simulink environment. The sampling time must be that of the sampling period.

2.2.2.3 Digital Filter

The digital filter is lowpass filter block from the Simulink library, with the cut off frequency 4 times the reference signal frequency. As the filter is lowpass, following bode plot describes the filter behavior.

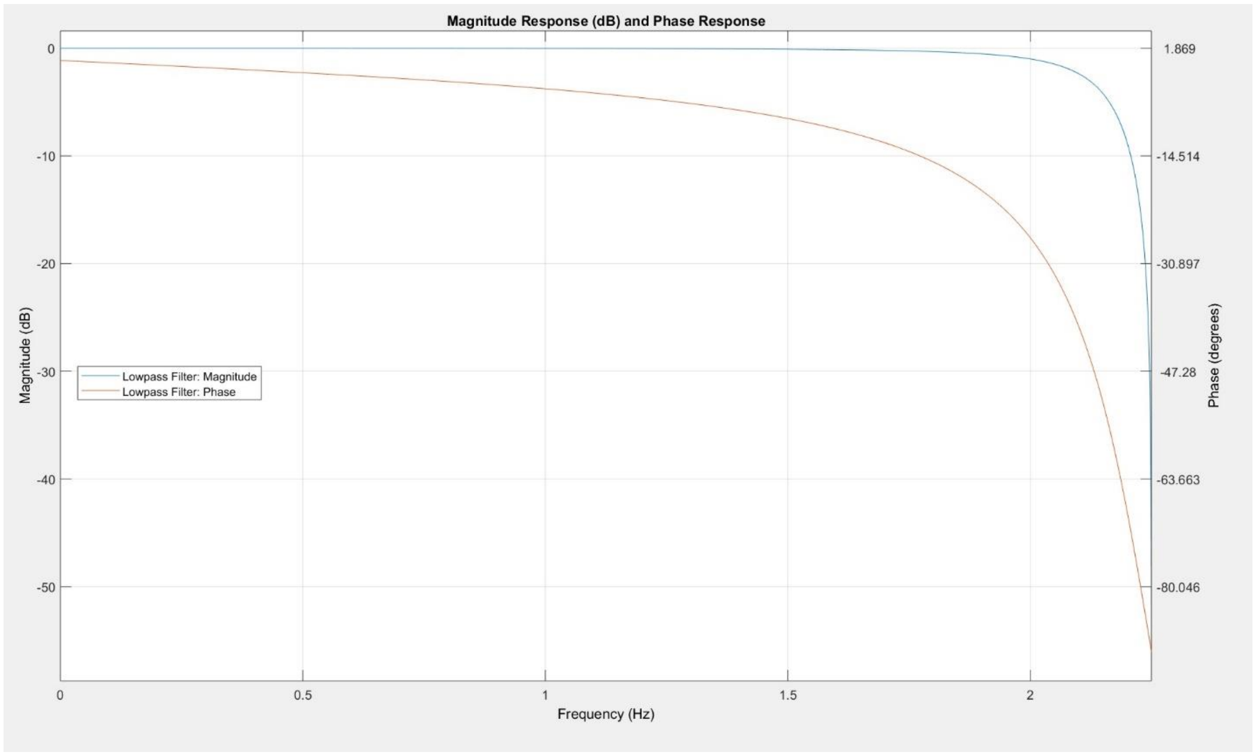


Figure 2.10: The Frequency response of digital filter of first order type IIR.

The digital filter type chosen is of type IIR for the following reasons:

1. As the microcontroller has a floating point capable processor, IIR is a better choice.
2. IIR requires less processing power as well as a low amount of RAM is sufficient for good results.

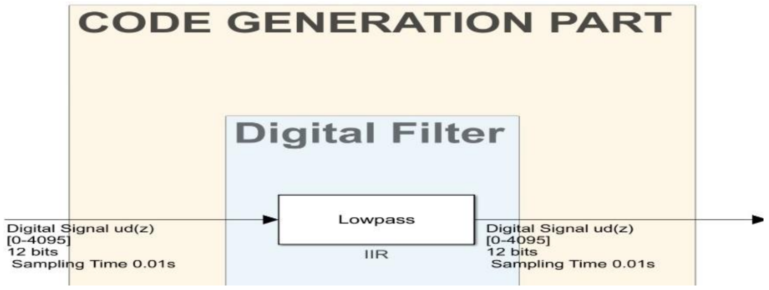


Figure 2.11: The digital filter.

Table 2.6: Signals at I/O of digital filter

Name	Symbol	Unit	Range
Digital Signal	ud(z)	-	[0 to 4095]
Filtered Digital Output	yd(z)	-	[0 to 4095]

2.2.2.4 Digital to analog converter block

The readings must be downscaled to board dynamics before the beginning of digital to analog process. The following equation gives the mathematical relation. The equations is implemented in Simulink by using the math blocks.

Table 2.7: Digital signal scaling to microcontroller dynamics before producing analog output

Name	Description	Symbol	UNIT	RANGE
Total Discrete Values	2^Bits-1	n		[0 to 2^Bits-1]
Full Scale Reading	Difference between maximum and minimum amplitude	-	V	-
Filtered Digital Output	-	yd(z)	-	[0 to 4095]
Scaled Filtered Digital Output	Filtered Digital Output scaled to range [0 to 3]	yds(z)	V	[0 to 3]

$$yds(z) = \frac{yd(z) \times FSR}{2^{Bits} - 1}$$

For generating PWM signal, PWM generator block is used which acquires duty cycle as the input and produces PWM output with the amplitude one, which is multiplied by the gain of 3.3 to produce a digital signal. For PWM the amplitude of the signal is scaled to width of the pulse, which is why Scaled Filtered Digital Output is divided by maximum value to produce outputs that are in the range [0 to 3].

In addition, PWM block requires 2 parameters, the carrier frequency(PWM) and the clock frequency. Both of these parameters are also used to determine the DAC resolution in bits.

$$DAC\_Bits = \log_2 \left[\frac{f_{clk}}{f_{pwm}}\right]$$

Table 2.8: PWM Characteristics

Symbol/Parameter	Name	UNIT	RANGE
ypwm	Output Analog signal after range limiter	V	[0 to 3.3]
ya(t)	Analog output signal	V	[0 to 3.3]
fclk	The PWM clock frequency	Hz	150 MHz max Simulated at 10e4
fpwm	The PWM carrier frequency	Hz	50Mhz max Simulated at 24
DAC_Bits/BitsPWM	The number of bits of the PWM	-	12

Given a number of bits of 12 and a clock frequency of 150 MHz (according to the board data sheet), the carrier frequency “fpwm” was set to 36 KHz. However, in the simulation fclk used is 10e4 Hz and fpwm is 24.4 Hz, these parameters result in a resolution of 12 bits.

Finally, the PWM signal must pass through a lowpass filter to generate the analog output signal ya(t).

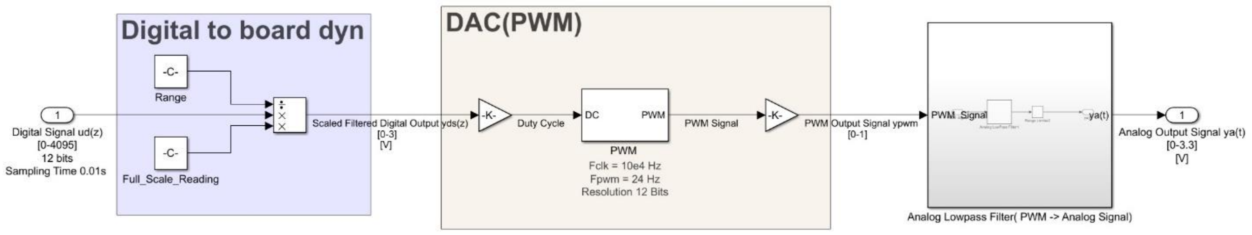


Figure 2.12: The DAC Simulink scheme.

2.2.2.5 Scaling output to Reference Signal Values

The last step requires to scale the signal back in the dynamics of the reference signal. The equation below the table identifies the equation used in scaling the signal.

Table 2.9: Signal Conditioning Block Characteristics

Name	Description	Symbol	Unit	Range
Output Signal	Analog output signal (Scaled)	y(t)	V	[-5 to 5]
Analog output signal	Analog output signal (Unscaled)	ya(t)	V	[0 to 3.3]
Scaling Coefficient	Scaling term, scales reference signal	m	-	-
Offset Term	Offset term	c	V	-
Analog_Input_Min	Microcontroller min input voltage (Parameter)	Am	V	0
Max_Signal	Reference Signal maximum value	MS	V	+5
Min_Signal	Reference Signal minimum value	Ms	V	-5
Digital_Output	Max digital output voltage from microcontroller (Parameter)	DO	V	3.3

The function of this block is given by the following equation:

$$y(t) = ya(t) \times m + c$$

2.2.3 System Parameter Selection and Behavior

For choosing the right parameters, i.e. number of bits and sampling time, the reference signal is digitized with 4-bits sampled at 0.1s and for comparison the digitization is done with 4-bits and 0.01 sampling frequency. From the picture below it's distinctly visible that high number of bits and lower sampling time translates to better signal representation. For our activity we have microcontroller that supports 12 bit ADC, which is compared with the previous parameters to conclude that indeed 12-bits contribute to a relatively better signal representation.

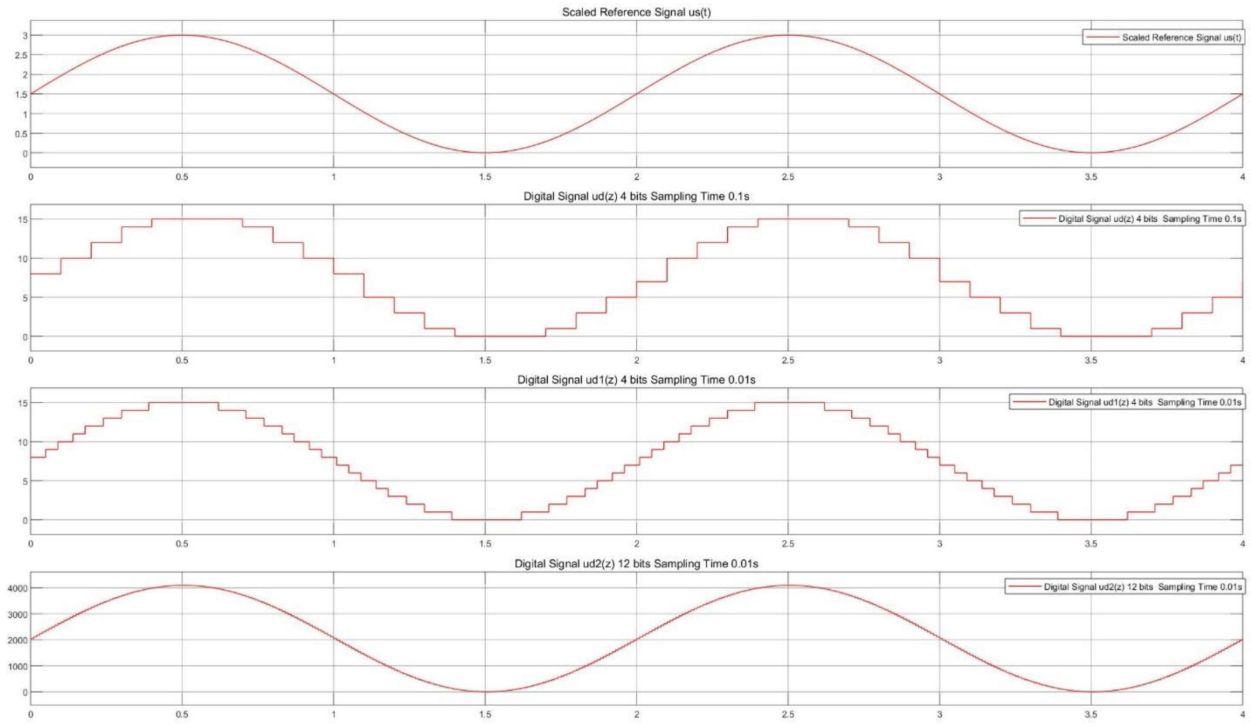
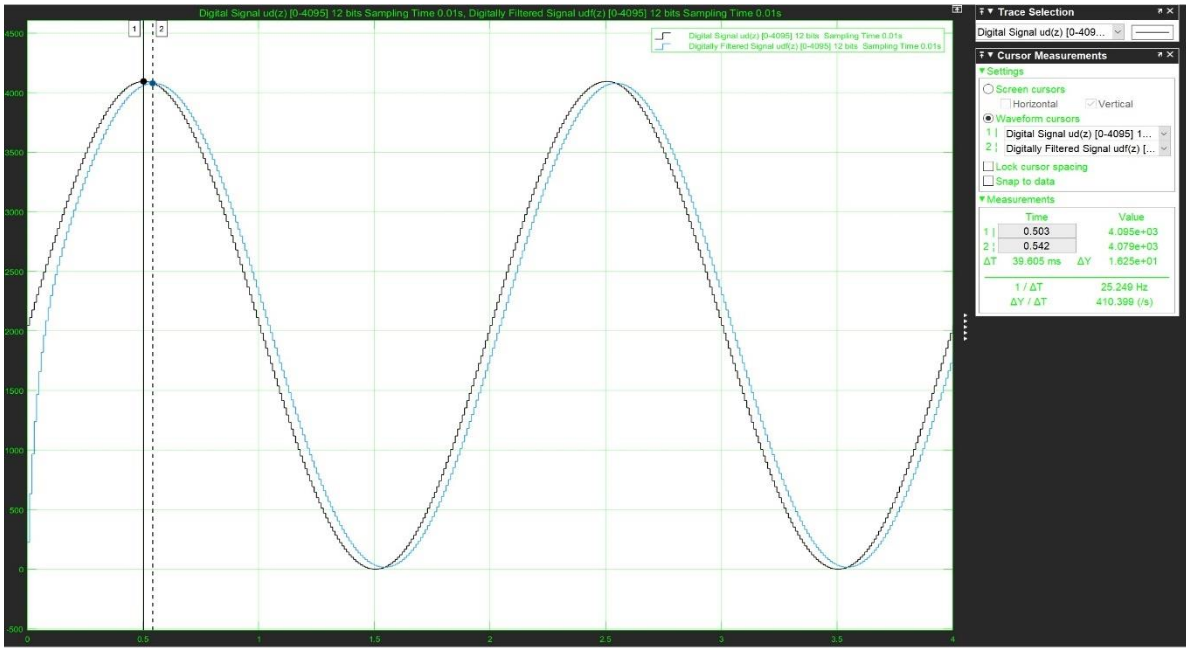


Figure 2.13: Parameter selection for number of bits and sampling time

$$F_s = F_0 \times 20$$

As for the sampling the frequency, it must be at least 20 times that of the fundamental frequency.

From the theory it is expected that in the bandpass region of the filter there will be a phase delay and a slight attenuation, which can be observed in the scope below.


Digital Signal ud(z) [0-4095] 12 bits Sampling Time 0.01s, Digitally Filtered Signal udf(z) [0-4095] 12 bits Sampling Time 0.01s
Digital Signal ud(z) [0-4095] 12 bits Sampling Time 0.01s
Digitally Filtered Signal udf(z) [0-4095] 12 bits Sampling Time 0.01s
Trace Selection
Cursor Measurements
Settings
Measurements

Trace Selection: Digital Signal ud(z) [0-409...

Cursor Measurements

Settings: ( ) Screen cursors — Horizontal, ✓ Vertical; (●) Waveform cursors

1 | Digital Signal ud(z) [0-4095] 1...

2 ¦ Digitally Filtered Signal udf(z) [...

☐ Lock cursor spacing

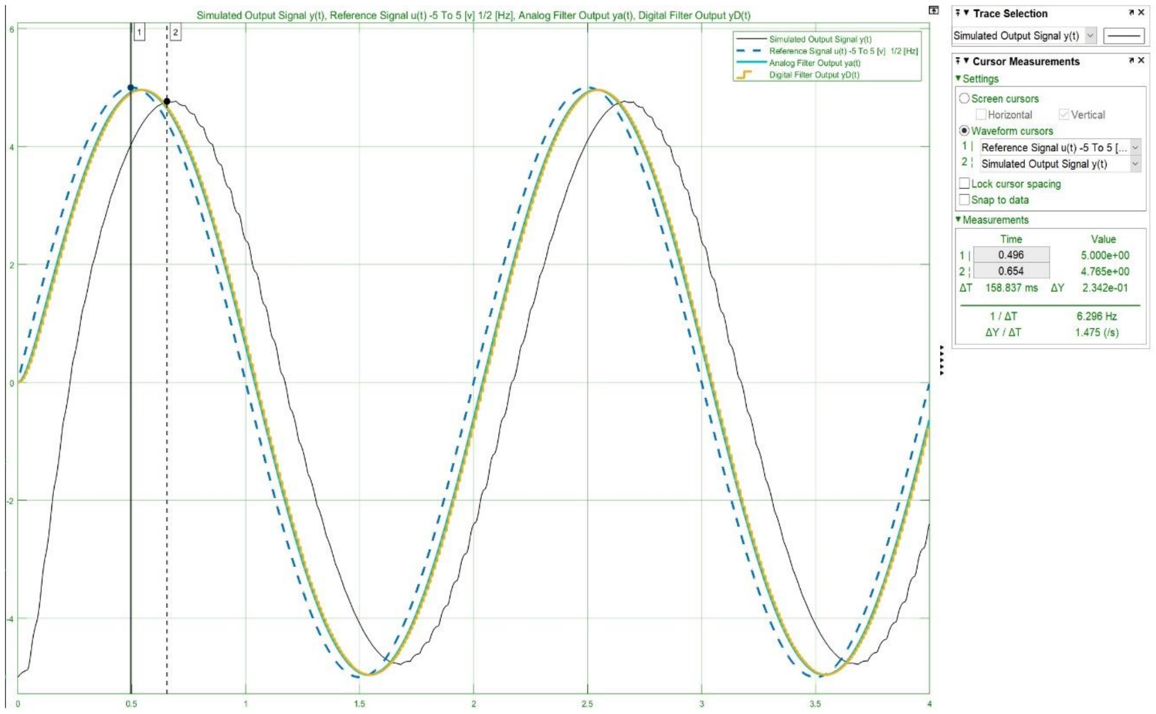
☐ Snap to data

	Time		Value
1 |	0.503		4.095e+03
2 ¦	0.542		4.079e+03
ΔT	39.605 ms	ΔY	1.625e+01
	1 / ΔT		25.249 Hz
	ΔY / ΔT		410.399 (/s)

Figure 2.14: The input and output relation between reference signal and analog/digital filter.

