University of Kentucky
UKnowledge

Theses and Dissertations--Electrical and Computer Engineering

Electrical and Computer Engineering

2011

CONTROL CHARACTERISTICS OF AN ALL-DIGITAL PROPORTIONAL-INTEGRAL-DERIVATIVE (PID) COMPENSATOR

David Michael Feinauer
University of Kentucky, dfeinauer@hotmail.com

Right click to open a feedback form in a new tab to let us know how this document benefits you.

Recommended Citation
Feinauer, David Michael, "CONTROL CHARACTERISTICS OF AN ALL-DIGITAL PROPORTIONAL-INTEGRAL-DERIVATIVE (PID) COMPENSATOR" (2011). *Theses and Dissertations--Electrical and Computer Engineering*. 3.
https://uknowledge.uky.edu/ece_etds/3

This Doctoral Dissertation is brought to you for free and open access by the Electrical and Computer Engineering at UKnowledge. It has been accepted for inclusion in Theses and Dissertations--Electrical and Computer Engineering by an authorized administrator of UKnowledge. For more information, please contact UKnowledge@lsv.uky.edu.

STUDENT AGREEMENT:

I represent that my thesis or dissertation and abstract are my original work. Proper attribution has been given to all outside sources. I understand that I am solely responsible for obtaining any needed copyright permissions. I have obtained and attached hereto needed written permission statements(s) from the owner(s) of each third-party copyrighted matter to be included in my work, allowing electronic distribution (if such use is not permitted by the fair use doctrine).

I hereby grant to The University of Kentucky and its agents the non-exclusive license to archive and make accessible my work in whole or in part in all forms of media, now or hereafter known. I agree that the document mentioned above may be made available immediately for worldwide access unless a preapproved embargo applies.

I retain all other ownership rights to the copyright of my work. I also retain the right to use in future works (such as articles or books) all or part of my work. I understand that I am free to register the copyright to my work.

REVIEW, APPROVAL AND ACCEPTANCE

The document mentioned above has been reviewed and accepted by the student's advisor, on behalf of the advisory committee, and by the Director of Graduate Studies (DGS), on behalf of the program; we verify that this is the final, approved version of the student's dissertation including all changes required by the advisory committee. The undersigned agree to abide by the statements above.

David Michael Feinauer, Student

Dr. Bruce Walcott, Major Professor

Zhi Chen, Ph.D., Director of Graduate Studies

CONTROL CHARACTERISTICS OF AN ALL-DIGITAL PROPORTIONAL-INTEGRAL-DERIVATIVE (PID) COMPENSATOR

DISSERTATION

A dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy in the
College of Engineering
at the University of Kentucky

By
David Michael Feinauer

Lexington, Kentucky

Director: Dr. Bruce Walcott, Professor of Electrical and Computer Engineering

Lexington, Kentucky

2011

Copyright © David Michael Feinauer 2011

ABSTRACT OF DISSERTATION

CONTROL CHARACTERISTICS OF AN ALL-DIGITAL PROPORTIONAL-INTEGRAL-DERIVATIVE (PID) COMPENSATOR

The digitization of classical control systems presents a number of challenges and opportunities with respect to the miniaturization, distribution, reliability verification and obsolescence of both the controller and the underlying system under control. A method for the design of proportional-integral-derivative (PID) compensators realized in the form of all-digital components is presented. All-digital refers to a system implementation that is realizable with a wide range of digital logic components including discrete digital logic elements and programmable logic devices (PLDs) such as field-programmable gate arrays. The proportional, integral and derivative components of the classical PID control law were re-envisioned in terms of frequency of occurrences or counts for adaptation to combinatorial and sequential digital logic. Modification of the control scheme around this newly formed representation of system error enables the development of a PID-like FPGA-based or PLD-based controller. Details of the design of an all-digital PID-like controller including abstract, causal block diagrams and a *MATLAB*® and *Simulink*® based implementation are presented. The compensator was simulated in a velocity tracking DC motor control application and was found to perform comparably to that of a classical PID based control. Methods for assessing the resultant stability of an all-digital PID compensated system under control are discussed.

KEYWORDS: Proportional-integral-derivative (PID) control, all-digital control, FPGA-based control, PLD-based control, frequency tracking control.

David Michael Feinauer

November 30, 2011

CONTROL CHARACTERISTICS OF AN ALL-DIGITAL PROPORTIONAL-INTEGRAL-DERIVATIVE (PID) COMPENSATOR

By

David Michael Feinauer

Bruce L. Walcott, Ph.D.
Director of Dissertation

Zhi Chen, Ph.D.
Director of Graduate Studies

November 30, 2011

This dissertation is dedicated to the many educators, family members and friends who have supported me throughout its development.

ACKNOWLEDGEMENTS

The development of this work benefited greatly from the support and guidance of many educators, friends and family members.

In terms of direct support and impact on this work, I would like to thank my committee members for the solid preparation they provided me throughout my coursework. Particularly, I thank Professors Walcott, Zhang and Holloway for the instruction they provided me in the areas of deterministic systems theory, feedback control, optimal control theory and control of manufacturing and event systems. It was instrumental in developing this work; a work I hope is reflective of the quality of the preparation I received. I would also like to express gratitude to Professor Jaromczyk for his tutelage in the area of algorithm design and complexity analysis—it has been beneficial to my re-envisioning classical control laws in the digital realm. I am grateful to Professor Eugene Bruce, although not directly reflected in this work, for inspiring my interest in biological systems through his biomedical signals courses. It was through this rigorous coursework that I honed my abilities to analyze information and think critically—skills that have proven critical in the conduct of research.

I have long felt that the true measure of any university is a metric that evaluates the impact the university—comprised of its people—has on the lives of the community it serves. The potential role that a university and its people can have as an agent of change in the local and broader community, for me, is one of the great appeals of life in academe. It is with this in mind that I wish to profoundly thank the members of my committee for reaffirming this belief and being exemplars of the many ways one can impact people through the university setting.

Dr. Yuming Zhang has demonstrated to me the value of applying scholarly results to problems existent in industry and the opportunity for transferring one's endeavors into the broader community. Most impressively, he has shown that one can approach all things with an overall jolly temperament that I find inspiring.

Dr. Larry Holloway has shown me the possibility for one to evolve one's academic pursuits based on changing market trends and the broad impact one can ignite by expressing a vision for an organization and inspiring its people to take action towards a realization of that vision.

Dr. Jerzey Jaromczyk has inspired me by his embrace of the role of mentor and advisor as demonstrated by the genuine, lasting interest he takes in his students with respect to their current and future pursuits. Throughout the years I was delighted by his inquiries as to my current path and progress in life—long after that first class I followed under him.

Dr. Suzanne Smith exemplifies the direct impact one's research can have on a state and the future career opportunities for its students. I'm continually inspired by both the volume and results of the active projects supported by Dr. Smith and the effort she takes to ensure that the endeavors she supports create numerous opportunities for student involvement at all levels.

Lastly, Dr. Bruce Walcott has confirmed the many opportunities for lasting impact one can have by supporting the local community through service to its people— be it directly through outreach endeavors or indirectly by way of positive changes owing to active service on steering committees. The number of varied interests including scholarly research, teaching, community outreach, entrepreneurship and charity successfully balanced by Dr. Walcott affirms my interest in further academic pursuits.

Collectively and individually, my committee members have served me well as role models and I expect they will continue to do so in my future pursuits. For this I am thankful.

I would be remiss without mentioning the encouragement, support and prodding of many family members and friends that ultimately made the completion of this work possible. I would especially like to acknowledge my parents for their continual love and support throughout this and all my endeavors. I am grateful to Jen Eli, Weston Johnson, Pat Quillen, Casey Harr, Dr. Jane Riggs and Jennifer Doerge for our many interactions during my graduate studies at UK; the experience would have been wanting in their absence.

Lastly, I am thankful to Dr. Bruce Walcott and Mrs. Katherine Walcott for their encouragement and support of my personal and professional development throughout my time at the university. Dr. Walcott, for embracing my diverse pursuits and meeting the ebbs and flows of my progress in this work with unbounded patience, I am grateful.

TABLE OF CONTENTS

List of Tables vii

List of Figures viii

I. Introduction 1
II. PID Control Background 5
A. Proportional–Integral–Derivative (PID) Control Background 6
B. PID Methods 7
i. Ziegler–Nichols 8
ii. Root Locus Methods 11
iii. Frequency Domain 19
III. Motivations for an All-Digital PID Compensator 21
A. PID Prevalence in Industry 21
B. Trends in Industrial Control Implementation 22
C. Proposed Problem Description 25
IV. Proposed Solution 31
A. All-Digital Controller / Plant / Controller Interface 32
B. All–Digital PID Controller Design Framework 33
i. Generation of a Proportional Error Signal 38
ii. Generation of an Integral Error Signal 41
iii. Generation of a Derivative Error Signal 43
iv. Generation of a Combined PID Error Signal 45
v. Generation of the Error and Controller Output Signals 47
C. Procedure for Determining All-Digital PID Control Parameters 50
V. Experimental Research 52
A. Revisions to the All-Digital PID Compensator 53
B. All-Digital PID Versus Traditional PID 56
i. ADPID comparison to a Baseline System 56
ii. Setpoint and Parameter Sensitivity in Velocity Tracking 69
iii. Baseline Comparison for an Additional DC Motor Plant 73
C. Stability Discussion 82
VI. Conclusions / Future Work 86

Appendix A 92
Appendix B 97
Appendix C 104
Appendix D 106
Appendix E 110
Appendix F 122

References 124

Vita 127

LIST OF TABLES

Table II-1 – Ziegler-Nichols parameters for PID controller tuning [8]. 10
Table II-2 – General Routh table constructed for a nth order polynomial..... 14
Table IV-1 – Truth Table of the error magnitude and sign functions..... 47
Table IV-2- Table depicting the operations required to map the error sign signal to the appropriate scaling factors. 48
Table V-1 – Parameters for DC motor mode from [10]. 52
Table V-2 – Performance characteristics of the PID compensated plants in Figure V-4 and Figure V-5. 63
Table V-3 – Armature-controlled DC motor with load specifications. 73
Table V-4 – Performance characteristics of the baseline and all-digital PID compensated armature-controlled DC motor plant with load..... 79

LIST OF FIGURES

Figure II-1 – A plant under control by a negative feedback control system with input U(s), output Y(s) and error E(s). 5
Figure II-2 – Block diagram of a PID compensator. 7
Figure II-3 – System step response depicting lag time (L) and reaction rate (R). 9
Figure II-4 – System GH(s) with negative feedback of variable gain K. 11
Figure II-5 – A root locus plot of the example system with PID feedback control. 18
Figure IV-1 – Block diagram of a typical plant under control by a discrete PID controller and the proposed all-digital PID controller. 33
Figure IV-2 – Diagram of five pulse trains–reference, output, error, error magnitude and error direction. 35
Figure IV-3 – High-level view of the constituent components of the proposed ADPID framework. 36
Figure IV-4 – Block diagram of 16-bit binary up/down counter. 39
Figure IV-5 – Detailed block diagram view of the proposed all-digital PID controller and its constituent components. 49
Figure V-1 – Initial all-digital PID results for a commanded tracking velocity of 60 rpm (top) and 120 rpm (bottom). 54
Figure V-2 – Root locus plot of the transfer function of a DC motor plant. 57
Figure V-3 – Root locus plot of the closed loop PID compensated DC motor system. ... 59
Figure V-4 – Velocity tracking classical PID compensated DC motor. 61
Figure V-5 – PID compensated DC motor plant model with typical controller to motor and motor to controller interfacing. 61
Figure V-6 – Position, velocity and acceleration outputs from the PID compensated DC motor system. 62
Figure V-7 - Position, velocity and acceleration outputs from the PID compensated DC motor system with typical controller-plant interfaces. 63
Figure V-8 – Absolute and percentage difference of the tracking velocities attained by the theoretical and modified systems. 65
Figure V-9 – Signals representative of the PWM interface to the classical PID compensator and one channel of the quadrature encoder output of the system plant. 66
Figure V-10 – Velocity tracking all-digital PID compensated DC motor. 67
Figure V-11 – Position, velocity and acceleration outputs from the all-digital PID compensated DC motor system. 68
Figure V-12 – Velocity tracking comparison of the baseline and all-digital compensators. 70
Figure V-13 – Velocity and control outputs of a closed-loop DC motor plant with an integral gain of 0.04 (left) and 0.06 (right). 72
Figure V-14 – Root locus plot of the transfer function for a DC motor plant. 75

Figure V-15 – Root locus plot of the closed loop PID compensated DC motor system. . 77
Figure V-16 – Position and velocity output of a PID compensated DC motor with load tracking a reference velocity of 10 rpm. 78
Figure V-17 – Position and velocity output of the ADPID compensated DC motor with inertial load tracking a reference velocity of 10 rpm. 80
Figure V-18 – Proportional only control of a motor plant tracking a rotational velocity of 10 rpm. 81
Figure V-19 – PID control of a motor tracking a velocity of 10 rpm with increased derivative gain, decreased proportional gain and significantly reduced integral gain. 82

I. INTRODUCTION

A common design and analysis method in engineering is the use of a "systems-based approach," where the output or operation of a plant is related to the corresponding inputs and conditions effecting the output through a description or model of the system dynamics. In bringing a concept or enterprise to fruition, design engineers are challenged with the task of optimizing a number of factors including: cost, time to market (TTM), quality and performance. The relative importance of the competing factors varies greatly from industry to industry. In the realm of commodity consumer products, for example, time to market may be considered the most critical factor to maintain competitiveness whereas in developing safety critical systems, reliability is of highest concern—one expects.