It can be observed that there is a delay of about 40ms between the reference signal (Dashed line in blue) and the output due to an analog filter (solid line in green (from Base model)). Moreover, there is no delay between the analog filter output and the digital filter output.

In the scope output below the output of the enhanced model and the reference model is discussed.


Simulated Output Signal y(t), Reference Signal u(t) -5 To 5 [v] 1/2 [Hz], Analog Filter Output ya(t), Digital Filter Output yD(t)
Simulated Output Signal y(t)
Reference Signal u(t) -5 To 5 [v] 1/2 [Hz]
Analog Filter Output ya(t)
Digital Filter Output yD(t)
Trace Selection
Cursor Measurements
Settings
Measurements

Trace Selection: Simulated Output Signal y(t)

Cursor Measurements

Settings: ( ) Screen cursors — Horizontal, ✓ Vertical; (●) Waveform cursors

1 | Reference Signal u(t) -5 To 5 [...

2 ¦ Simulated Output Signal y(t)

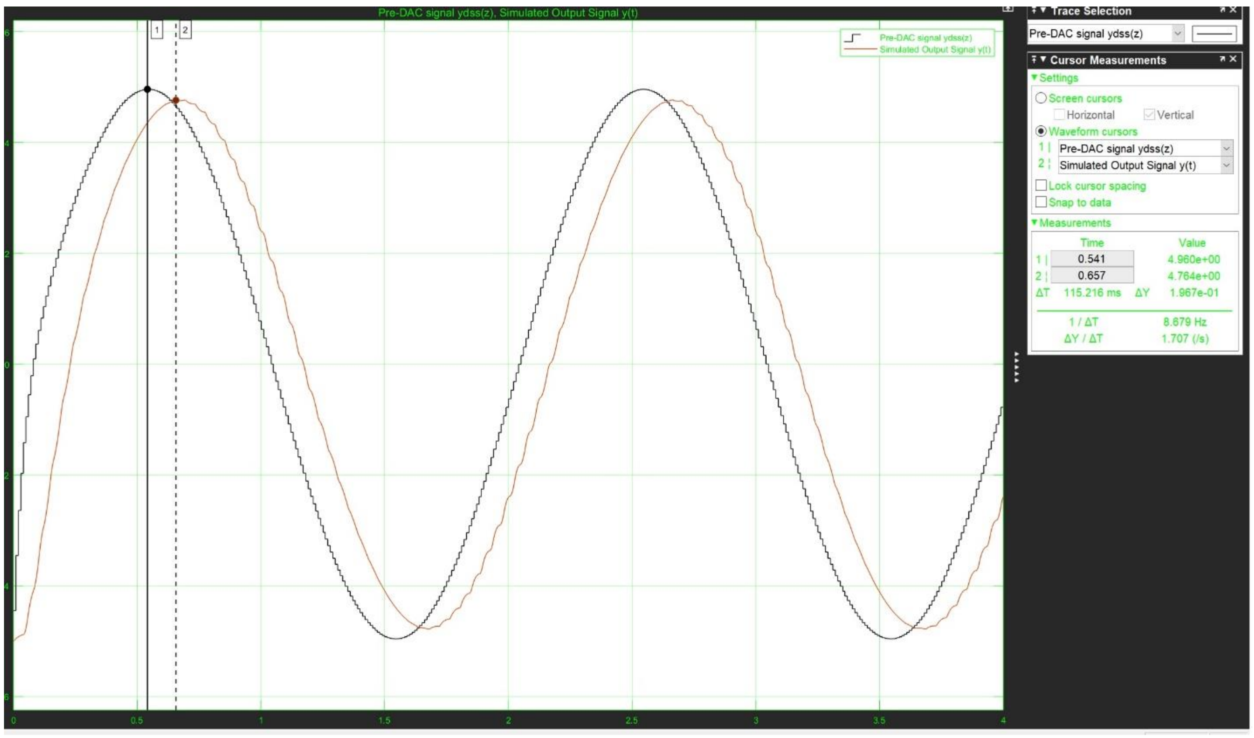
☐ Lock cursor spacing

☐ Snap to data

	Time		Value
1 |	0.496		5.000e+00
2 ¦	0.654		4.765e+00
ΔT	158.837 ms	ΔY	2.342e-01
	1 / ΔT		6.296 Hz
	ΔY / ΔT		1.475 (/s)

Figure 2.15: The input and output relation between reference signal and enhanced model output.

The delay between the reference signal and the enhanced model output is about 159 ms, which can be denoted from the picture above. The reason for a relatively large amount of delay is due to the analog filter that must be used at the output of digital PWM output to produce an Analog signal. The delay introduced by this lowpass filter is of about 116 ms, which is displayed in the figure below. The attenuation in the signal amplitude is approximately of the same magnitude as that of the output from the filters.



Pre-DAC signal ydss(z), Simulated Output Signal y(t)
Pre-DAC signal ydss(z)
Simulated Output Signal y(t)
1
2
0
0.5
1
1.5
2
2.5
3
3.5
4

Trace Selection

Pre-DAC signal ydss(z)

Cursor Measurements

Settings

( ) Screen cursors

☐ Horizontal ☑ Vertical

(●) Waveform cursors

1 | Pre-DAC signal ydss(z)

2 ¦ Simulated Output Signal y(t)

☐ Lock cursor spacing

☐ Snap to data

Measurements

	Time		Value
1	0.541		4.960e+00
2	0.657		4.764e+00
ΔT	115.216 ms	ΔY	1.967e-01
	1 / ΔT		8.679 Hz
	ΔY / ΔT		1.707 (/s)

Figure 2.16: The delay introduced by 2nd order lowpass filter, used for demodulating PWN signal.

2.3 Code generation model

2.3.1 Introduction And Objective

The model-based approach is carried out in Simulink for the microcontroller using hardware support package available for Simulink. In this stage, data acquisition Simulink blocks are used to acquire a signal via ADC, that is filtered with lowpass filter and then there is DAC block. For this activity, only the code generation part from enhanced model is chosen, as it contains the digital filter. After model competition the code is generated and deployed on target. It must be noted that the model between the I/O interface blocks is the same as that of the enhanced model, because it performs the core functionality of the task.

2.3.2 Description

The model setup requires to place an ADC block to acquire an analog signal and eCAP block to generate PWM output, both the blocks are obtained from the embedded coder support library. The sample time chosen is the same as that of the enhanced model.

Furthermore, the model is to be implanted in EXTERNAL mode to monitor the signal data through a scope block in Simulink. For this purpose, the signal acquired from ADC module is scaled to reference signal values in Simulink before connecting the signal to scope so that the signals are consistent. Similarly, a scope is connected after the digital filter to view the changes to signal in run time.

Lastly, the digital filter implemented is the same filter used for simulation in the enhanced model. The complete model is to be used for code generation and deployment on target.

2.3.3 Model Description

The microcontroller Simulink blocks are available by downloading hardware support package for C2000 microcontrollers.

Table 2.10: Signals description in code generation model.

Name	**Description**	**Symbol**	**Unit**	**Range**
Adc Output Signal	Digital Signal From Analog Pin A0	dy(z)	-	[0 to 4095]
Filtered Output Signal	Filtered Digital Signal From Analog Pin A0	u(z)	-	[0 to 4095]
ScaledSignalBoard	Signal scaled to board dynamics	s(z)	V	[0-3]
Values	All possible values, 2^Bits-1	vu	-	-
Scaled input	Scaled in Reference Signal Dynamics	y(z)	V	[-5 to 5]

The ADC block is chosen for C2833x microprocessor. The sampling time parameter is chosen the same as the one used in Enhanced model ADC i.e. 0.01s and the data type chosen is uint16. The obtained signal is a digital signal which must be scaled to represent the reference signal from the signal generator. The scaling process is done by first scaling the digital values to board voltage dynamics and then to reference signal dynamics.

Scaling the signal to board dynamics is done by using math operations that are given by the following equation.

$$s(z) = \frac{u(z) \times FSR}{vu}$$

The digital signal is scaled in reference signal values by the following equation.

$$y(z) = u(z) \times m + c$$

Two scopes are used to view the reference signal that is acquired by ADC. Scopes are used to monitor the effects of the digital filter.

The input data-type for the digital filter must be double precision, single precision, signed integer, which is why data type conversion block must be used. For the code generation model the block that is the same as the enhanced model is the digital filter. Before converting the signal to PWM, it's converted to duty cycle percentage. The relation is given below. Finally, the PWM period is set to 0.04 which is the same period used in enhanced model.

$$Gain = \frac{100}{2^{bits} - 1}$$

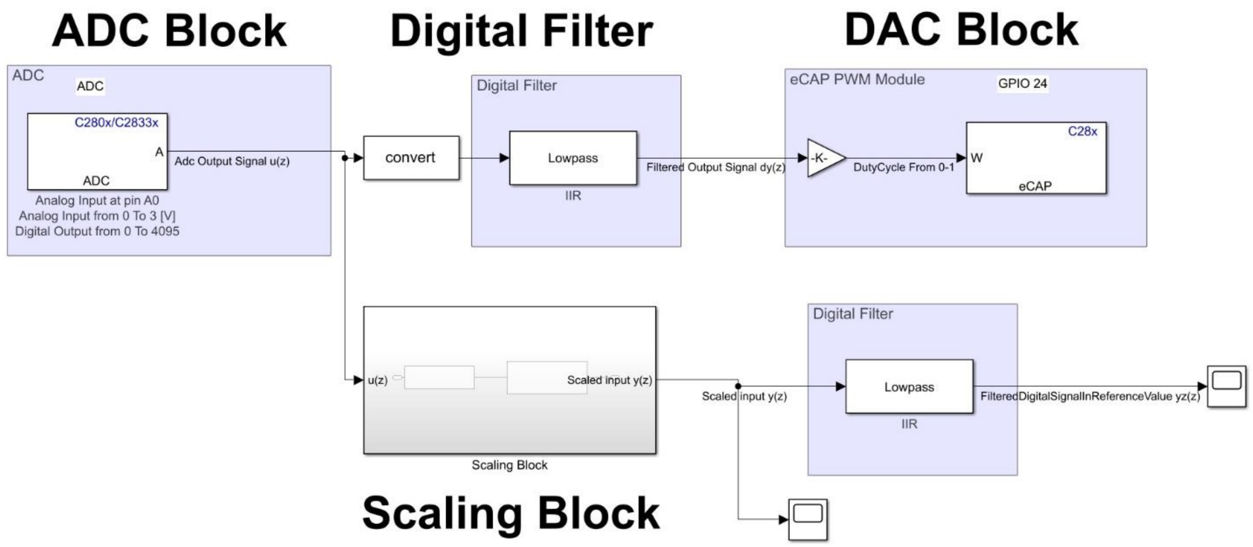
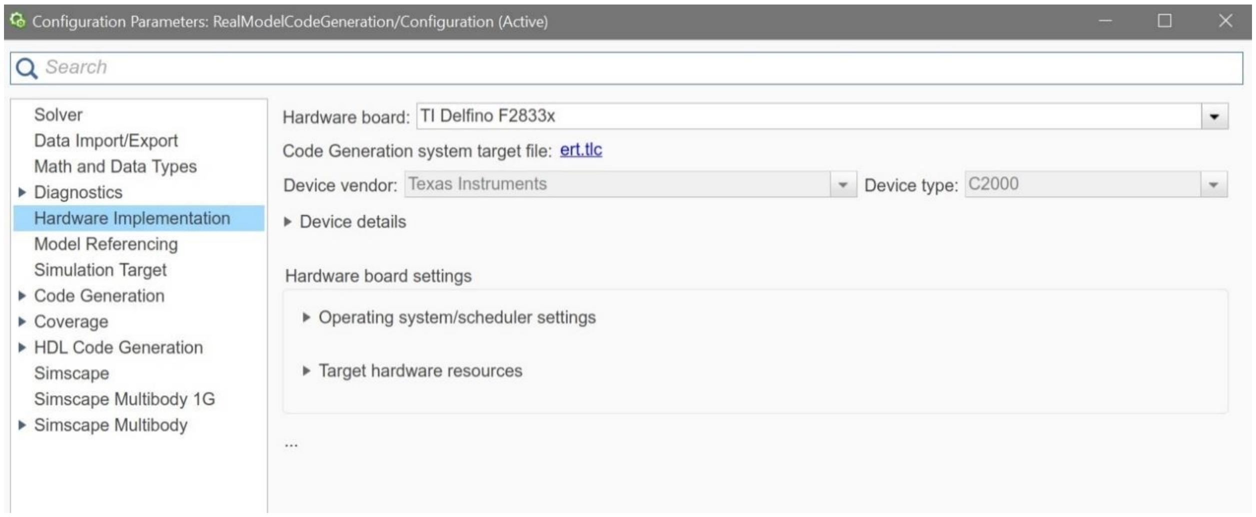


Figure 2.17: The Simulink scheme of the code generation model.

2.3.4 Code Generation Steps & Code Mapping

For the code generation, Model Configuration Parameters must be selected and in the Hardware Implementation tab TI Delfino F2833x must be selected as indicated in the picture below.

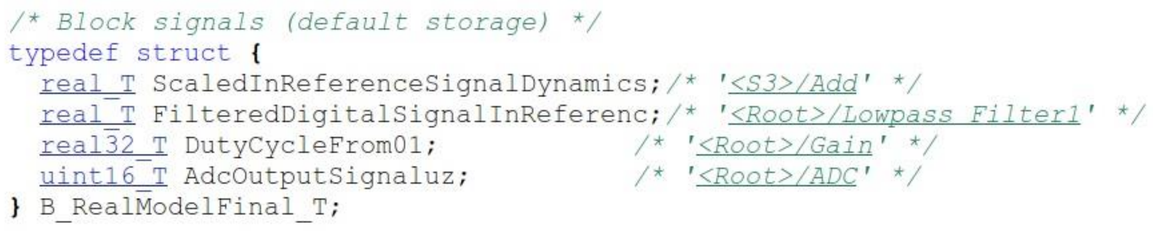


Hardware board: TI Delfino F2833x
Code Generation system target file: ert.tlc
Device vendor: Texas Instruments
Device type: C2000

Figure 2.18: Code Generation settings.

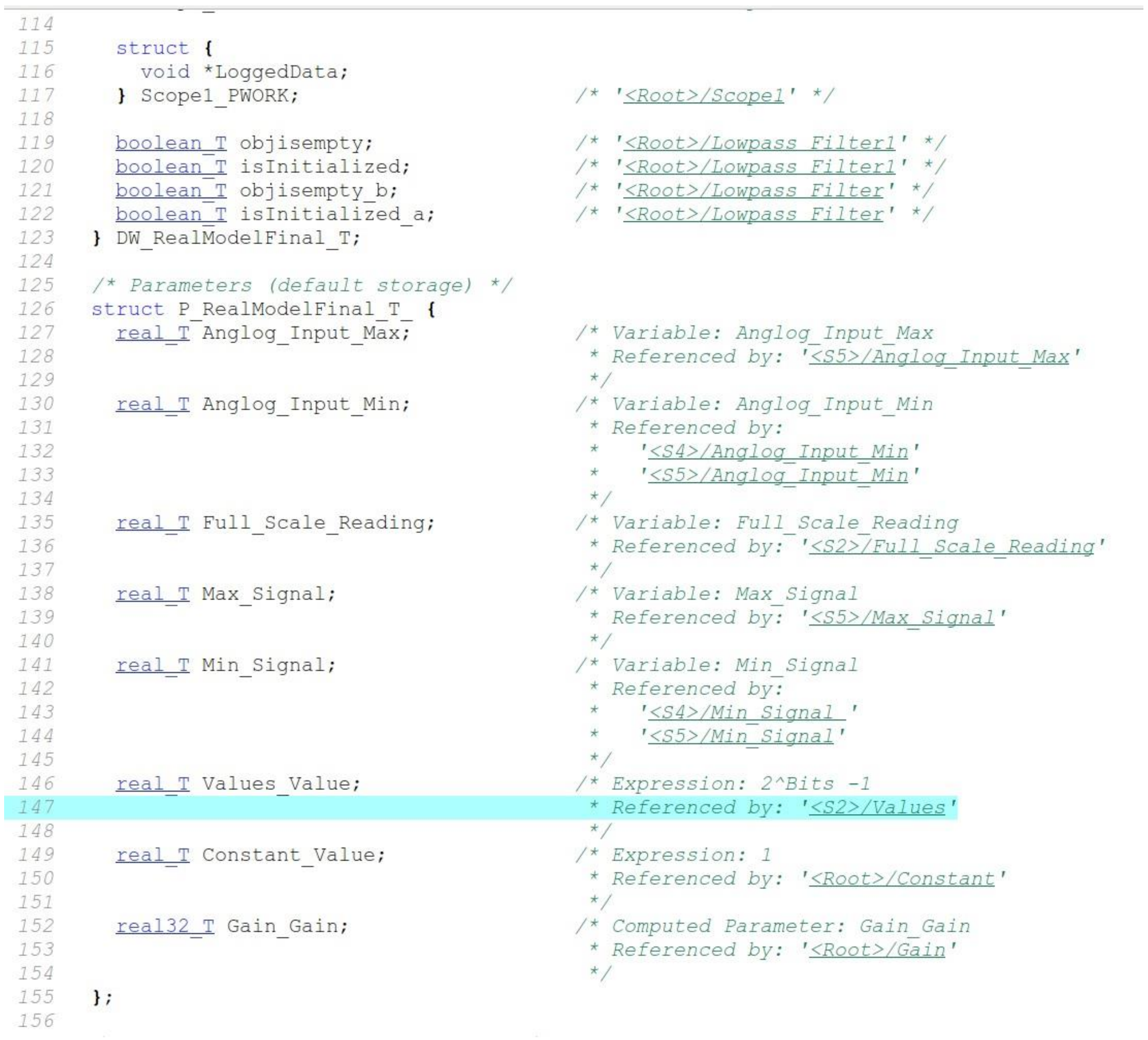
Code can be then deployed to board by clicking Deploy to hardware.

The code generation report clearly provides a mapping between the Simulink blocks, Signals and the generated code.



```
/* Block signals (default storage) */
typedef struct {
  real_T ScaledInReferenceSignalDynamics;/* '<S3>/Add' */
  real_T FilteredDigitalSignalInReferenc;/* '<Root>/Lowpass Filter1' */
  real32_T DutyCycleFrom01;            /* '<Root>/Gain' */
  uint16_T AdcOutputSignaluz;          /* '<Root>/ADC' */
} B_RealModelFinal_T;
```

Figure 2.19: Signals defined in the header file.



```

    struct {
      void *LoggedData;
    } Scope1_PWORK;                        /* '<Root>/Scope1' */

    boolean_T objisempty;                  /* '<Root>/Lowpass Filter1' */
    boolean_T isInitialized;               /* '<Root>/Lowpass Filter1' */
    boolean_T objisempty_b;                /* '<Root>/Lowpass Filter' */
    boolean_T isInitialized_a;             /* '<Root>/Lowpass Filter' */
  } DW_RealModelFinal_T;

  /* Parameters (default storage) */
  struct P_RealModelFinal_T_ {
    real_T Anglog_Input_Max;               /* Variable: Anglog_Input_Max
                                            * Referenced by: '<S5>/Anglog_Input_Max'
                                            */
    real_T Anglog_Input_Min;               /* Variable: Anglog_Input_Min
                                            * Referenced by:
                                            *   '<S4>/Anglog_Input_Min'
                                            *   '<S5>/Anglog_Input_Min'
                                            */
    real_T Full_Scale_Reading;             /* Variable: Full_Scale_Reading
                                            * Referenced by: '<S2>/Full_Scale_Reading'
                                            */
    real_T Max_Signal;                     /* Variable: Max_Signal
                                            * Referenced by: '<S5>/Max_Signal'
                                            */
    real_T Min_Signal;                     /* Variable: Min_Signal
                                            * Referenced by:
                                            *   '<S4>/Min_Signal_'
                                            *   '<S5>/Min_Signal'
                                            */
    real_T Values_Value;                   /* Expression: 2^Bits -1
                                            * Referenced by: '<S2>/Values'
                                            */
    real_T Constant_Value;                 /* Expression: 1
                                            * Referenced by: '<Root>/Constant'
                                            */
    real32_T Gain_Gain;                    /* Computed Parameter: Gain_Gain
                                            * Referenced by: '<Root>/Gain'
                                            */
  };

```

Figure 2.20: Constants defined in the header file.

All the blocks used to assign constants and signals are denoted as parameters in the header file generated from code generation. Moreover, from the Model configuration parameters, under the code generation tab in the report section, generate model web view can be selected to provide a direct mapping in graphical form with the Simulink blocks and generated code.

2.3.5 System Behavior and Output signal

Following the code generation and deployment, the acquired signals can be monitored by running the Simulink model in external mode, and they can be compared with the reference signal from signal generator by an Oscilloscope.

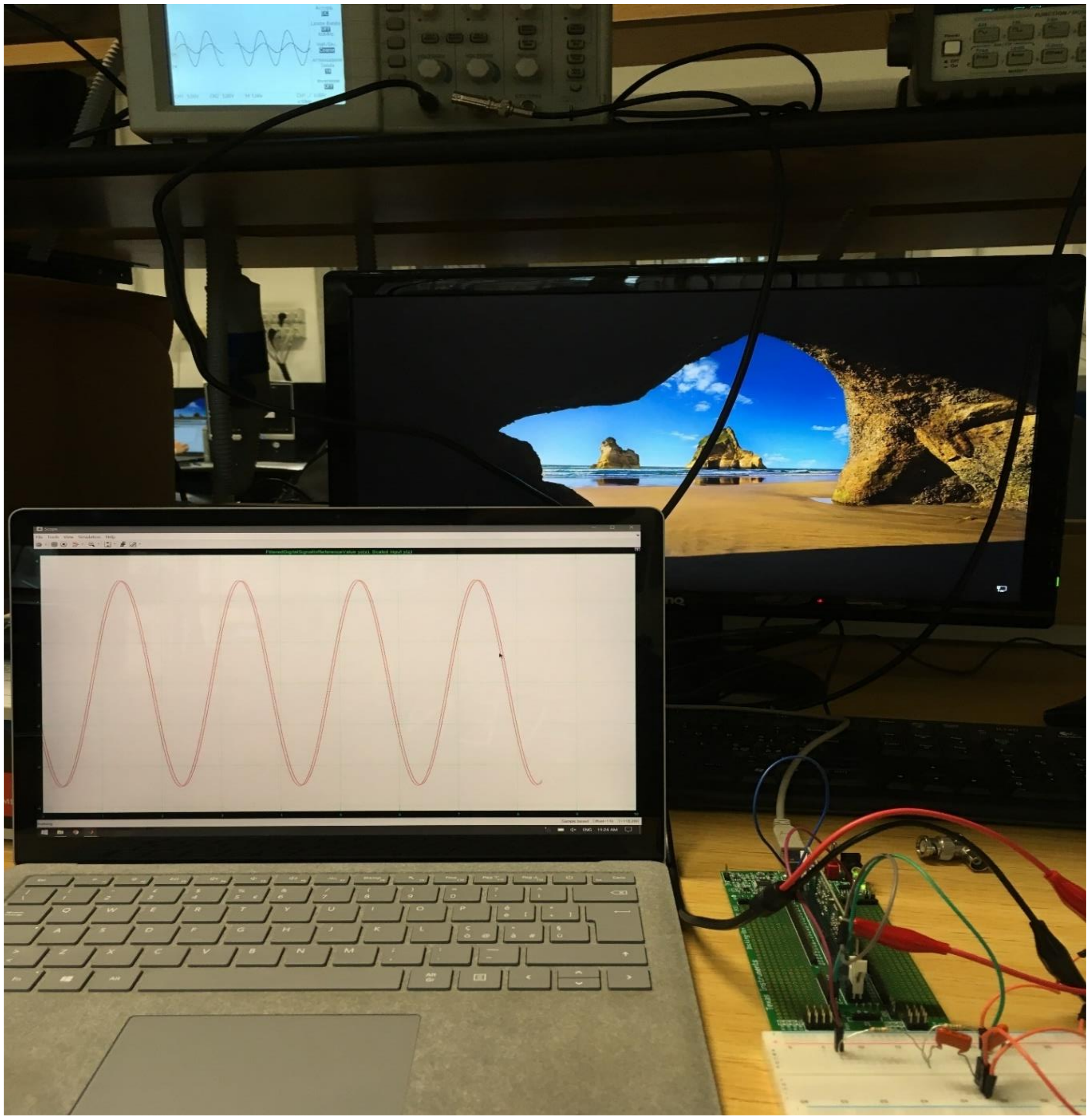
2.4 Testing

This phase is carried out in a laboratory to observe the code generated model's performance and to collect data, that is used for iterating the design for enhanced model.

2.4.1 Setup

The testing setup is composed of the following:

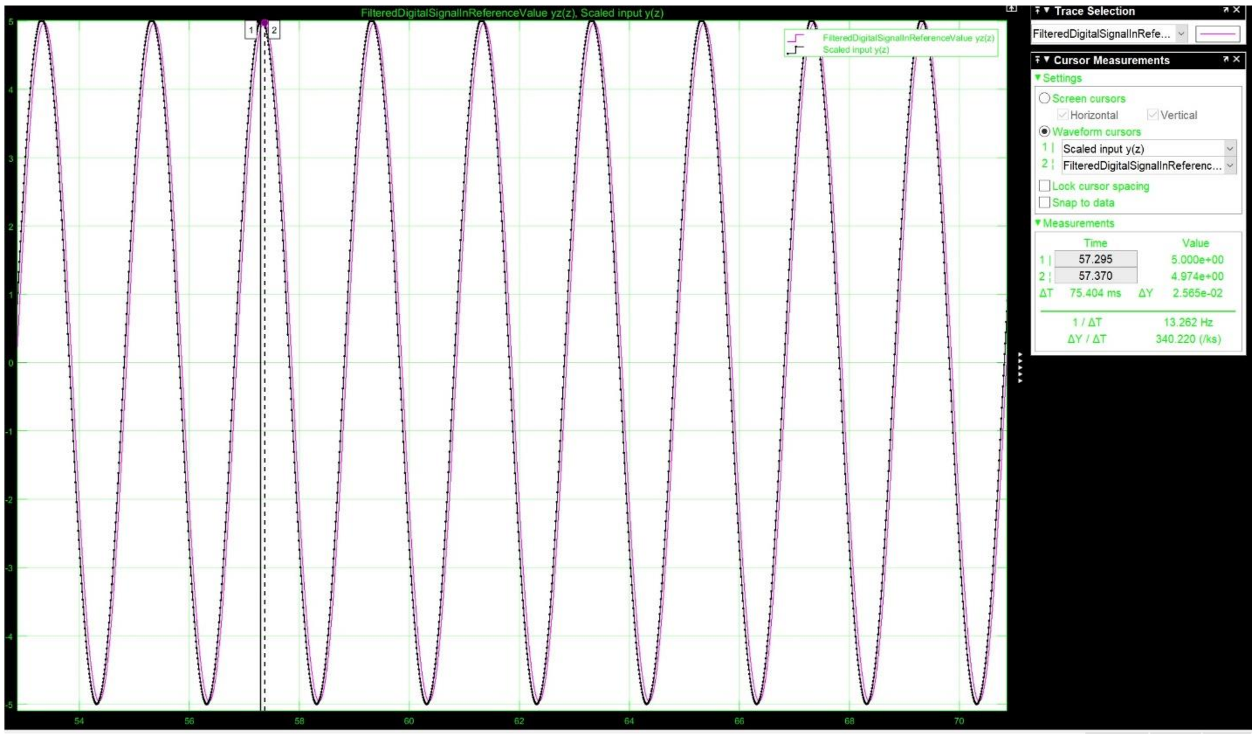
1. Texas Instruments C2000 Microcontroller F28335 Delfino Experimental Kit.
2. A breadboard, few jumper wires, 2 resistors(1Mohm) and 2 capacitors(68nF) to convert PWM to an analog signal.
3. A signal generator and an Oscilloscope, to generate and observe the signal respectively.
4. A windows pc to run the generated code in external mode to observe.



5. Figure 2.21: The experimental Setup.

2.4.2 Observed Results

The results obtained from the external model is show below.


FilteredDigitalSignalInReferenceValue yz(z), Scaled input y(z)
FilteredDigitalSignalInReferenceValue yz(z)
Scaled input y(z)
5
4
3
2
1
0
-1
-2
-3
-4
-5
54
56
58
60
62
64
66
68
70
1
2

Trace Selection
FilteredDigitalSignalInRefe...

Cursor Measurements
Settings
Screen cursors
Horizontal Vertical
Waveform cursors
1 Scaled input y(z)
2 FilteredDigitalSignalInReferenc...
Lock cursor spacing
Snap to data
Measurements

	Time		Value
1	57.295		5.000e+00
2	57.370		4.974e+00
ΔT	75.404 ms	ΔY	2.565e-02
	1 / ΔT		13.262 Hz
	ΔY / ΔT		340.220 (/ks)

Figure 2.22: The experimental scope output in Simulink external mode.

The acquired signal from ADC is in blue and the digitally filtered signal is depicted in purple. The delay between the two is 75.404ms.

Moreover, the results observed on the oscilloscope are discussed below.

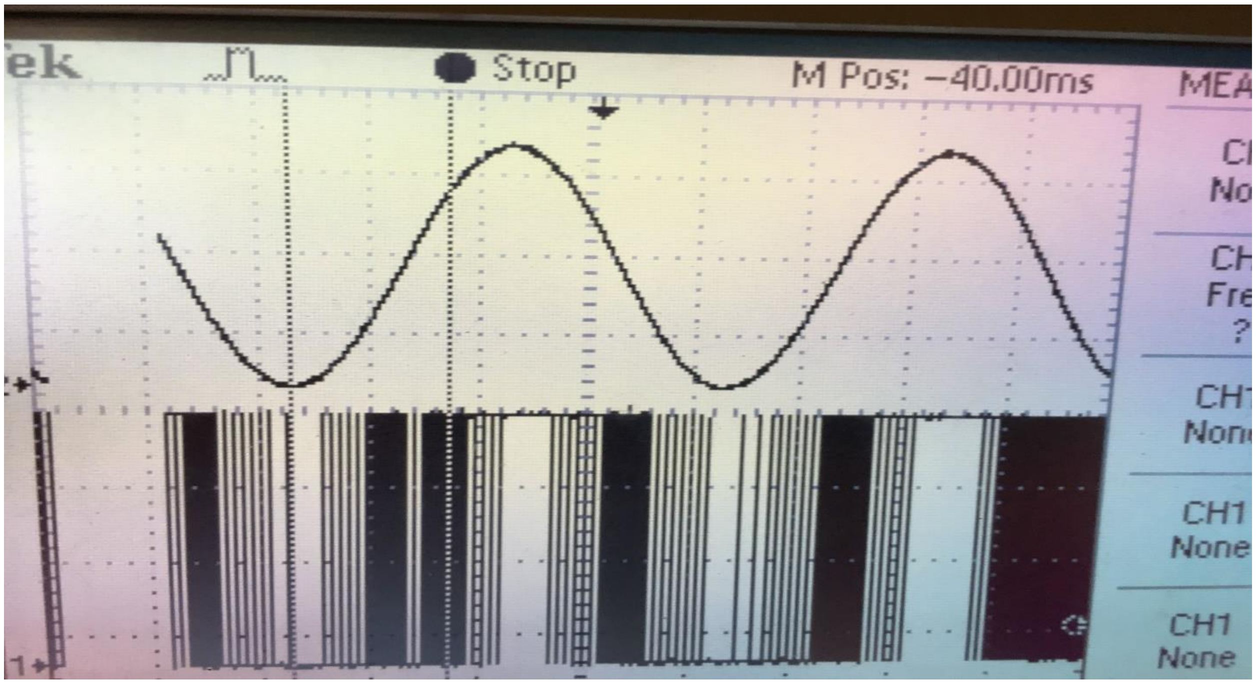


Figure 2.23: The oscilloscope output of reference signal and the PWM output signal.

The signal in the upper part of the figure is the reference signal from the generator and PWM signal is in the lower part which is produced from the microcontroller after the implementation of the generated code.

To know the delay between the two a passive 2nd order lowpass filter (as first order has a lot of ripples) is used to demodulated the signal.

$$Cutoff\ frequency = \frac{1}{2\pi RC}$$

I select Cutoff frequency to be 5 times that of the generated signal, and by imposing R to 1MΩ C is about 64nF. After passing the signal through a lowpass filter the result is given in the picture below.