The body of knowledge known as control systems theory can aid design engineers in the development of a system, the optimization of the implementation and the resultant performance of said system with regards to the previously enumerated factors and other relevant constraints. Ideally, the available expertise and the potential for its application are considered in conjunction with other criteria when the design is in its infancy. Oftentimes it is the case that design decisions are made to minimize certain factors before considering the resources necessary for control theory to be applied to the design. Treating the various considerations individually instead of in concert, potentially limits the overall design execution with respect to reliability and other performance-related factors. Historically, the digitization of systems has afforded engineers great opportunity to achieve improved performance with respect to a diverse set of design factors.

Despite the many advances, design engineers remain challenged with balancing the aforementioned characteristics along with new or emerging concerns including energy consumption and sustainability. As systems reduce in size, become more distributed and time-to-market concerns increase the rate of component obsolescence, design engineers

and control systems engineers in particular are faced with a number of new considerations. On a different front, the trend toward “cloud” computing challenges engineers to stop distinguishing hardware devices from software methods and to think of computation as a general more unified resource.

The trends and factors discussed are not exclusive to the high-level design of the overall digital device or system; the factors propagate to the design of all constituent systems including control systems. Proportional-integral-derivative (PID) control systems are a widely used control technology with a long history. Throughout this work, the fundamental components of a classical PID compensator are re-envisioned. A method for the design of a PID compensator realized in the form of all-digital components is presented. The compound adjective “all-digital” is used herein to describe a system implementation that is realizable with a range of digital logic components. The control method and design are well-suited to an implementation with a range of programmable logic devices or discrete components. The all-digital PID (ADPID) design capitalizes on the inherently digital nature and interfaces of many modern plants to be controlled. The compensator design presented is intended for the class of digital systems that are actuated by a pulse-width modulated (PWM) input signal with a system output that can be represented by a digital pulse train when measured by a digital transducer, such as an optical encoder. The design presents a number of opportunities to address design issues related to miniaturization, distribution, performance and obsolescence.

Chapter II of this document presents a summary of classical proportional-integral-derivative (PID) control theory from the theoretical-based academic domain and the practice-based industrial domain. An example of the selection or tuning of the PID control parameters based on root locus techniques is presented.

Chapter III explores the lasting dominance of PID based control in industrial applications and the trends emerging in digital, field-programmable gate array-based (FPGA-based)

control. Challenges and opportunities that remain within the well-established PID control systems domain are discussed. The obstacles that engineers face when using non-custom microcontrollers for the design of control systems are presented. Issues related to reliability and device obsolescence motivate the all-digital controller implementation proposed in this dissertation.

Details of the design of the all-digital PID-like controller are revealed in chapter IV. Classical, highly abstract, causal block diagrams of the design are presented. Implementation details of the digital design in The MathWorks *MATLAB*® and *Simulink® Release 2010b*, a software modeling and simulation package typical of the control systems academic domain, are described. A rule of thumb for tuning the control parameters of the proposed system is posed. The implementations for the models explained in this section are contained in the Appendices of this document and appropriately referenced throughout the discussion. The models and signal routings presented were completed with the goal of an all-digital implementation in mind. Construction of the models in this manner affords the use of the common controls simulation package for proof of concept and analysis while revealing a model that will readily map to a gate level or functional level description typical of a hardware description language (HDL) model. Such a model would be necessary for a custom digital implementation.

Chapter V introduces framework modifications to the initial design to correct the actual operation to comply with the stated functionality and for improved overall performance. The performance characteristics of the modified all-digital PID compensator design are contrasted with that of a classical PID controller when applied to models of a basic field-controlled and armature-controlled DC motor. Models for sensors required to close the loop of the control system were created. New methods for determining the error between a reference and feedback signal in terms of frequency are addressed. The ability of each controller to track a velocity (reference input frequency) is analyzed. The sensitivity of

the closed-loop system to the tuned control parameters selected under the guidance of the previously stated rule-of-thumb is analyzed. The proposed all-digital design effects a control law that is similar to a minimum time bang-off or hysteresis type controller. Methods for investigating the stability of the all-digital model are proposed.

A summary of the author's contributions is presented in chapter VI. The ability for the proposed ADPID model to the address current issues related to digital control system devices posed in chapter III is discussed. Opportunities for future work related to: further development of the all-digital model, stability analysis of the resulting model, porting of the model to an HDL-based compact ASIC design, development of an auto-tuning method for the control parameters, and implementation of the model as an FPGA-based system-on-a-chip are discussed.

Copyright © David Michael Feinauer 2011

II. PID CONTROL BACKGROUND

The fundamental function of automatic control is to monitor a system or process variable and to actuate a controlled response in the process or system. In a classical control system, the error between the present and desired system operating points or process control variables forms a negative feedback signal used by the controller to determine and affect the appropriate response in the system [1]. A block diagram of a negative feedback control system is shown in Figure II-1. A more advanced form of negative feedback control known as proportional-integral-derivative (PID) control utilizes negative feedback comprised of three components. Under this paradigm, the components are proportional to: the present system error, the accumulation of the error over time, and the rate of change of the error—hence the designation 'proportional-integral-derivative.' Background on the development of this control method and an in depth description of PID methods from the academic and industrial arenas are presented in sections A and B respectively.

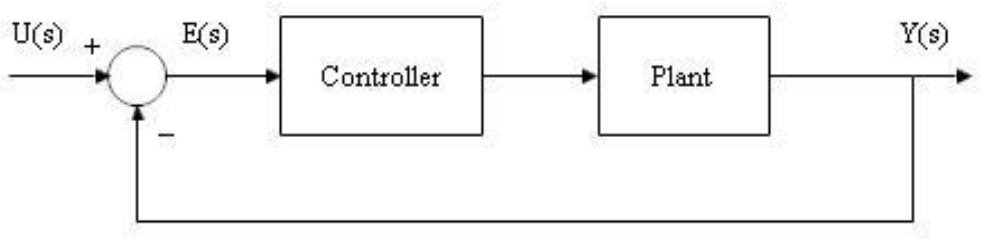


Figure II-1 – A plant under control by a negative feedback control system with input U(s), output Y(s) and error E(s).

A. Proportional–Integral–Derivative (PID) Control Background

The advent of PID control can be traced to the Taylor and Foxboro Instrument Companies in the late 1930s and the entrepreneurial culture of said companies as they assumed the roles of field-engineer and system integrator to address their customers' problems in order to sell their instrumentation and control products. Up to this time, the field of automatic control lacked a mathematical foundation and was primarily heuristic—based on observations by the instrument companies. Innovations to the PID controller led to its success and routine use as a control method by: developing methods for determining controller settings; creating techniques to assess a system's suitability for control (controllability); and improving the durability and robustness of the control system, actuators and sensors. These innovations developed out of the practice-based theory of the instrument companies and the mathematical-based foundations from the burgeoning academic field of study [2]. Today, PID control remains the most widely-used controller, pervading numerous industries, with experts roughly estimating that billions of new PID control loops are installed annually [3].

A control law with negative feedback formed by terms proportional to the system's error, the accumulation of the error over time and the rate of change of the error are known as proportional-integral-derivative control. A control law with negative feedback comprised of a subset of these three components is known simply by its constituent components (i.e. a proportional-integral (PI) control consists of a negative feedback signal proportional to the error and the accumulation of the error). A PID control law requires a control parameter (or tuning parameter) for each of the constituent control components. Classical analytical and experimental methods for determining the control parameters (or tuning the controller) from both the theoretical and industrial foundations of PID control are detailed in the following sections. The control expertise gap of the end user trying to implement the control loop contributes to the prevalence of PID because of its relative simplicity [4].

B. PID METHODS

A basic block diagram of a PID control system is shown in Figure II-2. A mathematical description of the system's transfer function is shown in (1). In this equation, the requisite control parameters (K_P, K_I, and K_D) need to be determined. The equation in (1) expresses the three parameters in "parallel" form representing the resulting effect of the compensator as the sum of the effects of the three individual components [5]. One of the early methods for determining the PID controller parameters was developed by Ziegler and Nichols in 1942 [2]. This method is demonstrated in the section below. Classical methods for determining control parameters based on root locus and frequency domain techniques are presented in the subsequent subsections.

$$GPID(s) = K_P + \frac{K_I}{s} + K_D s = \frac{K_D s^2 + K_P s + K_I}{s} \tag{1}$$

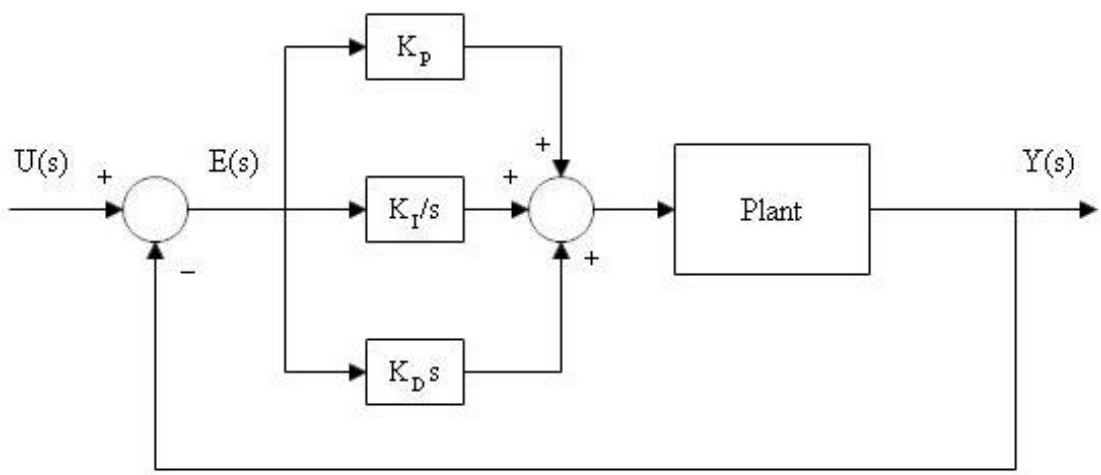


Figure II-2 – Block diagram of a PID compensator.

i. Ziegler–Nichols

The Ziegler Nichols method [6] is a technique for setting or tuning the sensitivity of a PID controller's reaction to measured error that developed out of the practice-based theory of the instrument companies and the process controls realm. The Ziegler-Nichols tuning techniques and the suitability of the method to be adapted into an auto-tuning framework contributed to the prevalence of PID control as the most widely used strategy in industrial feedback control applications [7]. The method involves a two-step process that begins with testing the response of the system to be controlled and using the resultant measurements from the test to compute the desired control parameters from a table of empirical equations [6] – [8].