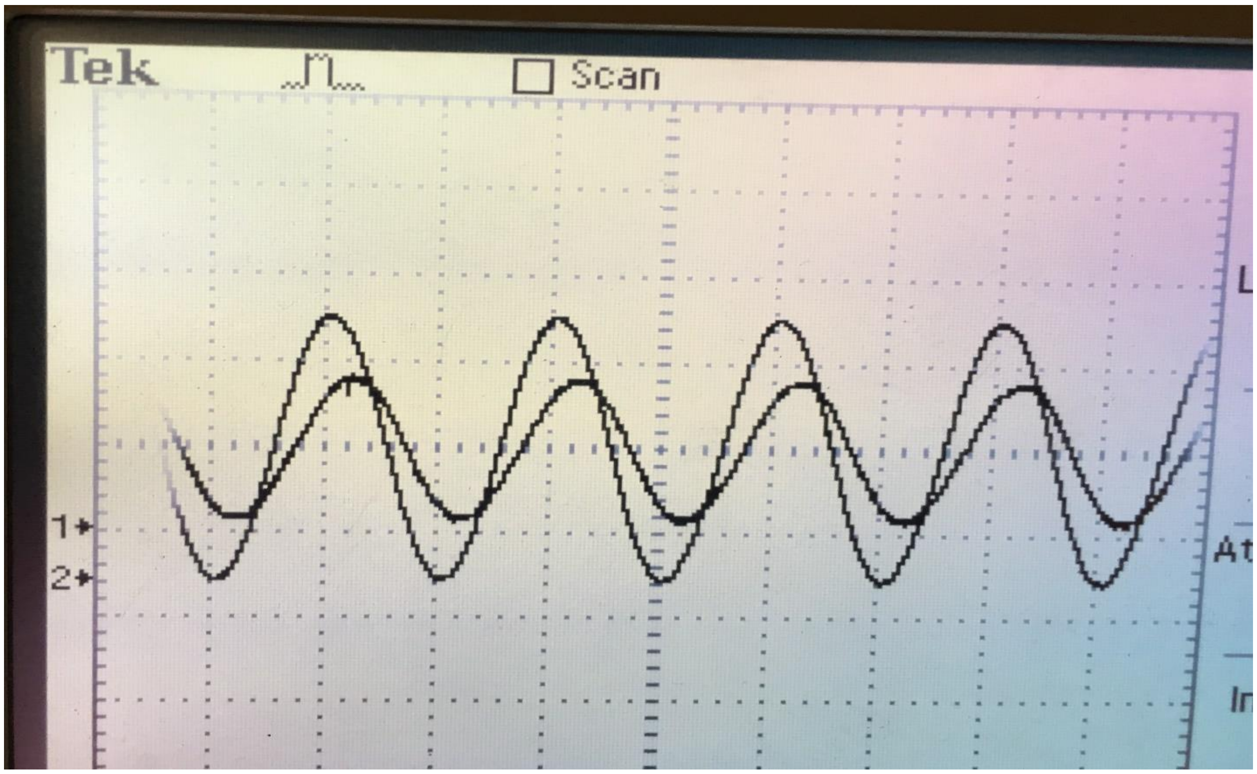


Figure 2.24: The oscilloscope output of reference signal and the demodulated output signal.

The delay between the reference signal and output signal is about 180ms.

2.4.3 Comparing the results

Table 2.11: Table of observed Delays.

Name of Signals to measure delay	Delay[ms]
Simulated Output	159
Code generation Output	180
Simulated lowpass filter delay	40
Digital lowpass filter delay in external mode	75.404

It can be observed that there are more delays observed in the real experiments when compared with the simulations. Also, when Simulink starts the simulation in external mode, the output on the Simulink scope is delayed due the connection setup time between the Simulink and the microcontroller.

2.5 Model Validation and Calibration

2.5.1 Delay Analysis in simulation and testing

Based on the experimental data from the testing phase, the enhanced model is calibrated to match the performance of the microcontroller.

The main reasons for delay in the simulation are digital lowpass filter (40 ms) and the analog low pass filter (116 ms), which adds up to the total in the simulation output. However, in the simulation the delays introduced don't consider the time needed for ADC and DAC conversion. This issue is discussed below.

In the testing to determine the delay acquired by the microcontroller to convert the reference signal from analog to digital PWM, and then to demodulated signal through 2nd order lowpass filter takes about 140 ms, which is measured by an oscilloscope.

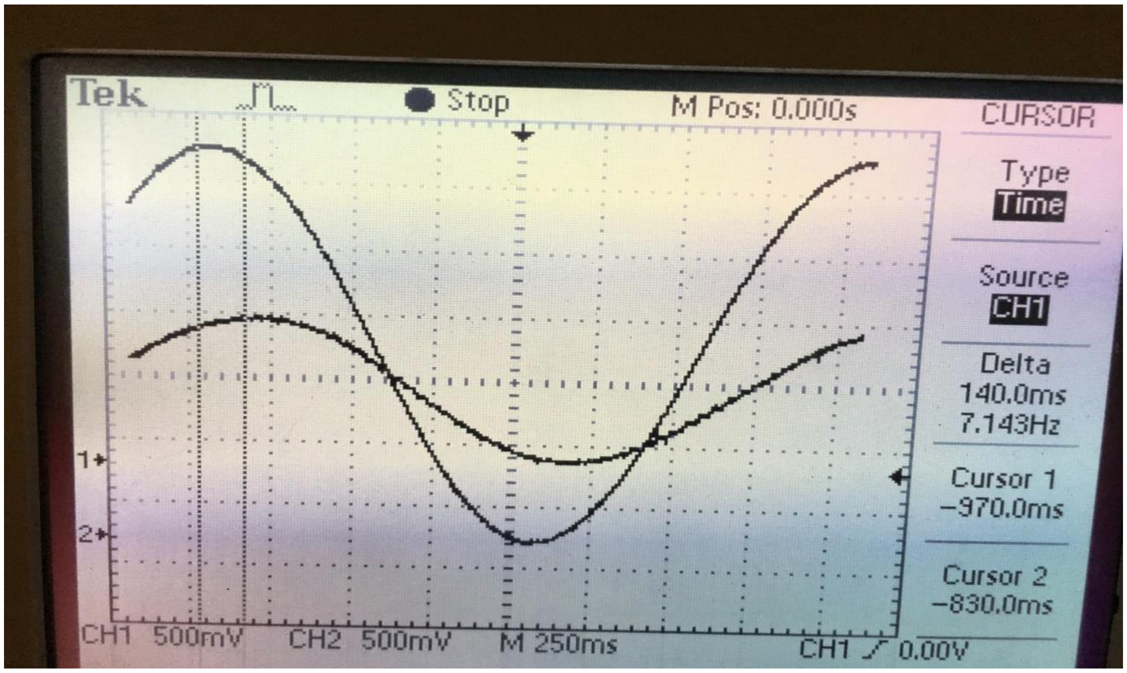


Figure 2.25: The time acquired for analog to digital conversion.

It must be noted that 140 ms is the time acquired by microcontroller traversal and analog lowpass filter. To know the microcontroller traversal time, testing for time delay introduced by the lowpass filter is done by measuring delay between the reference signal from signal generator and analog lowpass filter output signal on the oscilloscope.

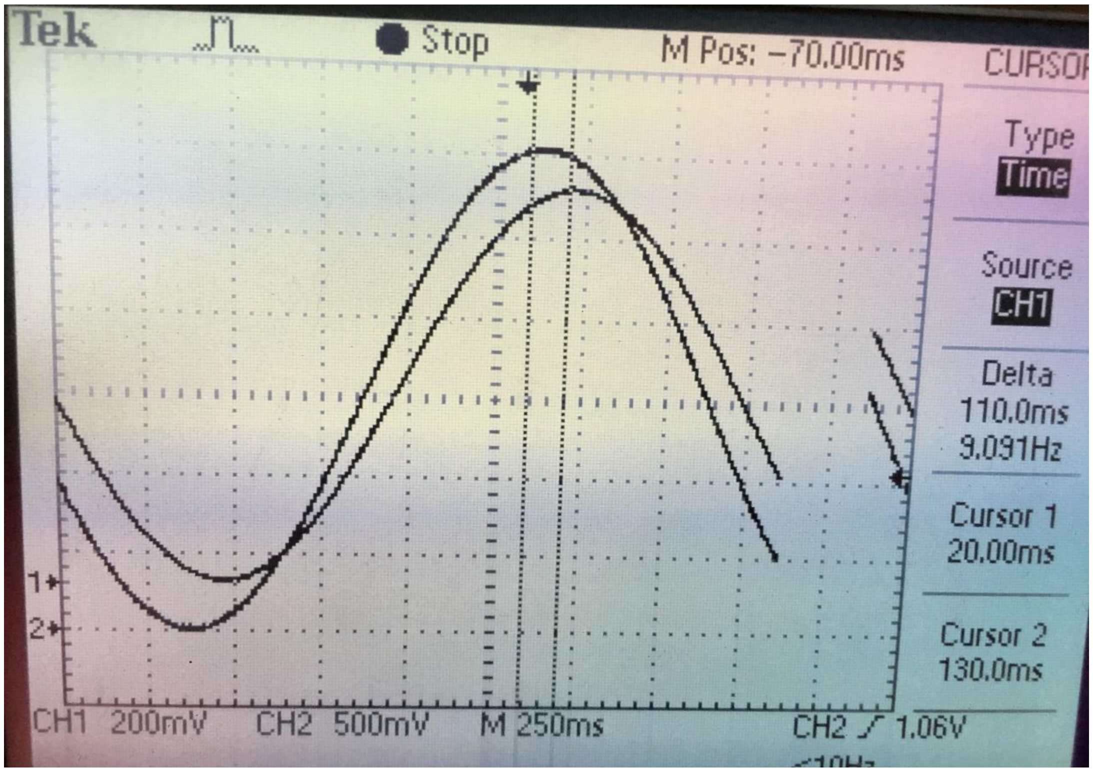


Figure 2.26: The delay produced by analog lowpass filter.

The delay by analog lowpass filter is 110 ms. Hence, the microcontroller traversal delay is the difference, which is 30 ms. From previous testing of code generation model, the total delay was 180 ms. To sum up, the ADC + DAC delay is about 30 ms.

2.5.2 Calibration

There must be a delay of about 30 ms introduced in the enhanced model to have a calibrated valid model that considers delay of ADC-DAC found in the microcontroller. The delay is introduced by a Simulink block.

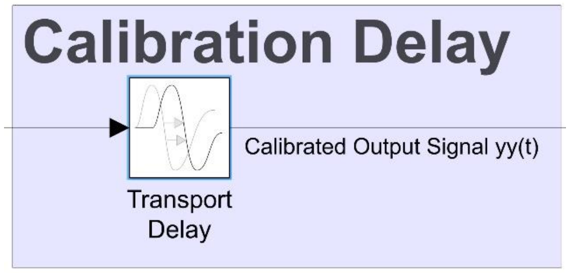


Figure 2.27: The Simulink delay block to maintain the simulation model.

Chapter 3-VMU And Plant Modelling Scheme

3.0 Introduction

This chapter introduces the model-based scheme of vehicle management unit and the plant to be controlled. The scheme is going to be used for system simulation and code generation for vehicle management unit, that is to be deployed on the target.

The models will be discussed in three models, namely:

1. System Layout Concept Model.
2. System Layout Enhanced Model.
3. Vehicle Management Unit Layout Model for Code Generation.

The model is based on the framework outlined in the introduction of the thesis.

3.1 System Layout Concept Model

3.1.1 Objective

The objective of the system layout concept model is to model the main blocks in the system.

3.1.2 Description

The model is composed of 2 subsystems, controller and plant. However, there is a I/O boundary between them which is discussed in the next topic. The following table mentions the signal connections between the blocks.

Table 3.1: The description of Signals of plant in concept model

Name	Description	Symbol
Disturbances	Disturbance Signals input to plant	dt)
States	State outputs from plant	x(t)
Commands	Commands input from controller to plant	u(t)
Measurements	Plant output values	y(t)

Table 3.2: The description of Signals of Controller in concept model

Name	Description	Symbol
References	Reference Signals input to Controller	rt)
Estimated States	Estimated States output from Controller	xe(t)
Commands	Commands output from controller to plant	u(t)
Measurements	Plant output values as input to Controller	y(t)

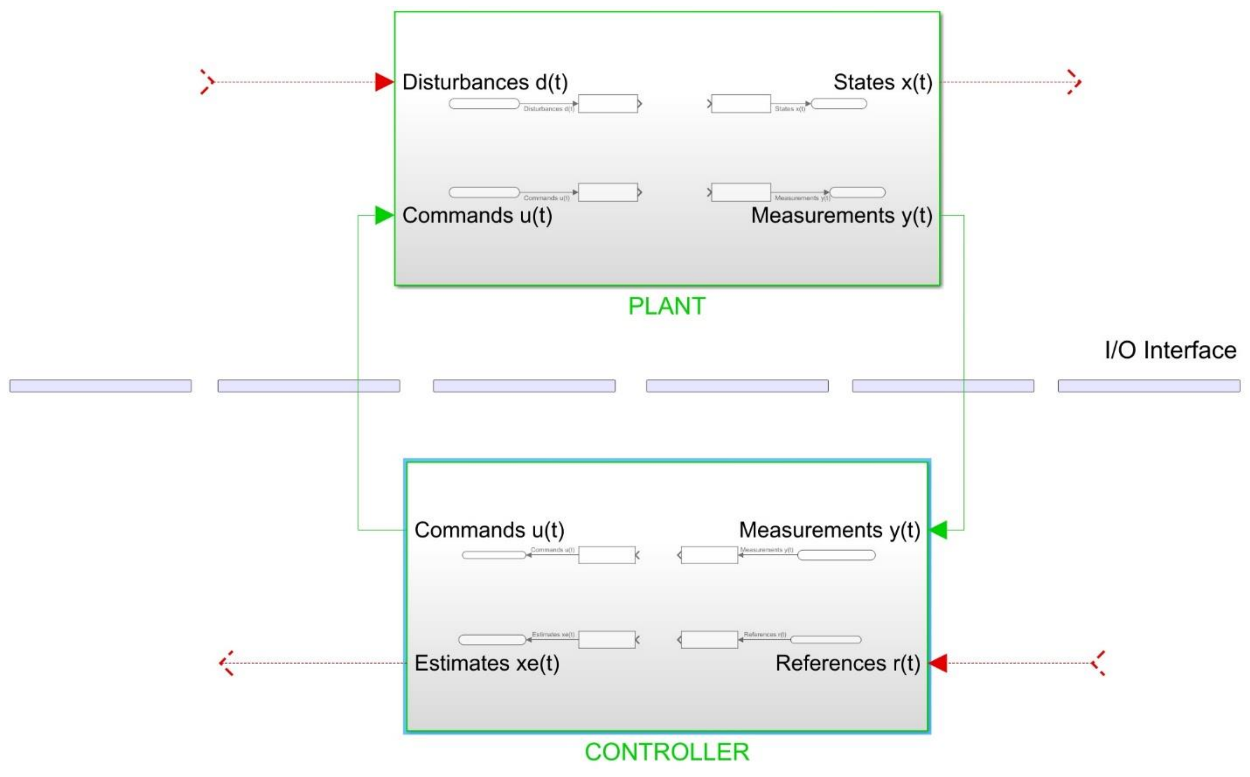


Figure 3.1: The Simulink Scheme for the System Layout Concept Model.

3.2 System Layout Enhanced Model

3.2.1 Objective

The objective of this model is to represent the I/O boundary in a modelled scheme. The entire scheme is enhanced to simulate the real system.

3.2.2 Description

This scheme adds I/O blocks, i.e. ADC and DAC blocks, these are the same blocks which are described in the chapter one. It should also be noted that the controller operates in the discrete domain and the plant in the continuous domain. The following table mentions the signal connections between the blocks.

Table 3.3: The description of Signals of plant in enhanced model

Name	**Description**	**Symbol**
Disturbances	Disturbance Signals input to plant	dt)
States	State outputs from plant	x(t)
Commands	Analog Commands input from DAC to plant	ua(t)
Measurements	Analog Plant output values	ya(t)

Table 3.4: The description of Signals of controller in enhanced model

Name	**Description**	**Symbol**
References	Digital Reference Signals input to Controller from HMI	rd(z)
Estimated States	Digital Estimated States output from Controller to HMI	xed(z)
Commands	Digital Commands output from controller to DAC to plant	ud(z)
Measurements	Digital Plant output values from ADC to Controller	yd(z)

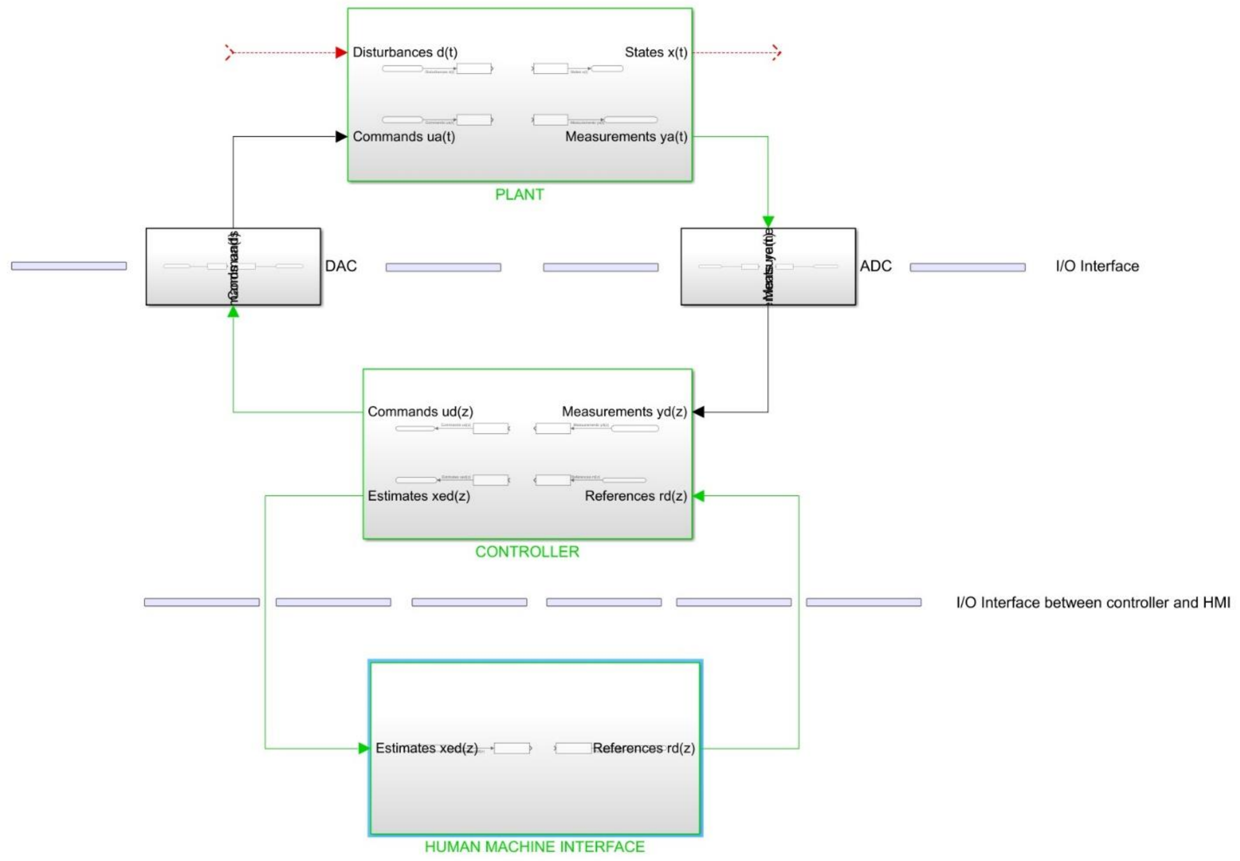


Figure 3.2: The Simulink Scheme for the System Layout Enhanced Model

Also, it must be realized that there is human machine interface which enables real-time monitoring of signals and modifications to parameters.

3.3 Vehicle Management Unit Layout Model For Code Generation

3.3.1 Objective

This section deals with modelling VMU, which consists of 7 subsystems. The purpose is to model the VMU for simulation and then excluding the I/O boundary of the VMU from the model to generate code for the remaining part, which will be deployed on the target.

3.3.2 Description

The VMU scheme comprises of VMU input and output blocks, these blocks are only included for simulation of input and output signal of VMU. The Simulink model compromises of 7 blocks of which 4 are selected for which code is generated, the rest are required for simulation.

1. VMU Input:
   This block models the analog input of the VMU. This input model is composed of a signal source and an ADC, which uses the same scheme addressed in chapter one. In other words this is the hardware modelling, as we are going to proceed with code generation for our system this part need not included.

2. Signal Preprocessing:
   At this stage, the signal is in digital form, however it must be scaled in the reference signal values. The scaling Simulink scheme is the same one addressed in the chapter one, nonetheless the parameters must be chosen as required.

3. Controller:
   The discrete controller block composes of suitable control strategies that meet the system requirements.

4. Signal Postprocessing:
   Finally, the controller output signals must be conditioned back to DAC specifications.

5. Clock Counter:
   The clock counter is used to simulate the internal clock of the target.

6. Interrupt Routines:
   The interrupt routines are confined to this block to configure the embedded system to respect the task priority and deadline requirements.

7. VMU Output:
   This block simulates the DAC hardware behavior, which is used for simulation but not used for code generation.

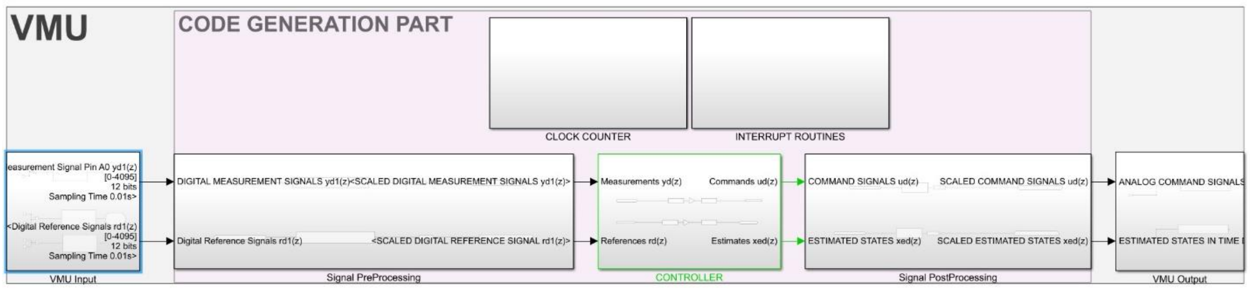

VMU
CODE GENERATION PART
CLOCK COUNTER
INTERRUPT ROUTINES
easurement Signal Pin A0 yd1(z) [0-4095] 12 bits Sampling Time 0.01s>
<Digital Reference Signals rd1(z) [0-4095] 12 bits Sampling Time 0.01s>
VMU Input
DIGITAL MEASUREMENT SIGNALS yd1(z)
<SCALED DIGITAL MEASUREMENT SIGNALS yd1(z)>
Digital Reference Signals rd1(z)
<SCALED DIGITAL REFERENCE SIGNAL rd1(z)>
Signal PreProcessing
Measurements yd(z)
Commands ud(z)
References rd(z)
Estimates xed(z)
CONTROLLER
COMMAND SIGNALS ud(z)
SCALED COMMAND SIGNALS ud(z)
ESTIMATED STATES xed(z)
SCALED ESTIMATED STATES xed(z)
Signal PostProcessing
ANALOG COMMAND SIGNALS
ESTIMATED STATES IN TIME
VMU Output

Figure 3.3: The Simulink Scheme for the VMU Model.

Chapter 4-Test Bench and VMU Implementation

4.0 INTRODUCTION

The Test Bench is setup to monitor the motor under test, which is countered by the bench motor. The aim is to develop a model-based control strategy for VMU in Matlab/Simulink, which controls both the motors, and to monitor the key variables, such as current, shaft speed, bench motor torque and test motor torque. Also, the development cycle includes automatic code generation. Both the motors are torque controlled, having same speed.

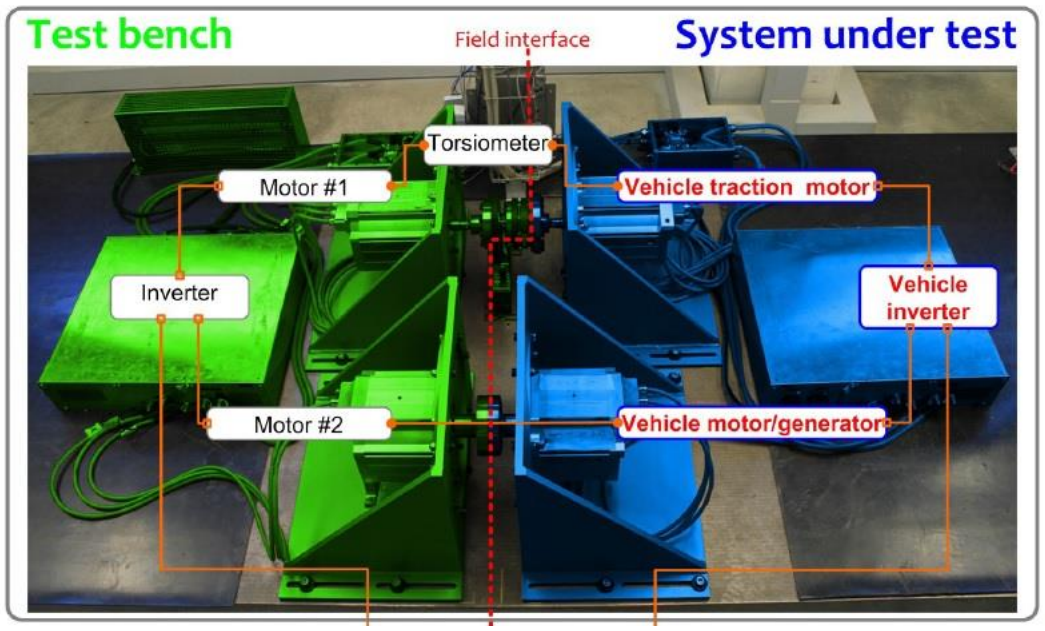


Figure 4.1: The external ECU bypass by dSPACE.

The bench is a dynamic test bench, meaning the applied torque can be either in direction. The setup is comprised of 2 test motors and 2 bench motors, with a torsiometer linking each test motor with bench motor. The test motor and the bench motor are connected to double inverters for acquiring the power. The inverters are connected via a common DC bus which supplies the power only for the losses during the testing. Thus, the testing system is based on power circulation.

The test motor control model and the bench motor control model can be simultaneously deployed on dSPACE target, as it can manage 2 ECUS concurrently. In addition, to have more control functionalities test motor model can be deployed on TI embedded ECU and have dSPACE target to bypass embedded ECU externally to support more functions and optimizations. With dSPACE system it's more convenient to use dSPACE ControlDesk for monitoring the system. Matlab/Simulink has full support for dSPACE software and hardware, which makes the model-based design to code generation a seamless process.

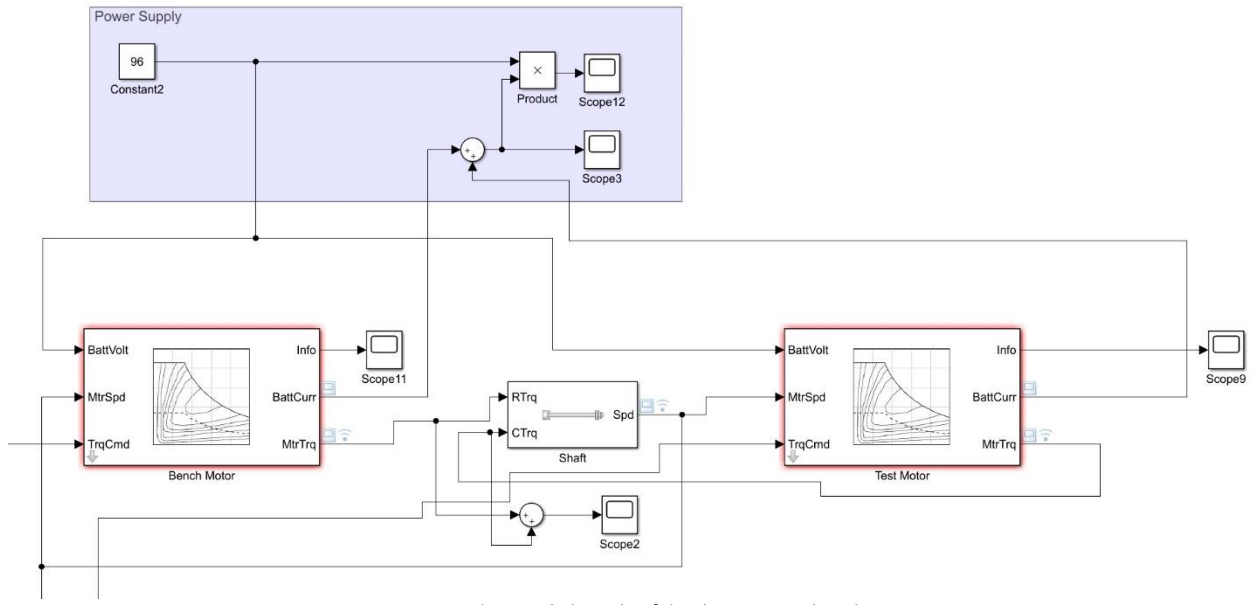


Figure 4.2: The Simulink mode of the dynamic test bench.

The model above is the Simulink model of the test bench. The bench motor and the test motor are torque controlled, meaning the command inputs are torques, and the shaft speed is a common input to both the motors. The test motor torque is acquired using a drive cycle, which simulates a driver input. On the other hand, the test bench motor torque command is the output of a PID controller that takes as input the error, that is the difference of the reference speed and the current shaft speed. This model is the model of the plant which is to be controlled by an embedded ECU or RCP. In the base model, the plant and controller (VMU) are used to acquire the results in simulation, which are used to validate the enhanced model.

The task is implemented using the same V-cycle scheme adapted in the introduction, which divides the development cycle in multiple stages and iterations. Also, along this chapter the base model and the enhanced model will be introduced. The Base model includes an ideal VMU, which disregards I/O interfaces, and the Simulink blocks representing the test bench setup. The enhanced model is based on the target hardware specifications. In this chapter dSPACE MicroAutoBox II is selected for reference. After the presentation of the 2 models the simulation results are discussed, and a conclusion made.

4.1 BASE MODEL

The base model of VMU takes into account only the control outputs and inputs. In the model, the drive cycle provides vehicle speed and acceleration. These are used as inputs in Vehicle Body Total Road Load Block to give the total force on the vehicle, which is transformed to as requested torque demanded by the driver. The vehicle speed is converted to [rad/s], because this speed is used as a reference speed which must be tracked by the shaft. The controller takes the reference as speed requested by user, which is subtracted by the shaft speed to close the control loop, obtaining the error by difference in speed. A simple PID controller has error in speed as input and output is the bench motor torque. The controller minimizes the speed difference and controls the test bench motor counter torque. The test motor receives a direct input from the VMU, i.e. requested torque by the driver.

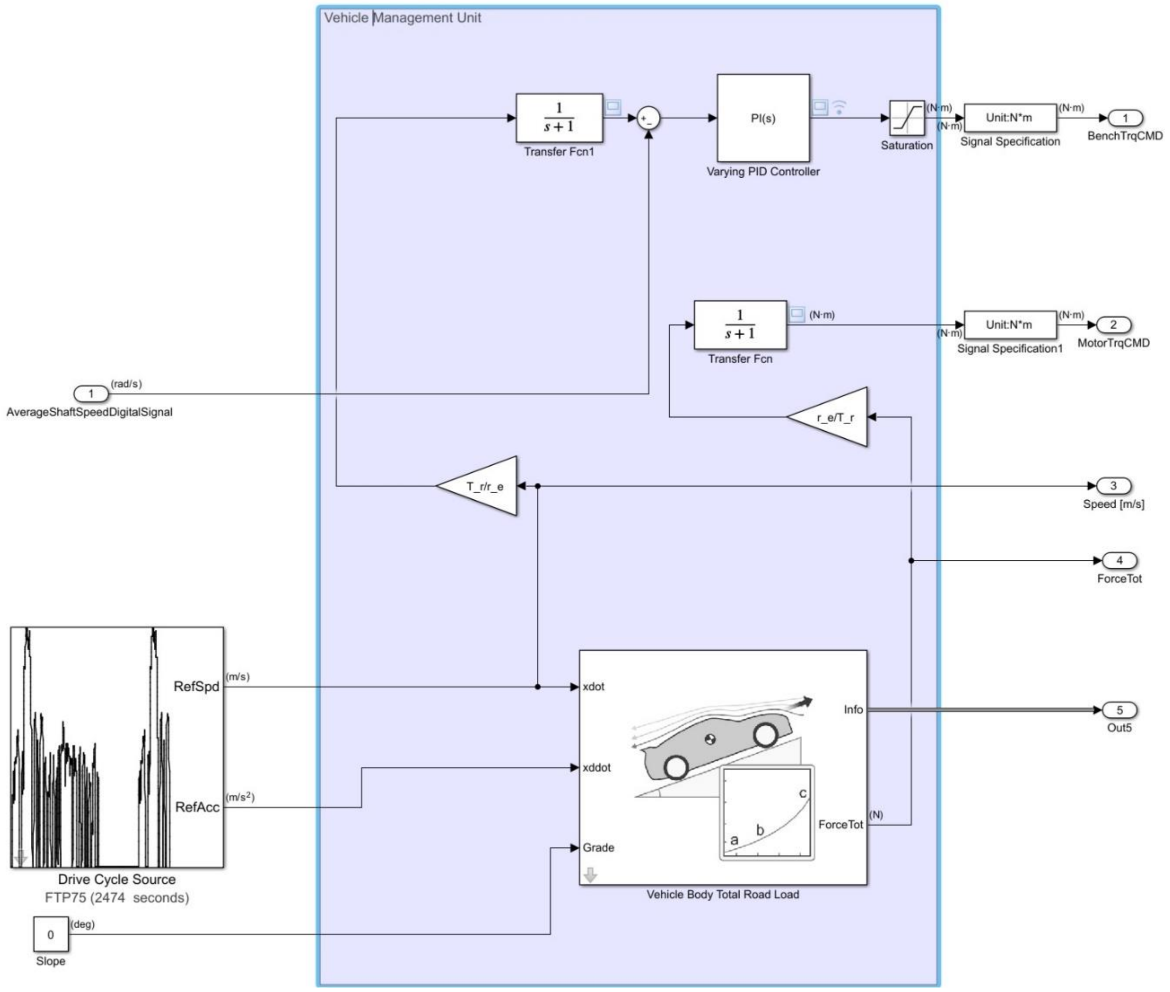


Figure 4.3: The VMU Simulink Model.

4.2 Enhanced Model

The enhanced model mainly considers the I/O interfaces of an embedded ECU or RCP. Since, the target is dSPACE MicroAutoBox II, it is known from the user manual, that the acquisition 16 Bits and signal input/output range is from -10 to +10 [V]. This is achieved in the simulation by introducing the ADC and DAC interface boundary. The new VMU scheme is as follows:

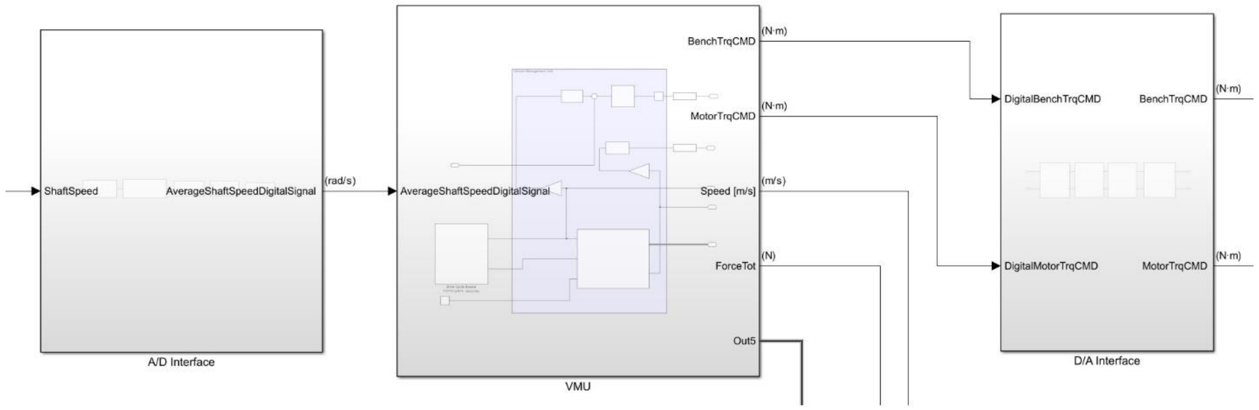


Figure 4.4: The VMU Simulink Model with I/O interfaces.