In [6], Ziegler and Nichols detailed the foundation of their two-step process by plotting the step response of the open-loop system to determine key parameters related to the "lag time" and "reaction rate" evidenced in the system output. The tunable control settings of industry standard PID process controllers were then calculated based on the experimentally determined relationship to the measured step response characteristics. The concepts of "dead time" and "rise time" that are currently pervasive in controls literature are used in some tutorials to determine the required tuning parameters; these concepts can be easily related to the lag time and reaction rate characteristics used by Ziegler and Nichols. Figure II-3 depicts the hypothetical step response of a system, and the formation of a line drawn tangent to the maximum rate of change of the system response (i.e. tangent to the inflection point). The slope of the tangent line, R, is referred to as the reaction rate. The x-intercept of the tangent line, L, is referred to as the lag time.

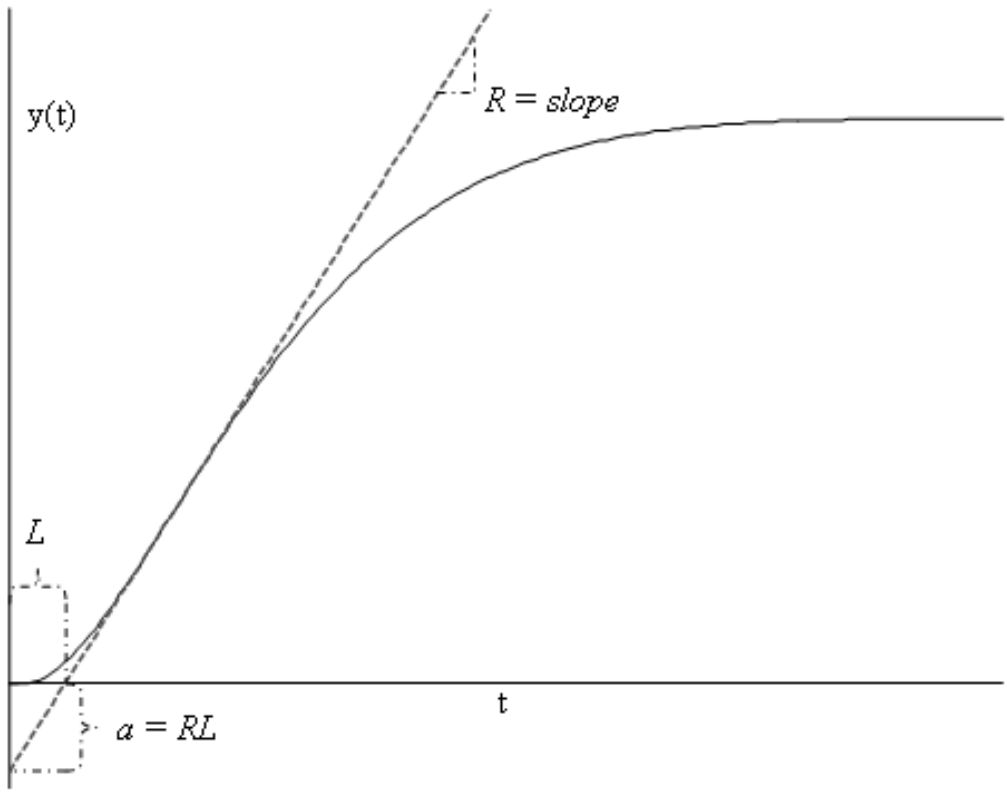


Figure II-3 – System step response depicting lag time (L) and reaction rate (R).

The Ziegler-Nichols tuning method can also be directly applied to the PID controller framework described above in section *B*. One begins by considering an equation for a PID controller, (2), that is slightly modified from the form listed in (1).

$$G_c = K_p(1 + \frac{1}{T_i s} + T_d s) \tag{2}$$

In this modified form, the parameters to be determined include the proportional gain, K_p, the integral time constant, T_i, and the derivative time constant, T_d. One may note that the relationship between these parameters and the parameters of the form listed in equation (1) is simply:

$$K_p = K_p,\ K_I = \frac{K_p}{T_i}, \text{and } K_D = \frac{K_p}{T_d} \tag{3}$$

Following the four-step design process detailed in [8], one begins the process with an open-loop test of the plant excited by a step input. From the measured system response, the line tangent to the point of inflection of the system response is drawn (this will be the steepest line tangent to the curve). The y-intercept of the tangent line, a, and its x-intercept, L, are determined. Lastly, the tunable parameters K_p, T_i, and T_d, are calculated from the heuristic relationships expressed in Table II-1. The relationship of the tunable parameters to the measured response characteristics effects a control design that may not be suitable for controlling systems where large overshoot is unacceptable due to the large control signal values and their tendency to lead to saturation typical of this tuning technique [8]. Figure II-3 displays the hypothetical step response of a system, the tangent line representing the maximum rate of change of the system, and the requisite parameters L and a, the x and y-intercepts of the tangent line, respectively. The reader may note that the parameter, a, is the product of the reaction rate and lag time parameters (R and L) previously discussed.

Table II-1 – Ziegler-Nichols parameters for PID controller tuning [8].

Parameter / Controller Type	K_p	T_i	T_d
P	$\frac{1}{a}$		
PI	$\frac{0.9}{a}$	3L	
PID	$\frac{1.2}{a}$	2L	$\frac{L}{2}$

ii. Root Locus Methods

A root locus analysis of a control system evaluates how the system roots, poles and a variable closed-loop gain affect the stability of a classical system represented by a transfer function. When combined with a set of desired performance characteristics, root locus analysis can be used to determine the parameters for a feedback control system, including the three control parameters required of a PID controller. Root locus analysis results in a plot of the real and imaginary coordinates of the poles and zeros of a system's transfer function as well as a curve plotting the locus of all possible values for the system poles— assuming that the system loop was closed with negative feedback and a variable gain, K. By examining the locus of all possible system poles as K is varied from 0 to ∞, one can determine the relationship between gain and stability for the system. Values of K at which the root locus crosses into the right-half plane result in an unstable system.

Given a system modeled by a transfer function $GH(s)$, a diagram of the closed loop system with negative feedback and variable gain, K, is shown in Figure II-4.

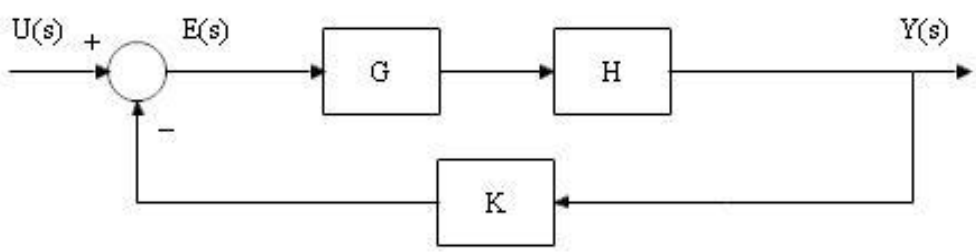


Figure II-4 – System GH(s) with negative feedback of variable gain K.

A root locus plot for the system in Figure II-4 begins with a graph of the poles and zeros of the open loop transfer function in the complex plane. Given any complex number, $s = \sigma + j\omega$, its coordinates can be graphed in the Cartesian plane with the x-axis representing all possible values for the real component, σ, and the y-axis representing all possible values for the complex component, ω. An O is placed at the coordinates of all

system zeros and an X is placed at the coordinates of all system poles. A plot of the locus of all possible closed loop system poles (the roots of $1 + KGH(s) = 0$) as the gain K is varied from 0 to ∞ can be generated according to the 10 rules listed below. Construction of a root locus plot as described below relies on the Routh stability analysis discussed in the Routh-Hurwitz Stability section.

Upon completion of the root locus analysis, a PID compensator can be designed so the closed loop system will meet steady state performance criteria as described in the PID Controller Design Based on Root Locus Methods subsection. A comprehensive example of the design of a continuous-time PID control law for a simple system follows the methodology. Subsequently, a discussion of the extension of the continuous-time PID design techniques to the discrete-time domain is presented in the Discrete Time PID Controller Design subsection. This chapter concludes with a subsection on compensator design in the Frequency Domain.

Ten Rules For Constructing a Root Locus Plot

I. The root locus starts at the open loop poles when $K = 0$.

II. The root locus ends at the open loop zeroes when $K = \infty$.

III. There are n root locus branches, where n represents the number of system poles.

IV. The root locus is symmetric about the real axis, σ-axis. The root locus is also symmetric about any other axes of symmetry formed by the configuration of the open loop poles and zeroes.

V. As the root locus approaches the open loop zeroes at $s = \infty$, the angle of approach is defined by the following angle condition:

$$\angle|_{s=\infty} = \frac{180° \times odd\#}{n - m} \tag{4}$$

where n and m denote the number of closed loop poles and zeroes, respectively, of the characteristic polynomials of the transfer function KGH, where $KGH(s) = K\frac{s^m + bs^{m-1} + \cdots}{s^n + as^{n-1} + \cdots}$.

VI. The asymptotes of the root locus intersect the real axis at:

$$\sigma = \frac{b - a}{n - m} \tag{5}$$

where b, a, n and m are determined from the characteristic polynomial defined in rule V.

VII. The root locus exists on the real axis if the total number of open loop poles and zeroes to the right is odd.

VIII. The angle of arrival / departure of the root locus at the zeroes / poles is defined as:

$$\angle zeroes - \angle poles = 180° * odd\# \tag{6}$$

IX. Routh's array is used to determine for which values of the gain, K, the system becomes marginally stable with the root locus crossing the imaginary axis. The points at which the root loci cross the imaginary axis occur by solving

$$1 + KGH(s) = 0 \tag{7}$$

X. The break-in and break-away points are determined by setting $\frac{\partial k}{\partial s} = 0$, with $K = \frac{-1}{GH(s)}$.

Routh-Hurwitz Stability

The Routh stability criterion allows for analysis of the stability of a closed loop system by classifying a system's poles instead of calculating them [9]. In order to determine stability, a Routh table is constructed using the method detailed below.

Given a system with the closed loop transfer function: $GH(s) = \frac{N(s)}{a_n s^n + a_{n-1} s^{n-1} + \cdots}$, the system's poles are examined and classified by forming a Routh table from the polynomial in the denominator of the transfer function. The Routh table is constructed based on the coefficients of the polynomial and a series of negative second-order determinants. Table II-2 shows a generalized Routh table for an nth order polynomial based on the equations described in [10].

Table II-2 – General Routh table constructed for a nth order polynomial.

s^{n}	a_{n}	$a_{\mathrm{n-2}}$	$\cdots$	a_0
$s^{\mathrm{n-1}}$	$a_{\mathrm{n-1}}$	$a_{\mathrm{n-3}}$	$\cdots$	a_0
$s^{\mathrm{n-2}}$	$\frac{-\begin{vmatrix} a_{\mathrm{n}} & a_{\mathrm{n-2}} \\ a_{\mathrm{n-1}} & a_{\mathrm{n-3}} \end{vmatrix}}{a_{\mathrm{n-1}}} = b_1$	$\frac{-\begin{vmatrix} a_{\mathrm{n}} & a_{\mathrm{n-4}} \\ a_{\mathrm{n-1}} & a_{\mathrm{n-5}} \end{vmatrix}}{a_{\mathrm{n-1}}} = b_2$	$\cdots$	$\frac{-\begin{vmatrix} a_{\mathrm{n}} & 0 \\ a_{\mathrm{n-1}} & 0 \end{vmatrix}}{a_{\mathrm{n-1}}} = 0$
$s^{\mathrm{n-3}}$	$\frac{-\begin{vmatrix} a_{\mathrm{n-1}} & a_{\mathrm{n-3}} \\ b_1 & b_2 \end{vmatrix}}{b_1} = c_1$	$\frac{-\begin{vmatrix} a_{\mathrm{n-1}} & a_{\mathrm{n-5}} \\ b_1 & b_3 \end{vmatrix}}{b_1} = c_2$	$\cdots$	$\frac{-\begin{vmatrix} a_{\mathrm{n-1}} & 0 \\ b_1 & 0 \end{vmatrix}}{b_1} = 0$
$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$
s^0	$\frac{-\begin{vmatrix} x_1 & x_2 \\ y_1 & y_2 \end{vmatrix}}{y_1} = z_1$	0	$\cdots$	0

The number of poles located in the right-half plane is then determined by the number of sign changes in the first column of values in the Routh table. A transfer function that produces a table without any sign changes in the first column is stable.