The I/O boundary at A/D side discretizes the signal, and D/A side makes the signal continuous time.

4.2.1 A/D Interface

The A/D interface comprises of 5 blocks, they are described respectively after the layout figure.

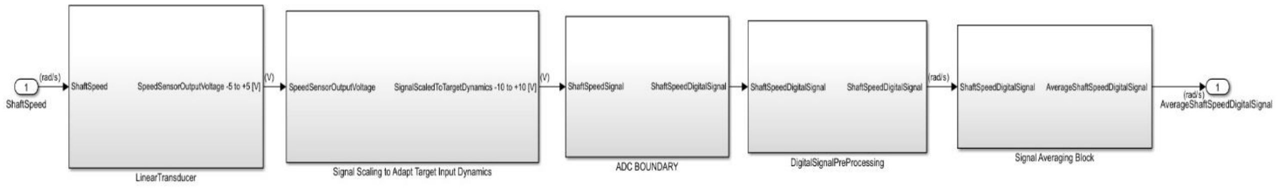


Figure 4.5: The A/D Interface Simulink Scheme.

The first block is a linear transducer that converts the measured physical quantity to an electrical signal in Volts. It is assumed that it is linear and introduces no delay. As it is just a transformation, it is composed of an input scaling term with an addition to offset term. Similarly, Signal Scaling to Adapt Target Input Dynamics block is an amplifier that scales the voltage to fit the input dynamics of dSPACE MicroAutoBox II. Again, it is simply composed of an input scaling term with an addition to offset term. The same is true for DigitalSignalPreProcessing block that converts the discrete digital value in the units the signal is acquired in, by scaling and offsetting the inputs.

These are the same blocks which are described in chapter 2 very clearly. Since all the blocks are parameterized, the specifications can be initialized from the Matlab script by entering specifications. Just for clarity the linear equation used for input/output relation is given below.

$$Output\ Signal = Input\ Sigal \times m + \ c$$

where m and c, are slope and offset terms, respectively.

The above equation is translated to Simulink scheme for linear transducer that converts speed to an electrical voltage, which is depicted below. The input signal is ShaftSpeed [rad/s] and the output signal is SpeedSensorOutputVoltege [V].

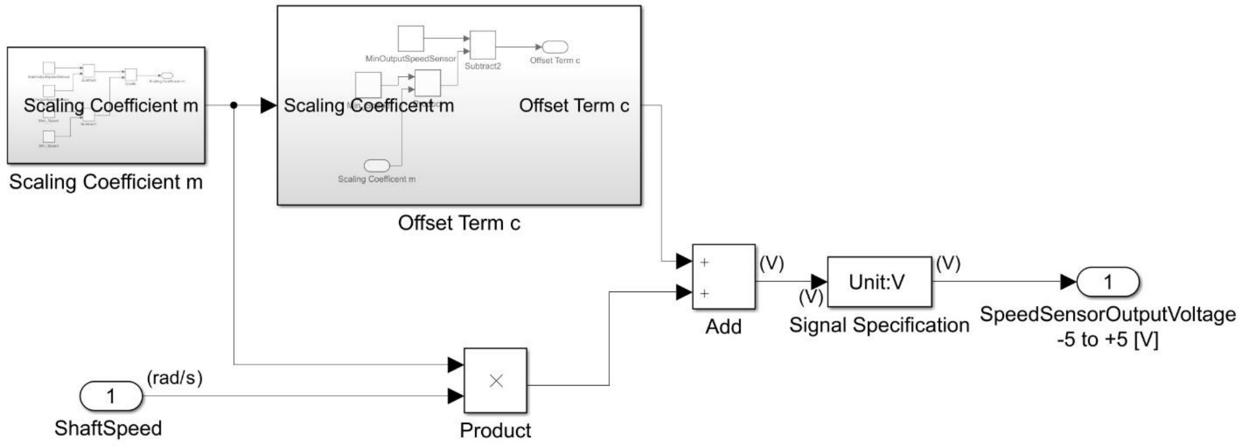


Figure 4.6: Linear Scaling of an input with offset addition.

The ADC block is the same as block used in Chapter 2, with just new specifications parameters. Here it worth noting, that the same enhanced model can be used for enhancing models seamlessly.

The signal averaging block is introduced here to clean the data by computing averages. The ADC is set to acquire the signal at 0.1 [ms] which is averaged after every 1 [ms]. This way the signal obtained is a clean signal that appears every 1ms for computation. A digital filter is not used as it may introduce unwanted delays to the signal.

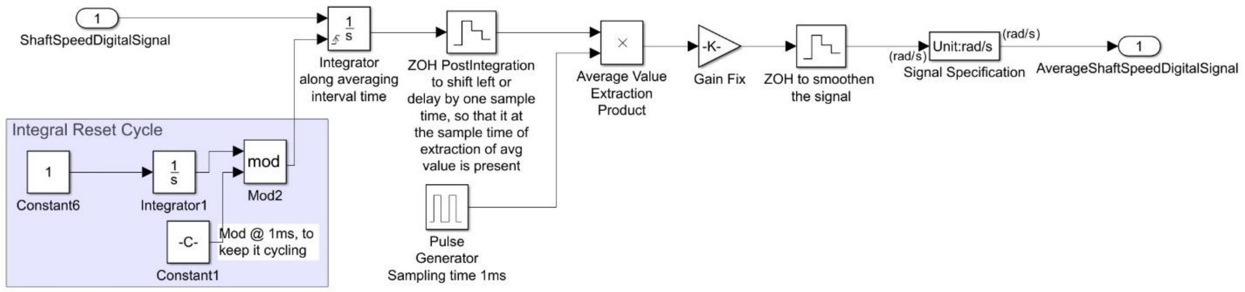

ShaftSpeedDigitalSignal
Integrator along averaging interval time
ZOH PostIntegration to shift left or delay by one sample time, so that it at the sample time of extraction of avg value is present
Average Value Extraction Product
Gain Fix
ZOH to smoothen the signal
Unit:rad/s
Signal Specification
AverageShaftSpeedDigitalSignal
Integral Reset Cycle
Constant6
Integrator1
mod
Mod2
-C-
Constant1
Mod @ 1ms, to keep it cycling
Pulse Generator Sampling time 1ms

Figure 4.7: Simulink Scheme to obtain signal average at periodic interval.

The idea is to acquire a signal every 1ms and integrate over a period of 1 [ms]. Then dividing the integral value by computed time gives the average value over the interval. The integral reset cycle takes mod of a ramp function and a constant, thus the output always stays from zero to the set constant value. For the application, it is required to obtain an average over an internal of 1ms, hence the constant is set to 0.001. Since, the integral is continuous and the only value interesting is the value at the time T + 0.001, the signal is multiplied by a periodic signal of period 0.001 that has amplitude one. The previous step gives the integral values at the required time period. Finally, it must be multiplied by the integration time to get the average value. Zero-order hold blocks are used to smoothen the signal.

This picture below illustrates what is described above. In the test condition, the test signal is a ramp, the sampling time is 0.1 [s] and averaging time is 1 [s].

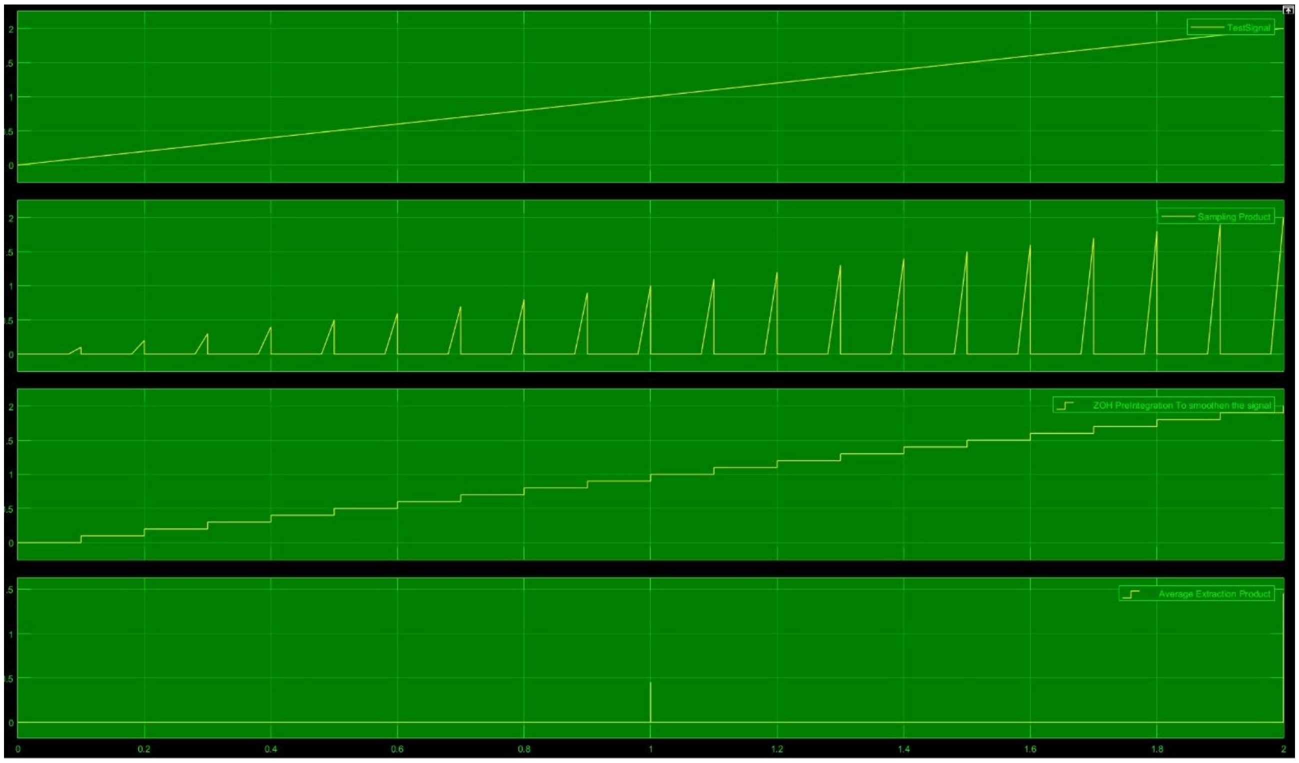


Figure 4.8: An example of periodic signal averaging.

4.2.2 D/A Interface

The D/A interface comprises of 4 blocks, they are described respectively after the layout figure.

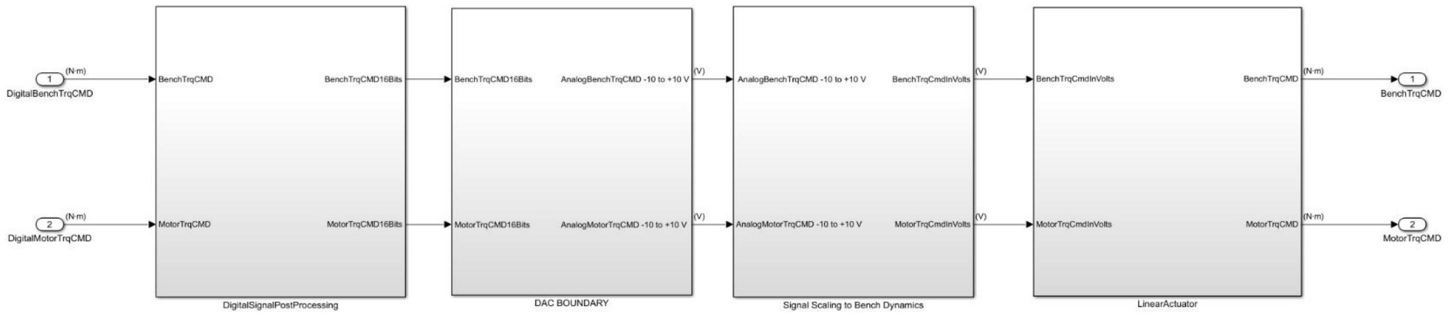


Figure 4.9: The D/A Interface Simulink Scheme.

The right most block is a linear Actuator that converts the electrical signal in to torque. It is assumed that it is linear and introduces no delay. As it is just a transformation, it is composed of an input scaling term with an addition to offset term. Similarly, Signal Scaling to Bench Dynamics block is an amplifier that scales the analog voltage that is output of dSPACE MicroAutoBox II, to fit the input dynamics of the bench motor. Again, it is simply composed of an input scaling term with an addition to offset term. The DigitalSignalPostProcessing block converts the signal in digital values from the signal, that was previously scaled to values of real measurement, by scaling and offsetting the inputs.

Also, the DAC is the same block from the chapter 2, with new specifications that are of the dSPACE MicroAutoBox II. Furthermore, the signal is smoothened by first order hold block in DigitalSignalPostProcessing block. This is performed to smoothen the signal before it is converted to analog signal.

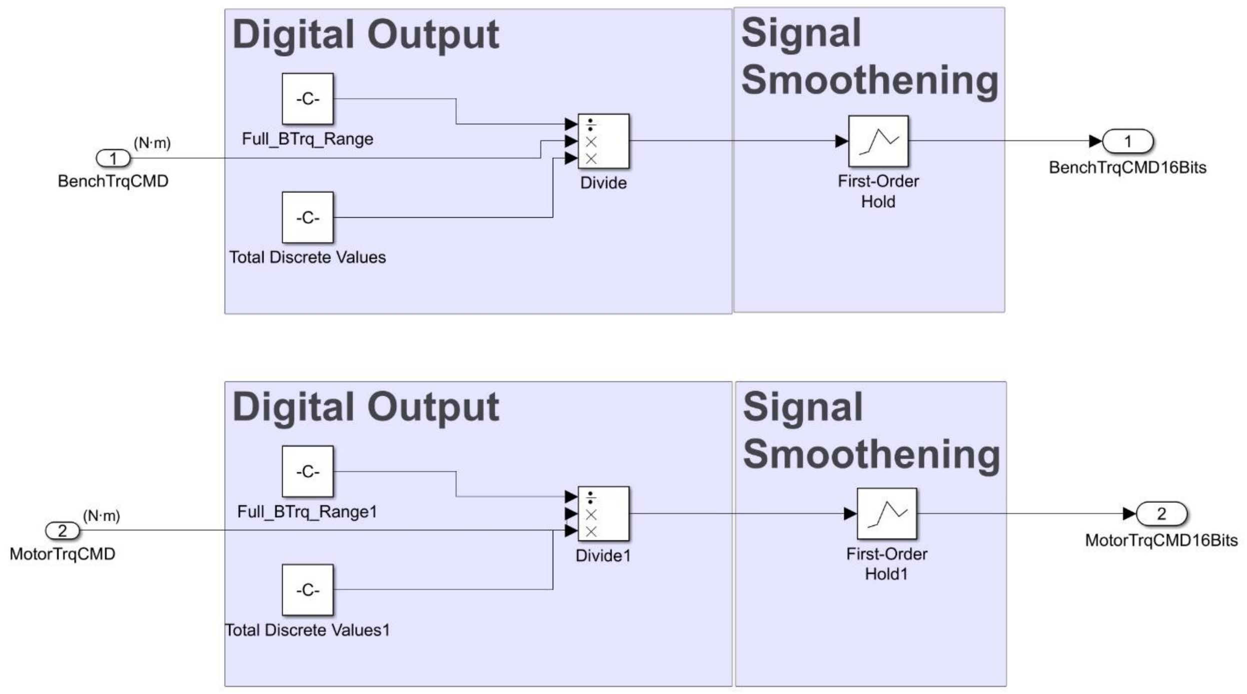


Figure 4.10: First-order hold for signal smoothening at output.

4.3 Comparison Of Base Model And Enhanced Model Results

The images below are of the shaft speed [rad/s] and Torques [Nm], from the base model and the enhanced model, respectively. The enhanced model takes into consideration the I/O interfaces with 16 Bits A/D and D/A sampled at 0.1 [ms]. The results are exactly the same in simulation, which leads to the conclusion that the dSPACE MicroAutoBox II is a good choice.