PID Controller Design Based on Root Locus Methods

The transfer function of a PID controller was defined in (1) as

$$GPID(s) = K_P + \frac{K_I}{s} + K_D s = \frac{K_D s^2 + K_P s + K_I}{s} \tag{1}$$

The control parameters are determined according to the following method.

I. Develop a set of desired transient and steady state error specifications based on the application and the underlying system type (number of open loop poles located at the origin). From these specifications, determine a complex conjugate pair of closed loop dominant poles, s_{max} and s_{max}^*, that meet these specifications.

II. Calculate K_I from the steady state error specification, ess.

III. Evaluate the combined system transfer function at the dominant pole and set it equal to zero to solve for the values of K_P and K_D.

$$\begin{gathered} 1 + G_{PID}G(s_1) = 1 + \left(K_P + \frac{K_I}{s_1} + K_D s_1\right) G(s_1) = 0 \\ \Rightarrow K_P + K_D s_1 = -\frac{1}{G(s_1)} - \frac{K_I}{s_1} \end{gathered} \tag{8}$$

Decompose the equation above into the real and imaginary terms. Solve for K_D from the imaginary component and then solve for K_P.

An example of this design method is presented in the next section.

Design of a Continuous-Time PID Controller

Given a system defined by the transfer function $(s) = \frac{10}{s+4}$, this section presents the design of a PID compensator to meet the following design criteria:

1. Settling time, $t_s = 0.5$ seconds
2. Damping coefficient, $\zeta = \sqrt{\frac{1}{2}} \approx 0.707$
3. Steady-state error, $ess|_{ramp} = 0.1$

The method for PID compensator design outlined is applied below.

I. The desired settling time and damping coefficient for the compensated system can be used to solve for the desired dominant poles.

$$s = \zeta\omega_n \pm \omega_n\sqrt{\zeta^2 - 1} \tag{9}$$

$$t_s = -\frac{4}{\zeta\omega_n} = -\frac{4}{Re(s_{\max})} \tag{10}$$

$$\Rightarrow Re(s_{\max}) = -\frac{4}{0.5} = -8$$

$$\Rightarrow Im(s_{\max}) = \pm\left(\frac{8}{\sqrt{\frac{1}{2}}}\right)\sqrt{\left(\sqrt{\frac{1}{2}}\right)^2 - 1} = \pm j8$$

$$\Rightarrow s_{max} = -8 \pm j8$$

II. The compensated system is of type 1 due to the pole at s=0 introduced by the integrator term of the PID compensator. The desired steady state error condition can then be used to solve for the integrator gain, K_I.

$$ess|_{ramp} = \lim_{s\to 0} s\frac{\frac{1}{s^2}}{\left(1+G(s)G_{PID}(s)\right)} = \lim_{s\to 0}\frac{1}{s\left(1+G(s)G_{PID}(s)\right)} \tag{11}$$

$$\Rightarrow ess|_{ramp} = \lim_{s\to 0}\frac{1}{s\left(1+\frac{(10)}{s+4}\left(K_P+K_D s+\frac{K_I}{s}\right)\right)} = \frac{4}{10K_I} = 0.1$$

$$\Rightarrow K_I = 4$$

III. Using the value of K_I solved for in part II, and selecting $s_1 = -8 + j8$ as one of the dominant poles from part I, the combined system transfer function is evaluated.

$$K_P + K_D s_1 = -\frac{1}{G(s_1)} - \frac{K_I}{s_1} \tag{12}$$

$$\Rightarrow (K_P - 8K_D) + j(8K_D) = -\frac{1}{\frac{10}{-4+j8}} - \frac{4}{-8+j8}$$

$$= 0.4 - j0.8 + 0.25 + j0.25 = 0.65 - j0.55$$

$$\Rightarrow K_D = -\frac{0.55}{8} = -\frac{11}{160} = -0.06875$$

$$\Rightarrow K_P = 0.65 + 8K_D = 0.1$$

Thus, $G_{PID}(s) = \frac{1}{10} - \frac{11}{160}s + \frac{4}{s} = \frac{-11s^2+16s+640}{160s}$

A root locus plot of the resultant PID compensated system is shown in Figure II-5.

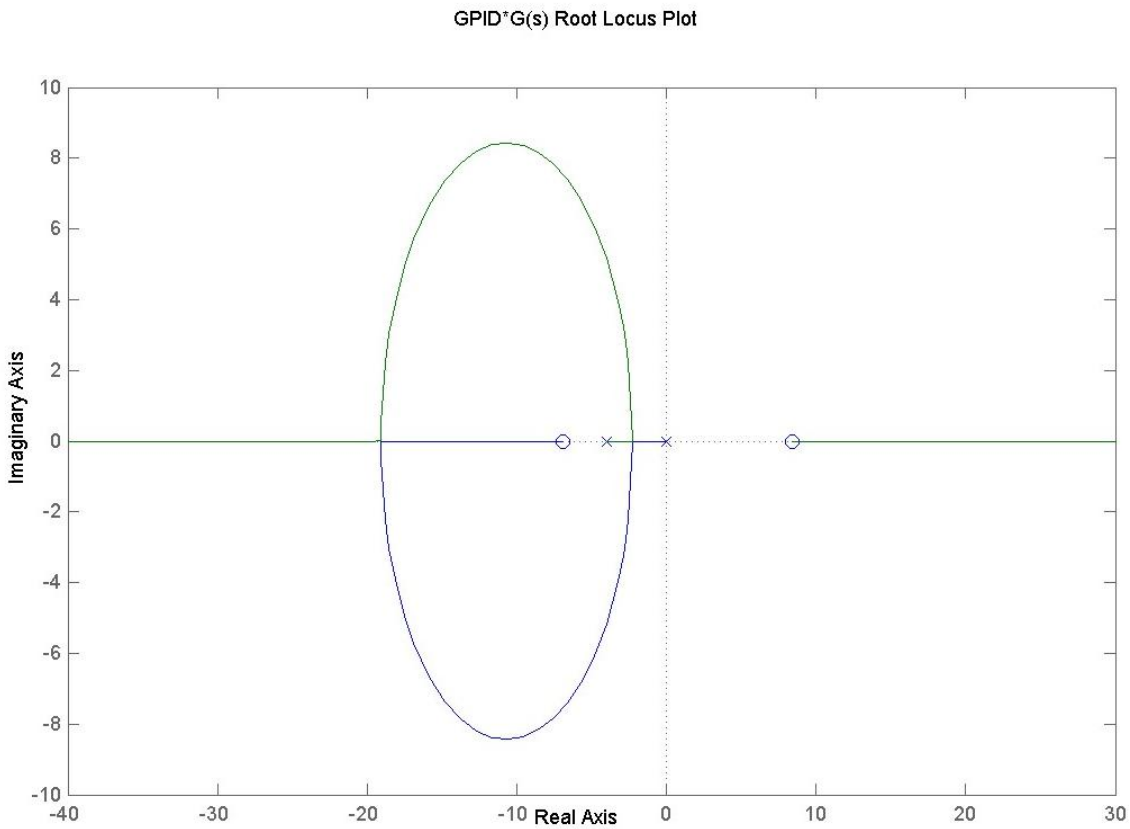


Figure II-5 – A root locus plot of the example system with PID feedback control.

Discrete Time PID Controller Design

The previous sections detail the application of root locus based PID control design techniques in the continuous time domain, the s-domain. One technique for addressing controller design for use in the digital realm is to follow the outlined procedure for s-domain PID controller design and then convert the model into the discrete-time domain, the z-domain, using an appropriate approximation. One such approximation is known as the bilinear z-transform and is given by

$$s = \frac{2}{T}\frac{z-1}{z+1} \tag{13}$$

Depending on the application, other approximations such as a zero-order hold, ZOH, may be used when appropriate.

iii. Frequency Domain

The root locus design method characterizes the system response in the time-domain based on an analysis of the system poles and zeroes. Alternatively, analysis of the frequency response of both the plant and controller in terms of gain and phase can be used to design a PID controller to meet a desired set of performance specifications.

The PID design procedure in the frequency domain relies on the analysis of the frequency response of the system to be controlled. Two parameters that characterize the stability of the system in the frequency domain can be defined. The gain margin (GM) of a system is defined as the amount of gain that can be applied before a closed loop system becomes marginally stable. The phase margin (PM) of a system is defined as the amount of additional lag phase that can be added before the closed loop system becomes marginally stable.

This frequency response can be represented graphically in the form of a Bode plot. A Bode plot is a chart of the magnitude in decibels (dB) and the phase angle in degrees of the system response versus the frequency of the system input on two distinct graphs. From the Bode plot, the aforementioned stability parameters can be determined. The first parameter, the gain margin (GM), is defined as the distance of the magnitude below 0dB when the phase angle is -180°. The phase margin (PM) is defined as the distance of the phase angle above -180° when the gain is 0dB. The frequency at which the gain margin is calculated is known as the phase crossover frequency, ω_{cp}. The frequency at which the phase margin is calculated is known as the gain crossover frequency, ω_{cg}. These parameters are related to the classical parameters of a second-order system as defined by (14) given in [9].

$$PhaseMargin = \tan^{-1}\frac{2\zeta}{\sqrt{-2\zeta^2 + \sqrt{1+4\zeta^4}}} \tag{14}$$

Using the parameters and the relationships defined in the equation above, a PID controller can be designed by selecting a gain to meet the steady-state performance specifications and determining the other parameters from the amount of phase lag that can be added by the PID controller without making the resultant system unstable.

Copyright © David Michael Feinauer 2011

III. MOTIVATIONS FOR AN ALL-DIGITAL PID COMPENSATOR

In this chapter, the dominant role of PID-based control for industrial applications and the emergence of controllers based on programmable logic technology are explored. Challenges and opportunities that exist in both domains are described in section C. The limitations of a particular application of prior art are catalogued. The discussion ultimately reveals the impetus for the all-digital proportional-integral-derivative model described throughout the remainder of this work.

A. PID PREVALENCE IN INDUSTRY

It has been observed that PID control is one of the most prevalent control methodologies applied in industry today despite decades of advancements in modern control theory and the availability of many sophisticated, modern control technologies [11] and [12]. As previously mentioned, there are estimates that billions of new PID control loops are installed annually [3]. In the realm of process controls, it has been asserted that more than 95% of control applications still use either PI or PID control [13]. The simplicity of designing PID controller parameters is cited as a reason for PID's continued industry prevalence, given the control theory expertise gap that sometimes exists among the end users [4]. In a survey of Japanese industry [12] it was found that PID control and many advanced implementations of PID control were widely applied with high user satisfaction.