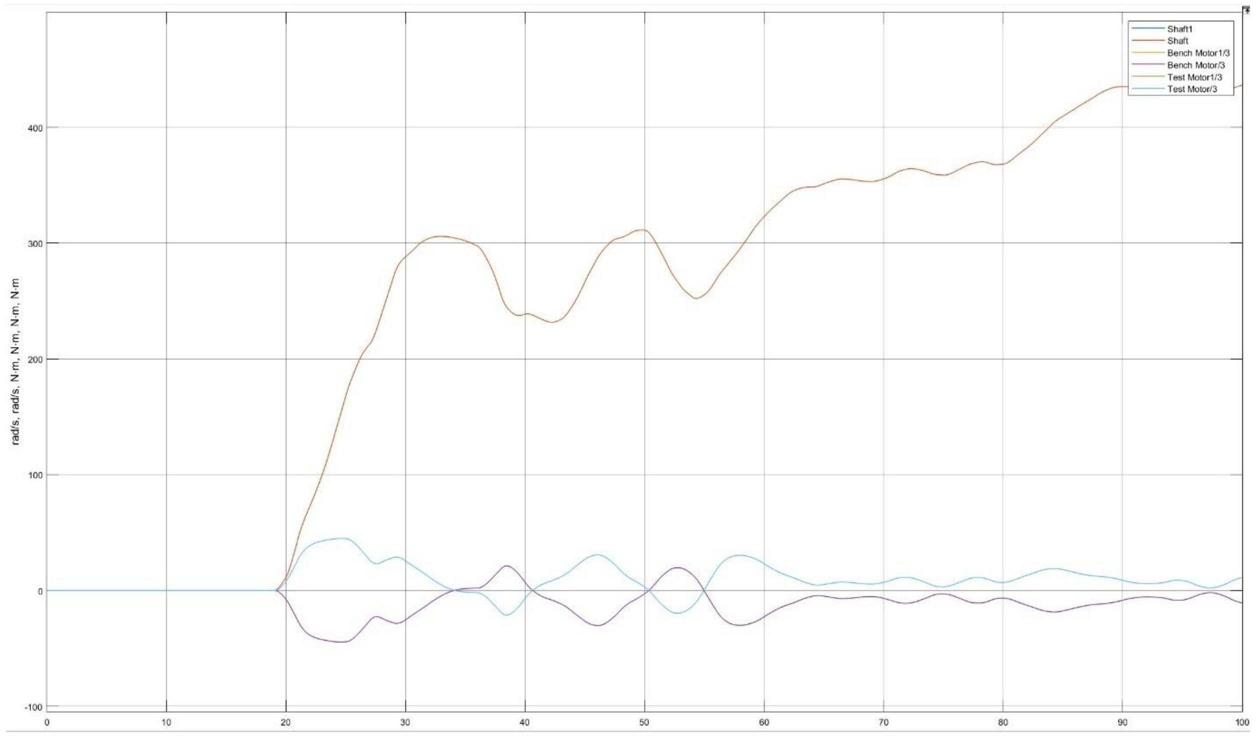


Figure 4.11: Base Model and Enhanced Model Speed and Torque Outputs.

It can be easily observed that the results of the enhanced model are almost identical to that of the base model. Hence, the enhanced model is valid. Also, the error between the 2 models at max is ± 0.04.

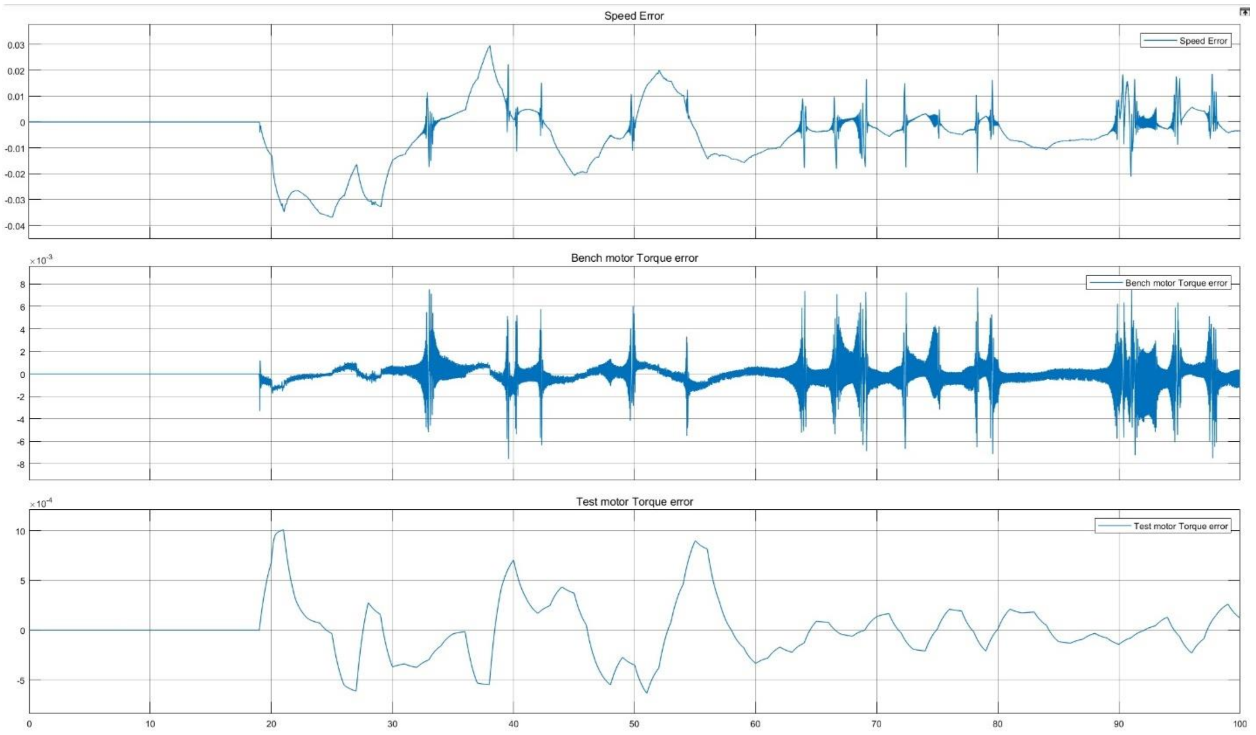


Figure 4.12: Error in Speed and Torque outputs, computed between base and enhanced model.

4.4 Code Generation Model Of Dynamic Test Bench

The following section of the chapter is dedicated for implementing code generation model. Few things are to be followed to achieve this. From the enhanced model, which is the model of the entire system, only VMU part and the signal Pre and Post processing blocks are required for code generation, since this is the part which must be deployed on the target.

This problem is not so trivial, because the blocks from Simulink used for modelling must be discretized and those blocks must be compatible for code generation. Also, when using subsystem blocks in Simulink, the sample time must be consistent. In addition, the code generation part must have some sort of interrupt routine, because the embedded systems lack enough resources, which is why, by scheduling code generation part is resource efficient. The model scheme is presented in chapter 3, titled as VMU model scheme.

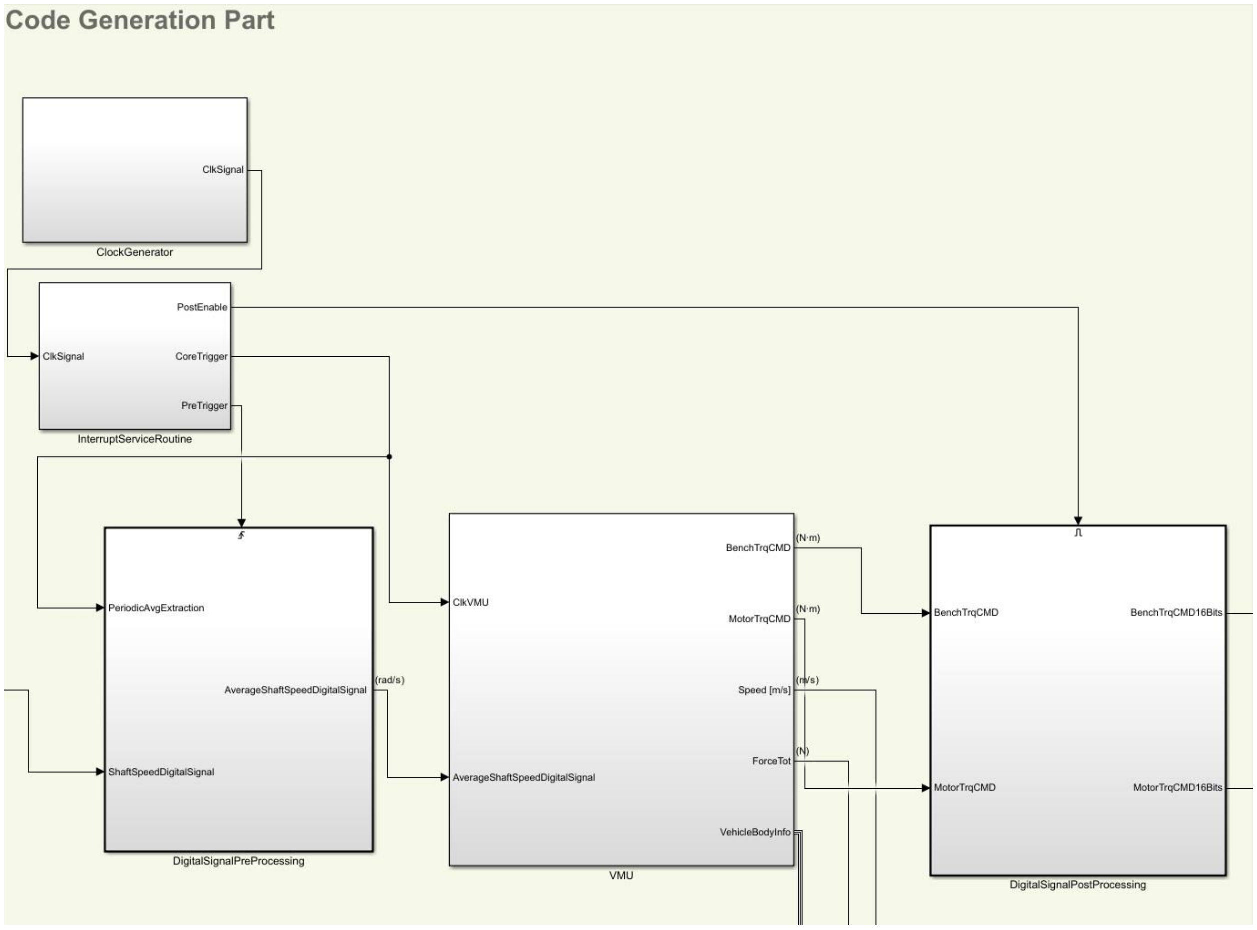


Figure 4.13: Code Generation Simulink model.

The code generation part of the system model has 5 blocks. In the next sections of this chapter, these blocks will be described, along with the changes made to these blocks to make them suitable for code generation. The 2 blocks on the bottom of the figure must be triggered and one enabled at accordingly. There are 3 periodic times, one for core computation, signal pre-processing, the last is for signal post-processing.

4.4.1 Discretization Of the Controller

The PID controller must be discretized, if it is to be used inside a triggered subsystem in Simulink. This is simply achieved by separating the drive cycle block and the VMU block, so that a triggered subsystem could be created for VMU. In the newly created subsystem, the continuous PID controller block is replaced by discrete PID controller.

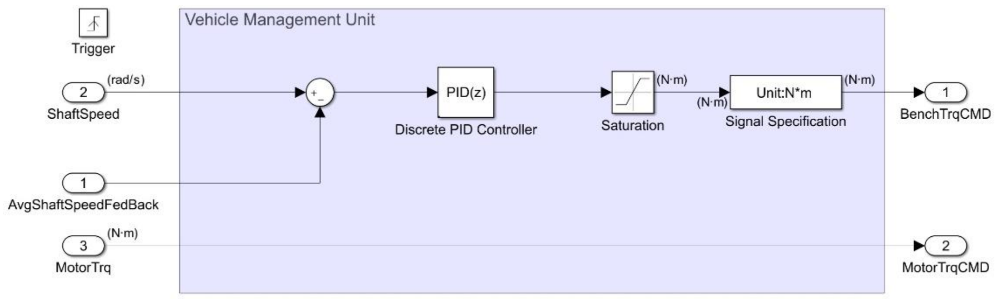


Figure 4.13: Discrete VMU Simulink model.

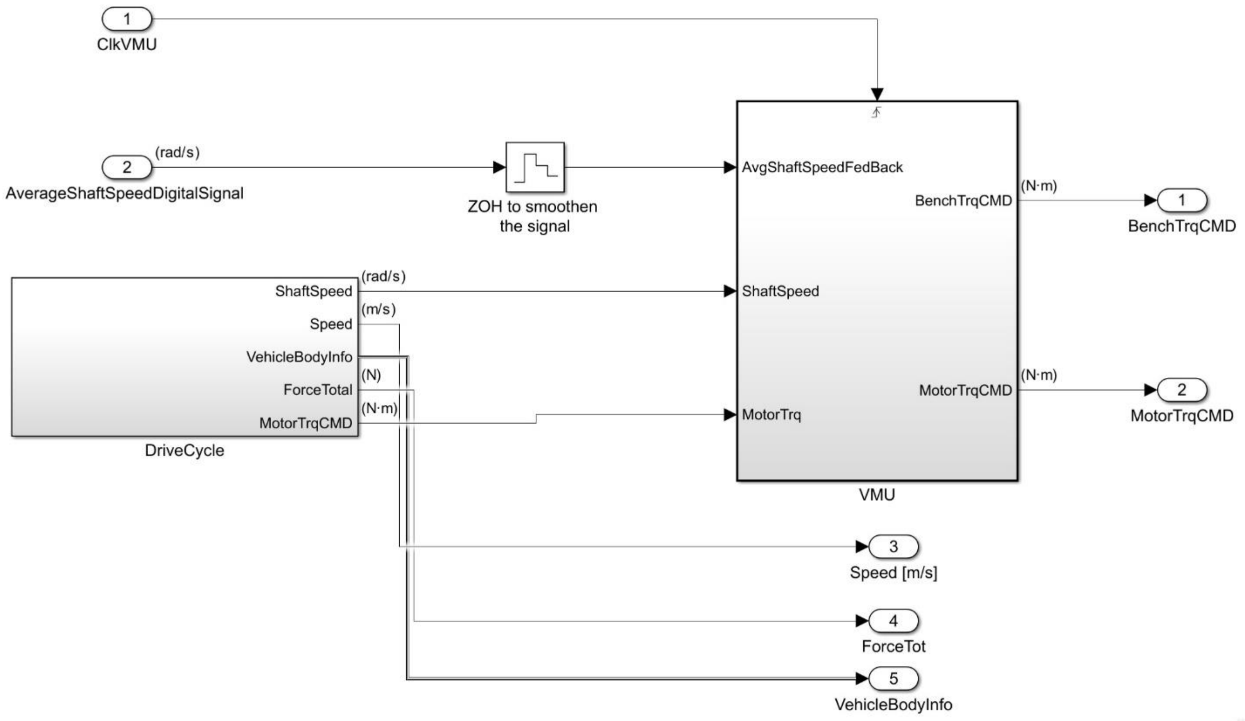


Figure 4.14: VMU Simulink model.

Furthermore, the VMU block must be triggered using clock signal. It should also be considered that the zero-order hold block is used here as it has a sampling time equal to core computation time, and it cannot be used inside signal pre-processing block due to different sample time. The need of smoothening the signal is necessary for the closed loop. Rapid fluctuations cause controller to behave aggressively.

4.4.2 Clock Generator And Interrupt Routines

The clock generator is comprised of pulse generator, which is used to simulate clock in the model. The pulse is generated at 100 [kHz], this is the clock frequency. The amplitude and the pulse period are set to one, since this is minimum pulse period that can be set. Furthermore, pulse generator block is suitable for code generation.

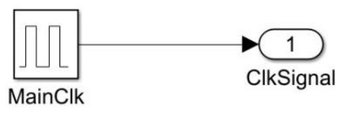


Figure 4.15: Clock Generator.

The interrupt routines should be done using the Simulink blocks that are provided for the target device. However, since we are dealing with only 3 processes, the following scheme is compatible for code generation.

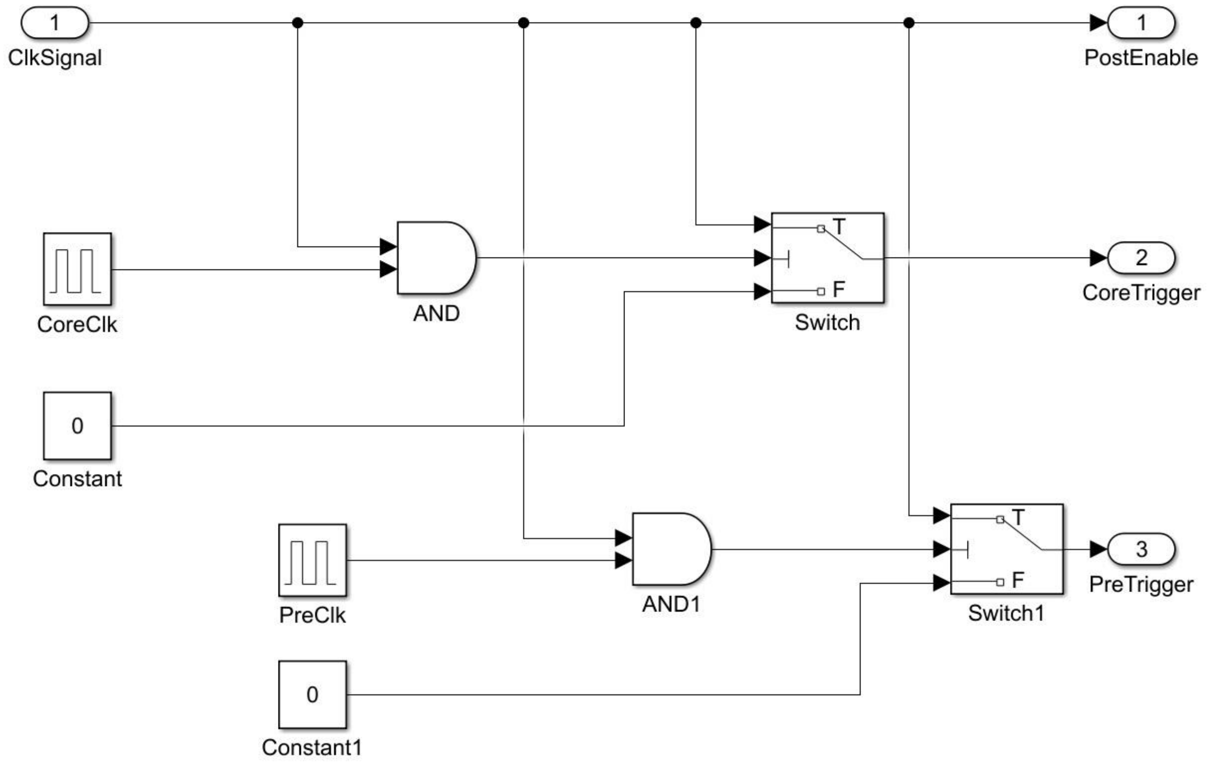


Figure 4.16: Interrupt Routine Scheme.