In spite of its nearly 80 year existence, PID control remains poorly understood in many applications which contributes to poor system performance. Notwithstanding the previously mentioned simplicity and rigorous development of PID control tuning techniques, complex issues such as output saturation as a result of "integrator windup" and noise induced by the differentiator combined with implementation and design complications can lead to control loop designs with undesirable performance. To address these complications, a number of auto-tuning techniques that are prevalent and simple

have resulted [3]. In a 2005 survey of controller customers, the availability of auto-tuning was specified as one of the most important features sought in a commercial controller [7]. In [13], O'Dwyer motivates the need for a more unified PID framework and notation to address the lack of understanding and the performance issues; he points out the existence of at least 408 separate sources of loop tuning rules [13]. In [5], the myriad of loop tuning rules were classified into five categories: analytical, heuristic, frequency response, optimization and adaptive tuning methods. It was also noted that nearly 80 international patents exist with specific reference to PID loop tuning methods [5]. The ability for successful application of a tuning technique can be further complicated in applications where control theory is applied for optimizing multiple objectives

B. Trends in Industrial Control Implementation

In a survey of Scottish industry [11], Hersh found that the perceived economic factors of implementing automated control systems in an industrial setting may lead to an avoidance of them altogether—motivating a need for further development of cheap and reliable commercially available controllers. In the automotive and aerospace realms, or for any safety-critical application, industry standards require rigorous verification of control techniques, laws, implementations and tuned parameters; such verification costs often constitute more than 50% of the cost of developing the system [14]. The expensive process is very labor intensive and is often costly in terms of time-to-market as well. The ubiquitous nature of general purpose, one-size-fits-most microprocessors among many modern control implementations can also compound the problem. The reliability verification of a general purpose processor implementing numerous safety-critical control loops can be prohibitively complex and costly. Systems realizing safety or critical processes on the same processor and hardware as lower priority tasks require verification of the isolation of the subsystem in response to a fault elsewhere and the maintenance of priority for the critical system in addition to requisite overall system reliability verification. However, the development of custom digital devices for each subsystem can also be extremely expensive and time-consuming [15].

For designs using commercial-off-the-shelf processors that have invested in the requisite reliability verification, processor obsolescence creates a complex predicament for design engineers that can be costly to solve. Consumer products are often made obsolete rapidly and calculatedly to spur future sales, driving the underlying processors and general purpose digital components into obsolescence [16]. This compounds the verification hurdles present for safety-critical systems which typically have a much longer design life, in part due to the already high development costs. FPGAs utilizing "soft core" microprocessors exist as a hybrid digital implementation technology that is a general purpose, off-the-shelf component that has a microprocessor core that is "soft developed" as an intellectual property that may persist independent of a particular hardware platform or span multiple technology generations [16]. The high-level descriptions of non-processor based digital systems for implementation in programmable logic devices (PLDs) can be developed in such a way that they are also independent of specific hardware architectures, thus reducing the potential for obsolescence issues.

Recent trends reflect an increased adoption of PLDs in motion control, power electronic, communications and signal processing applications [17] and [18]. A survey of the use of PLDs for the various components of motor control (i.e. excitation circuits, feedback tachometry, torque control, etc.) is presented in [17]. Novel implementations of PLD based controllers relative to motor control including induction machine drives [19], switched reluctance machine drives [20], permanent magnet motors [21] , and precision equipment such as atomic force microscopes [22] exist and are appearing more frequently in the industrial electronics and control systems literature. PLD manufactures are recognizing the role of their devices in the realm of motion control and are beginning to offer software development tools that allow for co-simulation of system designs in standard hardware description language (HDL) simulation packages along with *MATLAB* and *Simulink*, a software package frequently used by control systems engineers [17].

Technologies enabling the reduction of size (increase of component density) and the distributed implementation of systems and their components drive the development of system-on-a-chip (SoC) devices and micro-electro-mechanical systems (MEMS) which compels improvements to the cost, density and reliability of controllers for the systems and also requires improvements to precision controllers for the machines that fabricate the micro-sized devices [23]. Increased density of component integration in digital hardware combined with decreasing costs and highly developed design and simulation tools for programmable logic devices has led to their increased adoption for control engineering applications [17]. Complex programmable logic devices (CPLDs), field-programmable gate arrays (FPGAs) and application specific integrated circuits (ASICs) afford varying opportunities for implementing controllers. One could consider FPGAs as generic digital logic devices that can be configured and re-configured to implement user specified digital systems. One could consider ASICs as built-to-order digital logic devices that implement engineered digital systems. The implementation of systems on an FPGA as opposed to an ASIC complements an iterative development process allowing for reprogramming, re-use and lower cost hardware design verification. Selecting an ASIC over an FPGA affords the designer a more minimal hardware implementation as the devices are built with only the required components and functionalities.

PLDs offer a number of design benefits including: privacy of proprietary architectures and components, improved performance due to the parallel nature of logic hardware resources and reduced development costs due to the ability for the design to be fully simulated and tested before being implemented with a minimal required set of hardware components [18]. In addition to these enumerated benefits, innovations in the manufacture of FPGAs have led to products that are reconfigurable in real time with integrated components beyond the programmable logic blocks. FPGAs are available with multiple integrated hard and soft core processors, RAM, signal processing and arithmetic functions as well as the integration of a small number of analog devices such as analog-to-digital converters (ADCs), improving the growth of SoC devices [18].

C. Proposed Problem Description

An array of solutions for implementing digital control systems based on microprocessor and digital signal processor (DSP) technologies exist. These implementations rely on the core processor for input sampling, data storage and serial processing of the sampled and stored data for multi-component control laws as well as the actuation of a resultant controller output. The controller output is often realized in the form of a pulse-width modulated signal due to the number of low cost implementations available. The pulse-width modulated signal is actually an analog signal realized through a number of standard circuits based on the type of modulation desired. More and more, plants or devices to be controlled are of a truly digital nature, requiring PWM input signals and providing digital output signals from a digital transducer inherent in the digital system.

Building on the apparent proclivity for digital-based controller designs, their popular use with pulse-width modulation schemes and the design of many digital plants to accept PWM inputs, one might ask if there is a way to realize the same pulse train actuation signal directly—in an all-digital manner. Additionally, recognizing the digital output resultant from a class of systems with inherent digital transducers generating output signals, one may also question whether there is a way to realize a controller whose reference input signal is driven directly by the pulse train one would expect from a digital transducer.

The prevalence and widespread recognition of proportional-integral-derivative (PID) based control technologies in industrial applications, the need for cheap, reliable and certifiable control technologies and the advantages offered by custom programmable logic devices discussed in the previous section motivate a further need for improvements in the realm of digital controller design. To address these factors, an all-digital implementation of a PID compensator is proposed. Offloading the sometimes greedy process of input sampling and serial computation of parallel, multi-objective control laws to a custom digital device presents itself as one manner of addressing the design

restrictions imposed by the current bounds of the state-of-the art. Reducing the overall real estate needed for a system by incorporating a digital formation of the pulse-width modulated signal in the custom digital device provides additional benefits. The compound adjective “all-digital” is used to designate a system design and implementation that is entirely realizable with a range of digital logic elements.

The application of phase-locked loop (PLL) techniques has been extended from the realm of communication systems to motor control techniques and they have shown to be reliable in motor speed control applications [24]. PLL devices were initially adopted for widespread use in communication systems as they allow for an output signal to track a reference signal in both frequency and phase [25]. PLLs are typically implemented as mixed systems of both digital and analog components because of the difficulty of realizing digital versions of the high frequency components [25]. In the application of PLLs loops to motor control, a novel all-digital PLL control methodology has been previously proposed and implemented [26] and [27]. It was found that although achieving steady-state tracking, the ADPLLs exhibited poor transient performance with respect to the desired specifications. It is proposed that similarly, an all-digital implementation of the widely recognized PID controller can be realized and that the multi-objective control law formed from the proportional, integral and derivative components can address transient response requirements for the closed-loop system.

Implementation of the concepts of proportional gain, integration and differentiation in the form of all-digital logic hardware requires re-envisioning a PID control law for more simplified computation and reduced complexity. The all-digital PID controller proposed would rely on the use of digital counting techniques to convert a classical PID-based compensation method from the time-domain to the frequency domain. The initial approach for an all-digital PID solution will stem from the prior graduate research described in [28]. The framework proposed in [28] is bounded by a number of factors including but not limited to its:

- restriction of the control scheme to a finite state machine-only model;
- basing the model on a 4-bit counting scheme that would rapidly reach saturation when using counting as a crude form of sampling to represent the frequency information of a signal;
- modeling of the control law around input and reference signals whose "information content" resides in the frequency domain without addressing the concept of lag among the signals;
- deviating from the core purpose and benefit of an all-digital implementation through the acknowledgement that frequency lag would need to be addressed with analog methods;
- developing a simulation model of the concept that is restricted in terms of simulating and analyzing the internal dynamics and signal routings of the proposed system;
- developing the model in a manner that it is not reusable for further investigation with respect to its application to a variety of systems to be controlled;
- and neglecting the effects of the all-digital controller on the stability of the closed-loop system under control as well as neglecting to discuss the stability effects of a classical PID controller and the motivations for implementing one.

Throughout the body of this work, I will address these issues and advance the proposed concept and related design methodology by:

- capitalizing on the inherent operating states of a binary counter to map the digital controller design to a realization that can be described in a causal block diagram-like manner;
- simplifying the number of operating modes of the device as it relates to the implementation of the constituent components of a PID control law;
- improving the resolution of the counting functionality of the device;

- devising a logic based method for addressing lag between the input and reference signals;
- forming the above method in an all-digital manner;
- developing a simulation model of the concept that is far less restricted in terms of simulating and analyzing the internal dynamics and signal routings of the proposed system;
- implementing the model in a modular fashion for reuse with a number of applications;
- and discussing methods for determining the affect of the all-digital PID model on the underlying stability of the system to be controlled.

In addition to addressing the issues as described above, I will:

- compare the all-digital PID controller to the classical PID reference system as a baseline for analysis;
- develop the model with the goal of an eventual digital logic implementation of foremost concern throughout the design selection process;
- develop the model for simulation in The MathWorks *MATLAB* and *Simulink* packages to aid in suitability and stability analysis of the proposed system with respect to standards in the control systems engineering domain;
- implement additional models of hardware interface components to allow for more realistic simulation of the controller and the system under control;
- and investigate the performance of the system with respect to the presence of disturbances and the effect of higher order poles in the systems to be controlled—typically unmodeled or neglected in theoretical systems.

The proposed all-digital PID (ADPID) controller framework is to be designed at a high level of abstraction— independent of a specific hardware platform, keeping the goal of an all-digital implementation in mind such that it could be realized with an FPGA, ASIC, or discrete digital logic components. A PID design framework of this nature couples the widely accepted PID controller with the advanced design and simulation frameworks of PLD-based systems for an implementation that has a minimal hardware footprint without relying on a PC or microcontroller. The parallel nature of a logical hardware implementation complements the "parallel" form of a PID-based compensator where the resulting control law is formed as the summation of the three constituent control components as opposed to a microprocessor based PID control where the control law must be implemented and applied serially. Additionally, by simplifying and offloading the PID control law computation tasks from a processor to digital hardware, one could realize a number of benefits including: simplified reliability verification due to the decoupling of the system from a multipurpose processor, reduced demand for processor bandwidth and higher sampling rates of the systems interfaced to the controller due to the dedicated hardware.

An ASIC-based implementation of the controller framework would allow for a compact, minimal controller implementation that could be available as a commercial-off-the-shelf (COTS) component or as an embedded subsystem in a larger device. A COTS implementation could further the cost effectiveness and market penetration of basic, feedback control in industrial and commercial applications. Implementing the model on an FPGA platform allows for an extension of the controller to fault recovery and self tuning techniques due to the dynamic reconfiguration and processor cores availed by the platform. Developing the model at a high level allows for an extension of the design for implementation in a MEMS or SoC device.

Chapter IV details the design and functionalities of the proposed controller. Initial simulations of the model proposed in chapter IV revealed a need for further design modifications. The simulation results and the revisions they motivated are discussed in the beginning of chapter V. The details of the implemented design modifications and the control characteristics of the ADPID compensator as applied to a number of example systems are presented throughout the remainder of chapter V.

Copyright © David Michael Feinauer 2011

IV. PROPOSED SOLUTION

The all-digital PID controller detailed in this section uses digital counting techniques to convert a classical PID-based compensation method from the time domain to the frequency domain with an implementation based solely on digital circuit elements. The initial solution modeled is based on the foundations for the design proposed in [28]. The finite state machine model proposed in [28] was modified for simulation based on the use of edge detectors to determine transitions from the inherent states or modes of operation of the core digital component of the design, a synchronous binary up/down counter with asynchronous preset. The model was implemented to allow for the completion of an analysis of the proposed system, and for access to a richer set of analysis tools from the control theory domain. The model was also created in a modular, parameterized fashion to allow for re-use and application to additional research investigations.

The single-input, single output (SISO) model is designed to operate on the class of systems whose controller/plant interface is pulse-width modulated and whose plant/controller interface signal is a square wave—like that of an incremental encoder or other sensor. The model "converts" the signals from the time-domain to the frequency domain by characterizing the error between a reference and a feedback signal in terms of the sign of the error and the duration of the magnitude of the error. The duration (or pulse width) of the magnitude of the error between the signals is measured by an incremental binary counter. The speed at which counts occur is determined by a counting frequency parameter. In forming a proportional-integral-derivative controller, each constituent component operates on the error signal in parallel, measuring the duration of the error relative to each particular stage at a specified counting frequency. The overall control law is determined by accumulating the information from each of the stages and measuring the result of the accumulation with a counter incrementing at a distinct, base counting frequency. The ratio of the counting frequency of a given stage to the base counting frequency is related to the traditional PID control parameters or gains. In the subsequent

sections of this chapter, assumptions regarding the interface of the controller to a typical plant will be explained, the all-digital controller framework and its constituent components will be proposed, and the design procedure for determining and implementing the control parameters will be presented. Following this discussion, an abstract block diagram illustrating all of the system components and their interactions is depicted.

A. ALL-DIGITAL CONTROLLER / PLANT / CONTROLLER INTERFACE

In keeping with the all-digital design objective, the controller design framework presumes that the measurable outputs from the plant to be controlled are the result of a tachometry device such as an optical encoder. Optical encoders are sensors that are widely used to detect linear and rotary positions. They utilize photoelectric elements to detect the motion of an indexed scale with respect to the photo element and produce an oscillating output signal that corresponds to the motion of the scale. For motors, a rotary optical encoder can be used to measure angular rotation or position by attaching the encoder scale to the shaft of the rotating motor or other rotating element in the drive train. In 1999, it was found that position encoders represented the third most utilized sensor in terms of number of sales [29] in the automotive industry.

In addition to the assumption of a digital encoder-based output from the system to be controlled, the proposed model similarly presumes that the input to the plant under control is implemented as a pulse-width modulated signal. The resultant pulse train from the encoder output is then fed into the proposed controller framework. The use of binary counters and other basic digital circuit components effectively converts the time domain-based output of the plant into the frequency domain. The next section will detail the digital implementation, the conversion of the system output through counting and the parameters that can be varied to affect control in the system. The formulation of the PWM output signal that actuates the plant under control will also be detailed. Figure IV-1

depicts block diagrams of a system with optical encoder output under control by both a typical discrete-time PID controller and the proposed ADPID controller.

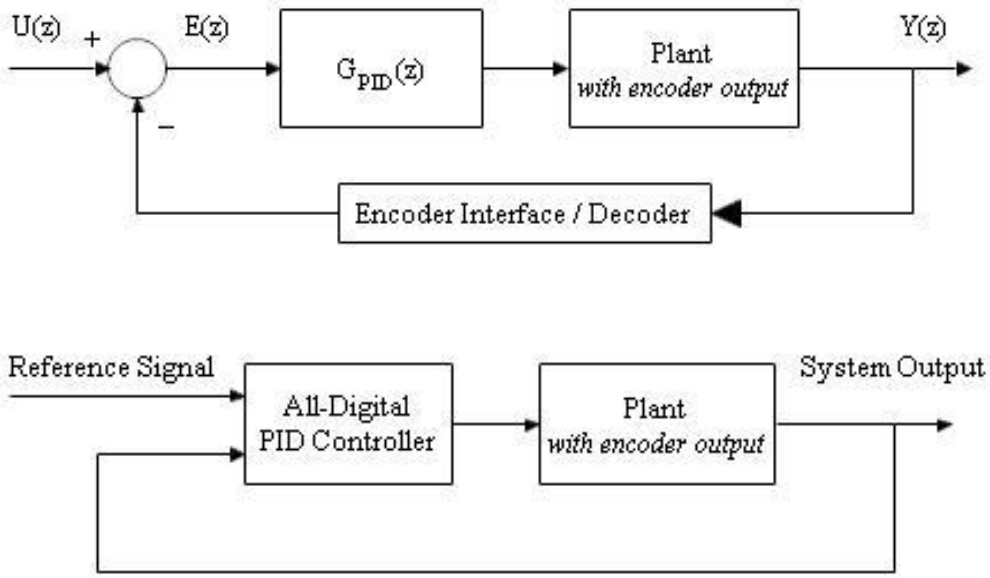


Figure IV-1 – Block diagram of a typical plant under control by a discrete PID controller and the proposed all-digital PID controller.

B. ALL–DIGITAL PID CONTROLLER DESIGN FRAMEWORK

Ultimately, the implementation of any proportional-integral-derivative (PID) based controller would generate three signals: one proportional to the system error, another proportional to the accumulation of the errors over time and a third proportional to the rate of change in the error over time. The three signals are then combined to generate a control signal. The same principle components exist in the framework proposed herein. Under the proposed high level framework in Figure IV-1 above, a reference signal for the plant (or system under control) to track is compared to the actual system output. Thus, the reference signal represents the overall desired system output. The system error is defined as the difference between the two aforementioned signals. In the initial proposal of the concept [28], the error was decomposed into both a magnitude and direction signal. A

digital logic formation of the magnitude of the system error was proposed and the possibility of determining the sign of the error through analog techniques was proposed and never realized. For the initial implementation of the ADPID compensator, the most basic definition of signal error based on the difference between the signal values was applied. Figure IV-2 displays two example signals and the resulting error. In addition to the three signals mentioned above, the error signal is decomposed into both a magnitude and direction signal. The result is also shown in Figure IV-2, with the value 0 representing nonnegative error, and the value 1 representing negative error. This formulation was developed to complement the proposed implementation of the model. Note that under this framework, the system output is a pulse train of variable frequency, consistent with the output of an optical digital encoder.

Keeping in mind the larger overall framework in Figure IV-1 above, a high-level view of the all-digital PID controller shown in Figure IV-3 depicts the “parallel” counting components that generate the proportional, integral and derivative error signals and the accumulation of the signals to create the control signal.

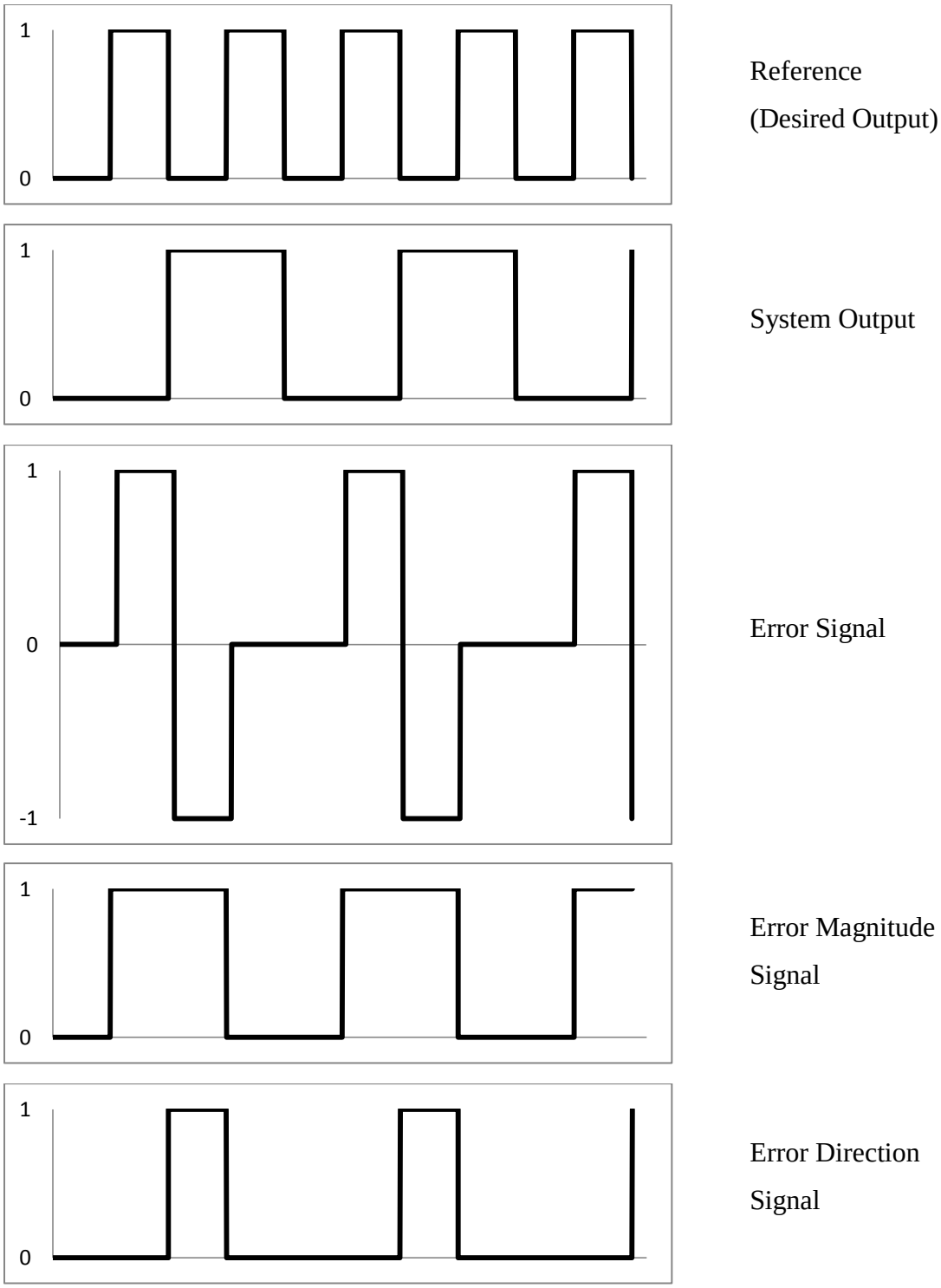


Figure IV-2 – Diagram of five pulse trains–reference, output, error, error magnitude and error direction.

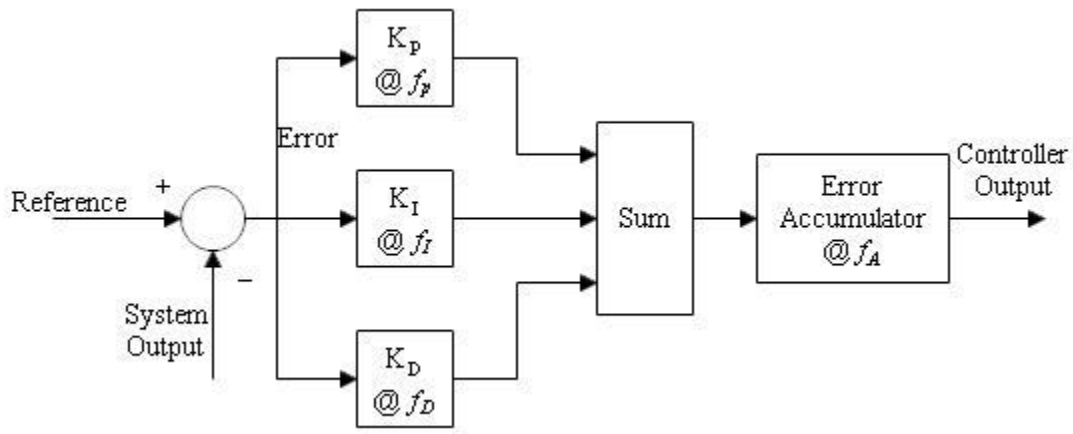


Figure IV-3 – High-level view of the constituent components of the proposed ADPID framework.