The clock signal is configured to generate a signal of amplitude 1 at the frequency of 100 [kHz], and a pulse period of 1%, this is the base frequency used. Only the frequency of other pulse generators will be changed to generate a pulse at a synchronous rate for individual process. For the signal pre-processing, it is required to be triggered at the frequency of 10 [kHz] and a pulse generator is used, because the signal is acquired at the same frequency using ADC. The logic is implemented using an AND block and a switch block, which is described as follows for the VMU core computation. For the VMU, to run at 1 [kHz], another pulse generator is used to generate a pulse at the frequency of 1 [kHz] with amplitude and pulse period set to one. The AND block outputs a Boolean true, i.e. 1 when both the inputs have the amplitude one. This is used to control the switch block, when the threshold condition that is the middle input to switch block is true, it lets the upper signal through otherwise it lets the lower signal through, which is a constant set to zero. Thus, the switch block outputs a clock signal at the frequency of 1[kHz]. The same is true for pre-processing clock, when both the generators produce an output of 1, than a trigger is send to the block. The idea is based on sending a clock signal when the slowest generator and the sends a signal.

4.4.3 Digital Signal Pre-Processing For Code Generation Model

The Pre-processing block is used for scaling the signal from bits to the actual signal values in its units. And to compute signal average periodically. The scaling block has math blocks from Simulink library, and they are compatible for code generation. However, for the enhanced model the periodic signal averaging is done using a continuous-time integrator. This is a problem, as the continuous-time integrator cannot be placed inside a triggered subsystem. This is done in a triggered subsystem.

Replacing the continuous-time integrator with discrete-time integrator is not an effective choice, as the results are not correct. The new strategy that must be implemented needs to be supported by Simulink for code generation and it must be suitable for use inside a triggered subsystem.

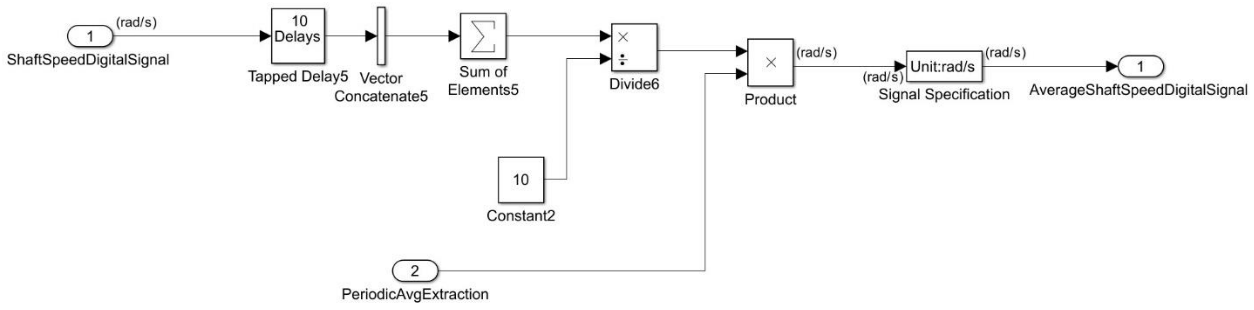


Figure 4.17: Periodic Signal Averaging scheme.

The goal is to acquire signal at the frequency of 10 [kHz], and compute average over 1 [kHz], meaning sampling every 0.1 [ms] and averaging every 1 [ms]. This can be done by using a tapped delay block, to acquire 10 samples and then arrange them in a vector using vector concatenate block. Now that the sample are stored in a vector, sum of all the elements is performed using the sum block. To compute an average of 10 samples the value obtained must be divided by 10. Finally, the values must be extracted from signal periodically, for this the signal is multiplied with the core trigger clock which operates at 1 [kHz]. Furthermore, a zero-hold is required to smoothen signal, because multiplying a signal with periodic pulses can cause rapid sudden fluctuations, which are unwanted in a closed loop control.

The plot on the following page, illustrates a linear ramp signal acquired every 0.1 [s]. An average must be computed every 1 [s]. The samples are delayed 10 samples, can be seen in the second subplot. Following this, sum is computed of these samples after they have been converted to a vector format. The sum must be divided by 10, as there are 10 samples. Then every 1 [s] the average value must be extracted from the signal. In the subplot, it can be clearly seen that the average of first 10 values is 0.45.

This hereby proves the strategy is effective and suitable for code generation because all the Simulink blocks are compatible.

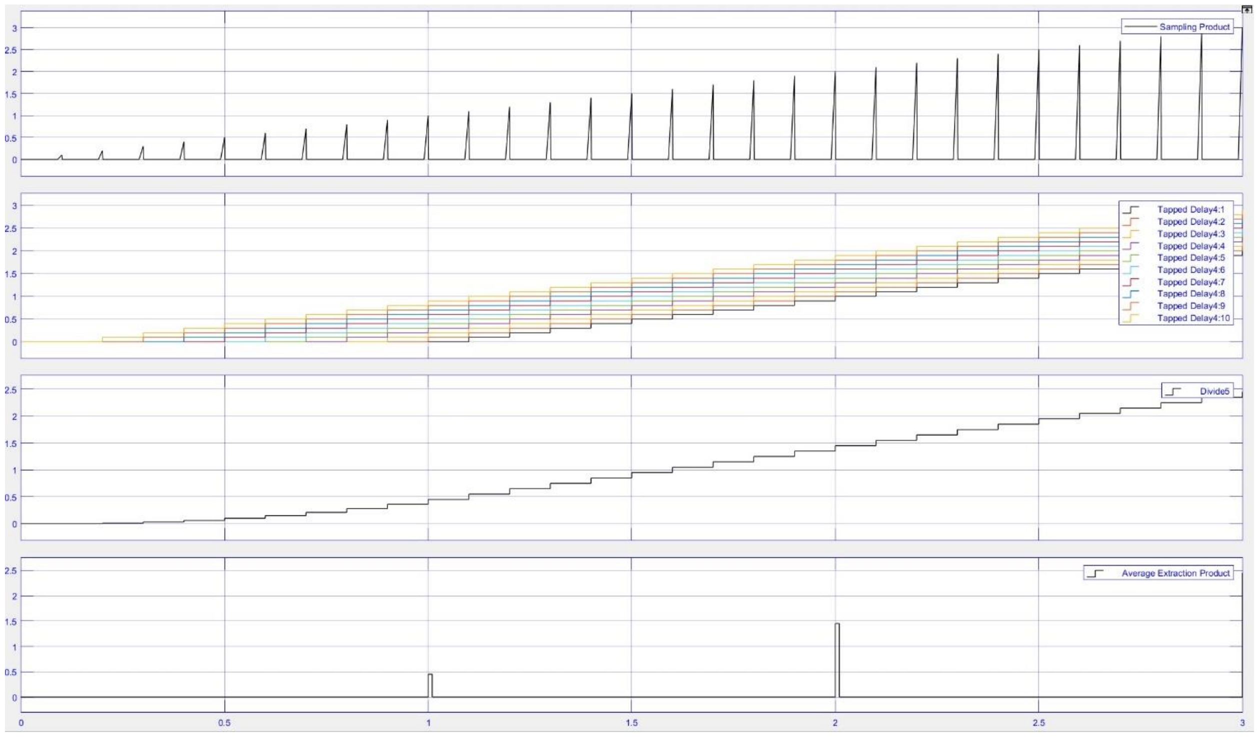


Figure 4.18: Periodic Signal Averaging example.

4.4.4 Digital Signal Post-Processing For Code Generation Model

For the signal post-processing, it is necessary that the blocks are used inside an enabled subsystem. It is crucial since, the enable block lets the blocks run for the next sample period, however a triggered subsystem runs only at the instant of triggering. The Post-processing block is used for scaling the signal from actual signal values in its units to bits, and signal smoothening periodically. The scaling block has math blocks from Simulink library, and they are compatible for code generation. Also, it has been discussed in the previous chapters and sections. However, for the enhanced model the signal smoothening is done using a first-order hold block. First-order hold cannot be used for discrete system as it contains few blocks which are not recommend for code generation. Moreover, first order hold blocks are not a good choice for using for signal smoothening, because it increases/decreases the current value linearly over the sampling period.

The plot on the next page addresses this issue visually. The black curve is the function to be smoothened, in red is the output from zero-order hold block from Simulink. In blue is the output from the signal smoothening implemented. It can be observed that the reference signal is smoothened in a linear manner. Also, it must be noted both the options, the first-order hold and the new strategy introduce a delay of one sample. Since the system to be controlled is mechanical delay of 1 [ms] is tolerable.

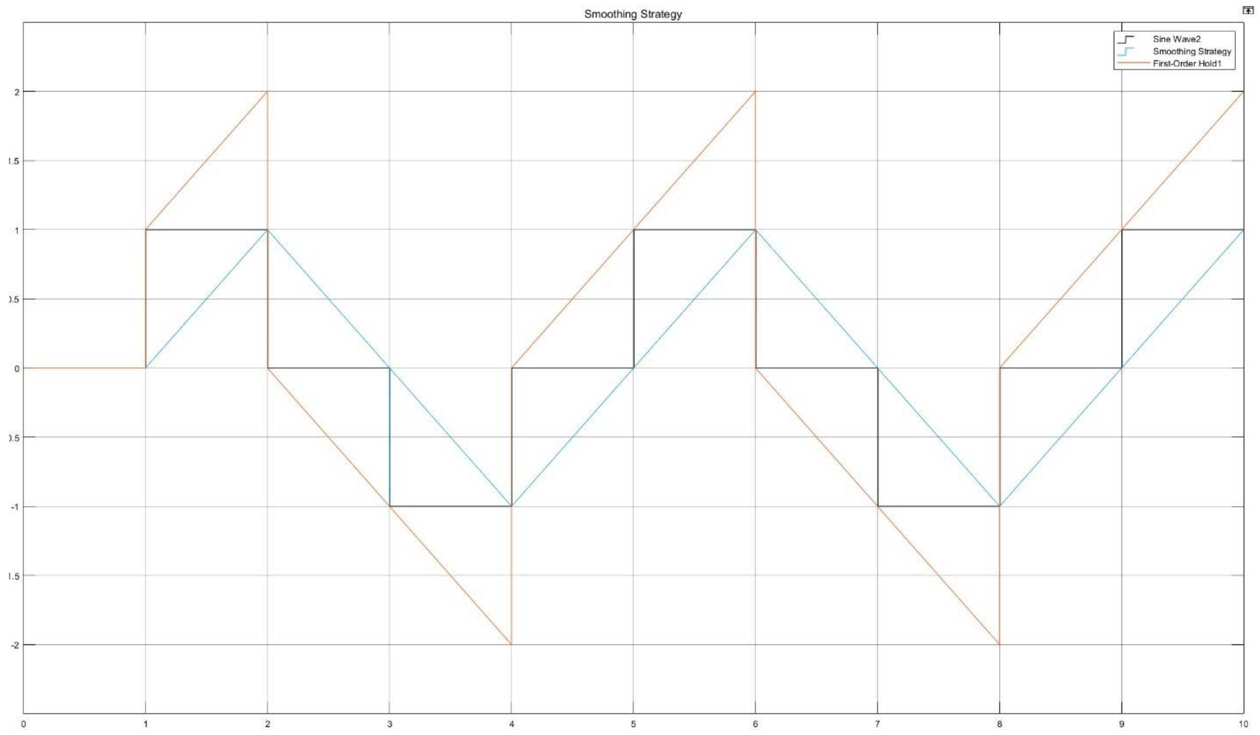


Figure 4.19: Signal smoothening comparison with first-order hold.

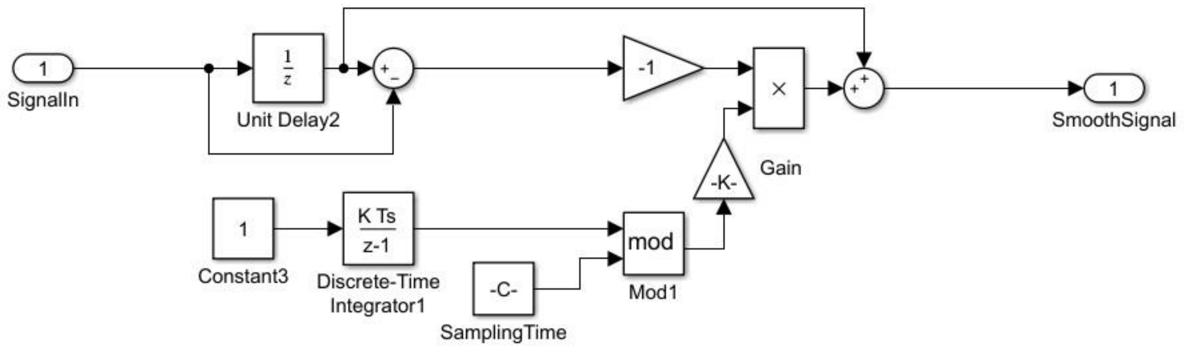


Figure 4.20: Simulink Scheme for signal smoothening.

The upper part of the Simulink model determines the difference between the current sampled value and the previous value, which is multiplied by -1 to invert the sign of the difference. The lower part generates a ramp for an interval of the sampling time. The idea is to add linearly to the sample before the current sample, the difference between the 2 sample points to obtain a first-order smooth function.

4.4.5 Simulation Result Validation

The result from the code generation model in simulation is identical to the one from the enhanced model. This step is necessary because of changes made to the model for code generation. The adaptions are a good fit.

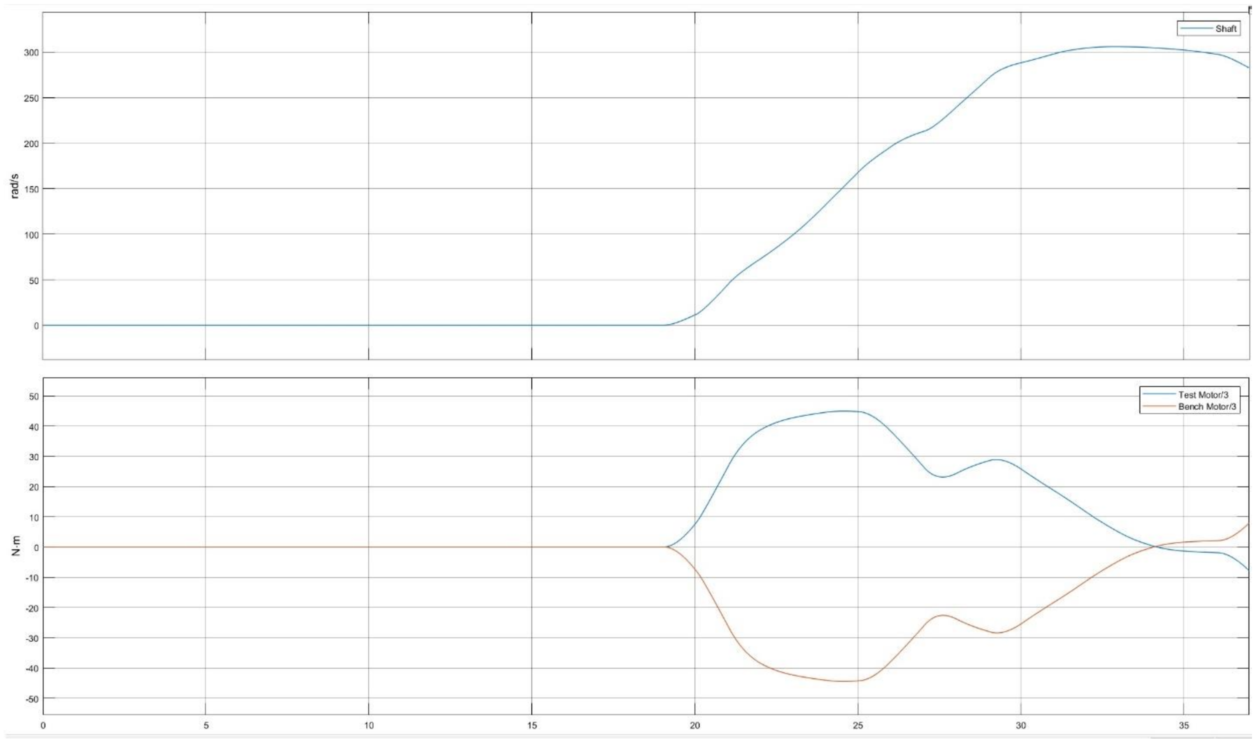


Figure 4.21: Code generation model results of simulation.

4.5 Code Generation Model Expected Outcomes

It is expected that there will be some unexpected delays introduced at the I/O interfaces. This is expected because in chapter 2, it is shown that a delay in order of [ms] is introduced in the code generation model. After through experimentations, the data collected can be used to calibrate the enhanced model by adding adequate amount of delays.

References

1. https://www.dSPACE.com/en/inc/home/products/systems/functp/bypassing.cfm
2. DataSheet For F28335 microcontroller.