The principle component of the individual building blocks (K_P, K_I, K_D and the *Accumulator*) is a synchronous binary up/down counter. The error signal is fed into each of the constituent P, I and D components and they perform a counting operation on the error signal at speeds of f_P, f_I and f_D respectively. The counting operations combined with variations in when and how frequently the counters reset produces results representative of the three principle components of the compensator. The results of the three subsystems are added and the combined result is counted to zero at a frequency of f_A. This resultant signal is used to generate the PWM output of the controller. Analogs to the typical control parameters K_P, K_I and K_D can be achieved by allowing the control designer to tune the ratio of the counting frequencies f_P, f_I and f_D to the accumulation frequency, f_A. As in the case with classic PID controller implementations, each component is generated independently which allows for a subset of the proportional, integral and derivative components to be implemented if desired. In this model, it is important to note that the a*ccumulation* stage is always required to be implemented, even in the case of a proportional only controller (P control). It is also important to note that the proposed implementation of the proportional, integral and derivative terms are parallelized and thus, grounding the control parameter for one of the constituent PID

components (implying that the corresponding control parameter in a classical PID compensator has a gain equal to zero) effects a controller instantiation with a subset of the proportional, integral and derivative control filters implemented. For example, grounding the counting frequency f_D, allows for the implementation of a PI controller, which is frequently used instead of classical PID-based control due to issues related to differentiator noise that may result in poor performance.

To prove the concept of this all-digital control, the functional components were designed and simulated in The MathWorks *Simulink*® and *MATLAB*® Release 2010b packages. *MATLAB and Simulink* are frequently used packages with respect to control systems simulation. They are not commonly used for digital logic design. The models presented in the following subsections consist of either graphical model implementations relying on fundamental logic components or text based models that rely on functional descriptions of the model's performance. In both cases, the models were developed in such a way that they would resemble gate-level or functional-level implementations in a hardware description language such as VHDL. Hardware description languages are commonly used to design, simulate and implement custom or programmable digital logic hardware. The designs of the *Simulink* models were restricted to use constituent components with likely equivalents in the HDL realm in accordance with the stated goal of producing an all-digital design that could be implemented using standard commercial-off-the-shelf digital devices and programmable logic devices such as FPGAs.

Details about the implementations of the proportional, integral, and derivative subcomponents are presented in the following three subsections and the appendices referenced therein. Lastly, the digital circuitry to combine the error signals and generate a resultant PWM is detailed in the fourth subsection. Section C discusses the proposed methodology for determining the base counting frequencies.

i. Generation of a Proportional Error Signal

The core element of the proportional controller component, K_P, is an n-bit binary synchronous counter. The 74LS169 [30] is a readily available commercial-off-the-shelf discrete logic component that functions as a 4-bit synchronous binary up/down counter with load and reset functionalities. Four of the counters can be cascaded to produce a 16-bit counter. The *Simulink*-based implementation of the aforementioned 4-bit counter and its requisite components (i.e. a positive edge triggered J-K flip flop) as well as a cascaded, gate-level description of a 16-bit binary up/down counter can be found in Appendix A. A generic n-bit counter was also implemented as a functional description of the desired performance characteristics of an up/down counter. The implementation is provided in Appendix B. For the implementation of the ADPID compensator described in this work, the functional level description of the counter parameterized to 16 bits was used. This counter was selected due to its improved simulation performance when compared to the gate-level model developed. In later updates to the ADPID model, an integer based counter with implicit handling of signed numbers and without an implicit maximum count was selected due to the data type handling of various library blocks in *Simulink. The* generic counter described in Appendix B was modified to operate in this manner— supplying the value infinity (inf) in lieu of a binary bit length, N, allows for this operating mode. For the simulations presented in chapter V, it appears that an 8 to 10-bit binary counter would provide a sufficient numerical range for the counts that are formed in implementing the control law.

The operation of the K_P functional block is defined in terms of the modes of operation of the synchronous up/down counter and its associated inputs. The implementation of the proportional gain control function relies on the following operating conditions on the inputs and outputs of the counter in Figure IV-4:

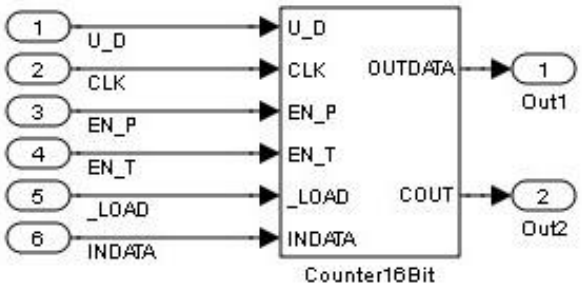


Figure IV-4 – Block diagram of 16-bit binary up/down counter.

I. CLK – *1-bit input* – A digital clock pulse that varies between 1 and 0 at a frequency of f_P. This frequency determines the speed at which the synchronous counter increments or decrements. The count will increase or decrease on each rising edge of the CLK if enabled.

II. U_D – *1-bit input* – Active high signal that sets the direction of the count. A logic 1 (high) value increments the count and a logic 0 (low) decrements the count. The U_D input is tied to logic 1 to force the counter to always increment.

III. EN_P – *1-bit input* – Active low input that enables the operation of the counter. Both EN_P and EN_T must be low in order for the count to increase. The EN_P input is tied to logic 0.

IV. EN_T – *1-bit input* – Active low input that enables the operation of the counter. Both EN_P and EN_T must be low in order for the count to increase. The EN_T input is tied to the inverse of the magnitude of the system error. The result is that as long as the error is non-zero, the counter is enabled and the count changes at frequency f_P.

V. _LOAD – *1-bit input* – Active low input that sets the value of the counter to the value available on input INDATA. The signal is connected to the inverse of an edge detector that is high when a positive edge occurs in the magnitude of the error. The result is that the integer value zero is loaded into the counter, resetting it; every time the error transitions from zero to non-zero, the counter is reset.

VI. INDATA – *16-bit input* – Data on this input can be loaded into the counter to set its value. This input is tied to the integer value zero to create a reset functionality that allows the count value to be reset to zero by loading the value into the counter.

VII. OUTDATA – *16-bit output* – This output represents the value of the 16-bit counter. The count value is the principle output of this functional block whose value is a numerical representation of the duration of the magnitude of the system error.

VIII. COUT – *1-bit output* – This output is an active low representation of the overflow, or carry-out, of the 16-bit counter. In incremental counting, this output will have a value of 0 when the counter exceeds the count of 65535 (maximum binary count of 16 consecutive 1's).

The result of the proportional control block with the signal operating modes described above is a numerical binary representation of the pulse width of the tracking error of the system. The output resulting from the counter is modified to reflect the sign of the error signal before it is accumulated with the other blocks in the K_A stage. A diagram showing the signal routing and operating mode for this functional block and the additional functional blocks that form the all-digital PID controller is shown in Figure IV-5.

ii. Generation of an Integral Error Signal

The core element of the integral controller component, K_I, is also a 16-bit binary synchronous counter. The operation of the K_I functional block is defined in terms of the modes of operation of the 16-bit counter and its associated inputs. The implementation of the integral gain control function relies on the following operating conditions on the inputs and outputs of the counter in Figure IV-4:

I. CLK – *1-bit input* – A digital clock pulse that varies between 1 and 0 at a frequency of f_I. This frequency determines the speed at which the synchronous counter increments or decrements. The count will increase or decrease on each rising edge of the CLK if enabled.

II. U_D – *1-bit input* – Active high signal that sets the direction of the count. A logic 1 (high) value increments the count and a logic 0 (low) decrements the count. A "binary" definition for a function that computes the sign of a number was constructed such that non-negative numbers map to a binary 0 value and negative numbers map to a binary 1 value. The U_D input is determined by the inverse of the error sign signal.

III. EN_P – *1-bit input* – Active low input that enables the operation of the counter. Both EN_P and EN_T must be low in order for the count to increase. The EN_P input is tied to logic 0.

IV. EN_T – *1-bit input* – Active low input that enables the operation of the counter. Both EN_P and EN_T must be low in order for the count to increase. The EN_T input is tied to the inverse of the magnitude of the system error. The result is that as long as the error is non-zero, the counter is enabled and the count changes at frequency f_I.

V. _LOAD – *1-bit input* – Active low input that sets the value of the counter to the value available on input INDATA. The signal is tied to the logic value 1. This prevents the counter from being reset to its initial integer value of 0, as the integral block represents the accumulation of error over the runtime of the system.

VI. INDATA – *16-bit input* – Data on this input can be loaded into the counter to set its value. This input is tied to the integer value zero as in the previous case.

VII. OUTDATA – *16-bit output* – This output represents the value of the 16-bit counter. The count value is the principle output of this functional block whose value is a numerical representation of the accumulation of the system error over the operating time of the system to be controlled.

VIII. COUT – *1-bit output* – This output is an active low representation of the overflow, or carry-out, of the 16-bit counter. In incremental counting, this output will have a value of 0 when the counter exceeds the count of 65535 (maximum binary count of 16 consecutive 1's).

The result of the integral control block with the signal operating modes described above is a numerical binary representation of the accumulation of the pulse widths of the tracking error of the system over the system's runtime. When the system is lagging the reference to track, the output of the functional component increases; when it is leading, the output decreases.

iii. Generation of a Derivative Error Signal

The core element of the derivative controller component, K_D, is also a 16-bit binary synchronous counter. The operation of the K_D functional block is defined in terms of the modes of operation of the 16-bit counter and its associated inputs. The implementation of the derivative gain control function relies on the following operating conditions on the inputs and outputs of the counter in Figure IV-4:

I. CLK – *1-bit input* – A digital clock pulse that varies between 1 and 0 at a frequency of f_D. This frequency determines the speed at which the synchronous counter increments or decrements. The count will increase or decrease on each rising edge of the CLK if enabled.

II. U_D – *1-bit input* – Active high signal that sets the direction of the count. A logic 1 value increments the count and a logic 0 decrements the count. The U_D input is tied to logic 1 to force the counter to always increment.

III. EN_P – *1-bit input* – Active low input that enables the operation of the counter. Both EN_P and EN_T must be low in order for the count to increase. The EN_P input is tied to logic 0.

IV. EN_T – *1-bit input* – Active low input that enables the operation of the counter. Both EN_P and EN_T must be low in order for the count to increase. The EN_T input is tied to the inverse of the magnitude of the system error. The result is that as long as the error is non-zero, the counter is enabled and the count changes at frequency f_D.

V. _LOAD – *1-bit input* – Active low input that sets the value of the counter to the value available on input INDATA. The signal is connected to the inverse of an edge detector that is high when a positive edge occurs in the magnitude of the error. The result is that the integer value zero is loaded into the counter, resetting it; every time the error transitions from zero to non-zero, the counter is reset.

VI. INDATA – *16-bit input* – Data on this input can be loaded into the counter to set its value. This input is tied to the integer value zero as in the previous case.

VII. OUTDATA – *16-bit output* – This output represents the value of the 16-bit counter. The count value is the principle output of this functional block whose value is a numerical representation of the duration of the magnitude of the system error.

VIII. COUT – *1-bit output* – This output is an active low representation of the overflow, or carry-out, of the 16-bit counter. In incremental counting, this output will have a value of 0 when the counter exceeds the count of 65535 (maximum binary count of 16 consecutive 1's).

In addition to the 16-bit binary counter, the derivative functional block relies on a register or other form of digital memory to implement the difference or differentiation functionality. The output resulting from the counter is modified to reflect the sign of the error signal. The previous value of the derivative counter that was stored in memory is subtracted from the current value of the counter. The result of the derivative control block with the signal operating modes described above is a numerical binary representation of the difference between the current and previous pulse widths of the tracking error of the system over two consecutive periods of the error signal.

iv. Generation of a Combined PID Error Signal

The core element of the accumulation (or combination) of all error components, K_A, is also a 16-bit binary synchronous counter. In the K_A functional block, the outputs of each of the functional blocks are added together using a digitally implemented binary adder. This value along with a number of control signals are inputs into the 16-bit counter and its associated modes of operation are defined in terms of the inputs. The implementation of the function relies on the following operating conditions on the inputs and outputs of the counter in Figure IV-4:

I. CLK – *1-bit input* – A digital clock pulse that varies between 1 and 0 at a frequency of f_A. This frequency determines the speed at which the synchronous counter increments or decrements. The count will increase or decrease on each rising edge of the CLK if enabled.

II. U_D – *1-bit input* – Active high signal that sets the direction of the count. A logic 1 (high) value increments the count and a logic 0 (low) decrements the count. The U_D input is tied to logic 0 to force the counter to always decrement.

III. EN_P – *1-bit input* – Active low input that enables the operation of the counter. Both EN_P and EN_T must be low in order for the count to increase. The EN_P input is tied to logic 0.

IV. EN_T – *1-bit input* – Active low input that enables the operation of the counter. Both EN_P and EN_T must be low in order for the count to increase. The EN_T input is tied to the logical "NOR-ing" of the bits of the output of the binary adder (the combined error signals from the constituent components). The result is that as long as the combined error count is non-zero, the counter is enabled and the count changes at frequency f_A.

V. _LOAD – *1-bit input* – Active low input that sets the value of the counter to the value available on input INDATA. The signal is connected to the inverse of an edge detector that is high when a negative edge occurs in the magnitude of the error. The result is that the integer sum of the error counts is loaded into the counter, every time the error transitions from non-zero to zero.

VI. INDATA – *16-bit input* – Data on this input can be loaded into the counter to set its value. This input is the magnitude of the output of the binary adder (the combined error signals from the constituent components).

VII. OUTDATA – *16-bit output* – This output represents the value of the 16-bit counter. The count value is used to generate the PWM control signal that is the expected output of the controller proposed.

VIII. COUT – *1-bit output* – This output is an active low representation of the overflow, or carry-out, of the 16-bit counter. In incremental counting, this output will have a value of 0 when the counter exceeds the count of 65535 (maximum binary count of 16 consecutive 1's).

The result of the accumulation control block with the signal operating modes described above is a numerical binary representation of the combination of the three constituent control signals counting towards zero at a frequency f_A.

v. Generation of the Error and Controller Output Signals

For the operations proposed above, the error signal is generated and decomposed into two components, the magnitude of the error and the sign of the error. Using a "binary" definition for the sign of a number, a value of logic 0 indicates a nonnegative number and logic 1 indicates a negative number. A truth table that relates the magnitude and sign of the system error in terms of the reference signal and system output is shown below in Table IV-1. The magnitude component of the error is formed by relating the desired system output to the actual system output with the exclusive-or logic function and is implemented with an XOR digital logic gate. The sign function is constructed from the relevant minterm in the truth table and is implemented with a NOT and an AND digital logic gate.

Table IV-1 – Truth Table of the error magnitude and sign functions.

Reference	System Output	Error Magnitude	Error Sign
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

The outputs of the proportional and derivative counting stages need to be scaled by a factor of 1 or -1 to reflect the appropriate sign for the error they represent. The binary error sign signal was defined with 0 representing all nonnegative numbers and 1 representing all negative numbers. The error sign signal can be mapped to create the appropriate scaling signal by inverting the signal, multiplying by 2 and subtracting 1. This mapping is shown in Table IV-2. It is important to note that the inversion can be realized in the form of a digital NOT gate, the subtraction can be performed by a simple

arithmetic logic unit (ALU) which is already required by the design for the accumulation phase. While multiplication is typically a more complex arithmetic operation, the task of multiplying by 2 can be implemented with a simple bit shift.

Table IV-2- Table depicting the operations required to map the error sign signal to the appropriate scaling factors.

	ErrorSign	Invert	Multiply by 2	Subtract 1
Positive	0	1	2	1
Negative	1	0	0	-1

In regards to the system output, a PWM control signal is generated by logically OR-ing the output of the cumulative error count from the K_A block. By forming the error in this way, the resulting signal will have a value equal to logic 0 when the result of the error signal is zero and will have a value equal to logic 1 for non-zero system error.

An exploded view of the all-digital proportional-integral derivative controller from Figure IV-3 is shown below in Figure IV-5. The block diagram details the interconnection of the proportional, integral and derivative components as well as the signal routings to operate the functional blocks described in the previous four subsections. The digital components for accumulation, memory storage, and the formation of the error signals are also represented. The *Simulink* implementation of the model is shown in Appendix C.

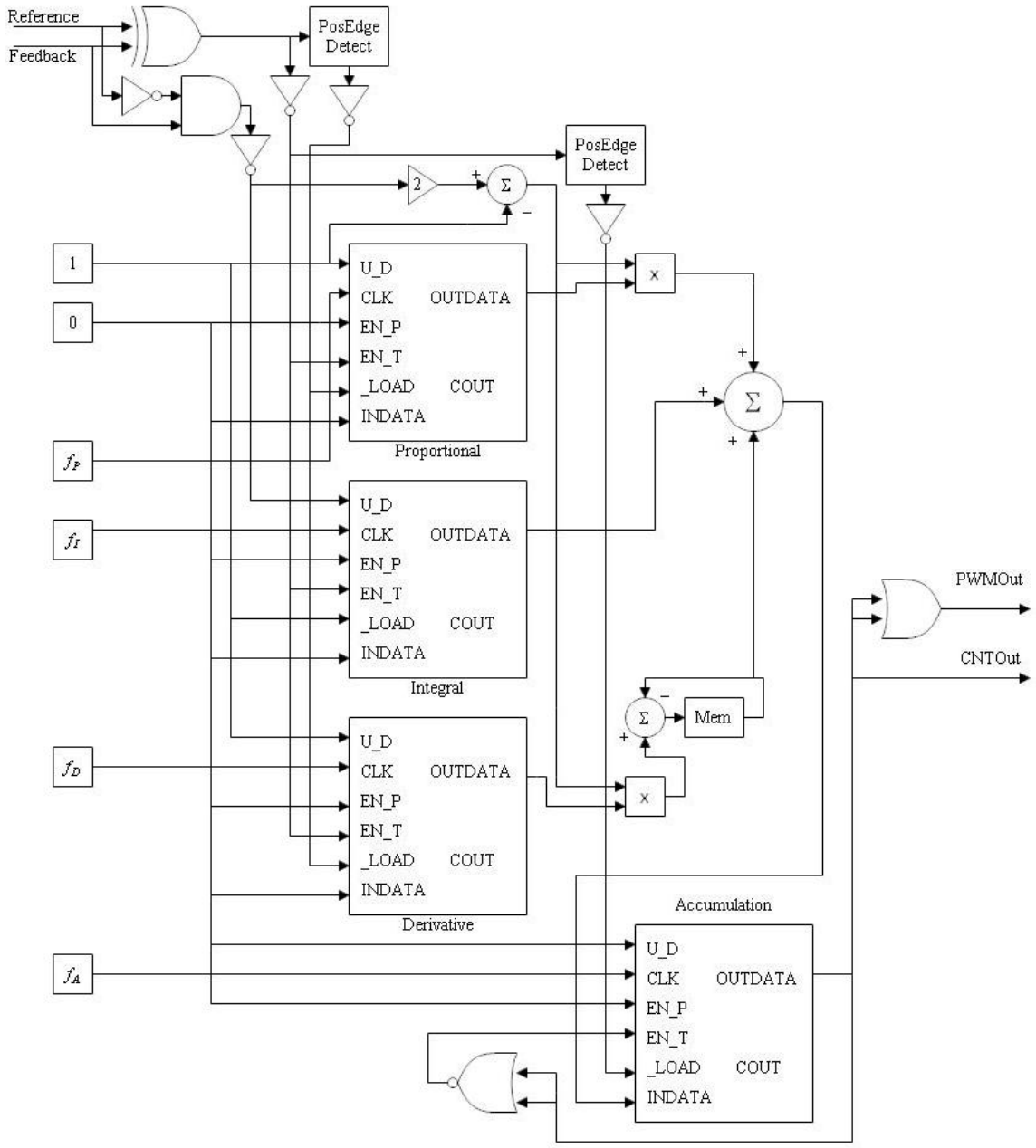


Figure IV-5 – Detailed block diagram view of the proposed all-digital PID controller and its constituent components.

C. PROCEDURE FOR DETERMINING ALL-DIGITAL PID CONTROL PARAMETERS

Within the scope of the all-digital PID instantiation described in this document, the relevant control or tuning parameters, (K_P, K_I and K_D), can be generically referred to as K_N, where $N = P, I$ *or* D. Each subcomponent of the control system described has an associated frequency, (f_P, f_I and f_D). Additionally, a counting frequency for the overall accumulation phase, f_A, is defined. Together, the set of four counting frequencies constitute the control or tuning parameters.

In designing a control law or loop with an all-digital PID controller implementation, one would determine the typical PID control parameters, (K_P, K_I and K_D), using one of the methods described in Section II B. Once the classical control parameters are determined, the process described below can be used to compute the required clock frequencies required to tune the all-digital PID controller.

First, a counting frequency for the accumulation phase needs to be selected. Each of the counting frequencies used as control parameters should be at least twice as large as the reference frequency that the system is to track. As a rule-of-thumb, the base frequency should be as large as practical, understanding the limitations of the hardware that will be driving the clock signals and that the computed gains will require the scaling of the base frequency to subsequent rates that may be higher by an order of magnitude or more. Guidance regarding the selection of the frequency, f_A, is expressed in (15). Once this parameter is fixed, the subsequent frequencies can be calculated from the relationship given in (16).

$$f_A \geq 2 \cdot \max\left(f_{Ref}, \max\left({}^{1}/_{K_P}, {}^{1}/_{K_I}, {}^{1}/_{K_D}\right) \cdot f_{Ref}\right) \tag{15}$$

$$\begin{aligned} K_N &= \frac{f_N}{f_A} \\ \Rightarrow f_N &= f_A K_N \end{aligned} \tag{16}$$

If the computation of (16) results in one or more control frequencies that push the limitations of the underlying hardware driving the signal, one should reduce the frequency of the accumulation stage, if possible, and recalculate the other frequencies. The thought behind selecting the minimum effective input frequency to be at least twice that of the desired reference is that the counting proposed is effectively sampling. As a result, the well-known Nyquist Sampling Theorem would suggest that the underlying information of the relevant controller stage would need to be counted or sampled at this rate in order to be accurately represented.

Copyright © David Michael Feinauer 2011

V. EXPERIMENTAL RESEARCH

Given the all-digital PID design proposed above, the performance of the controller is contrasted with that of a classical PID control law in this section. The controller performance is investigated in the application of controlling a basic DC motor for the purpose of tracking a velocity. The DC motor plant is modeled as a transfer function between rotor position and input voltage. The transfer function for a DC motor (17) and the model parameters for a basic one-quarter horsepower motor were obtained from [10] and are shown in Table V-1. The asterisk designation of parameters in the table indicates additional values that were selected based on the supplied field time constant parameter, as the effects of the field time constant, although negligible, were included in the transfer function model.

$$\frac{\theta(s)}{V(s)} = \frac{K_m \Big/ {J_m}/{\tau_L} \cdot {L_f}/{\tau_f}}{s(\tau_f s + 1)(\tau_L s + 1)} \tag{17}$$

Table V-1 – Parameters for DC motor mode from [10].

Symbol	**Value**	**Description**
K_m	$50 \times 10^{-3}\ \mathrm{N{\cdot}m}/\mathrm{A}$	Motor Constant
J_m	$1 \times 10^{-3}\ \mathrm{N{\cdot}m{\cdot}s^2}/\mathrm{rad}$	Rotor Inertia
τ_f	1 ms	Field Time Constant
τ_L	100 ms	Rotor Time Constant
L_f	30 mH	Field Inductance*
R_f	0.3 Ω	Field Resistance*
MaxPower	$1/4$ hp	Maximum Output Power

For the proposed framework, an optical encoder sensor that measures the motor / plant output is required to close the loop and provide a feedback signal. A model for an incremental quadrature optical encoder that produces 360 pulses per revolution is given in Appendix E.

An initial functional verification of the proposed model was conducted. Analysis of the initial design verification simulations revealed some fundamental issues in the design as proposed in Chapter IV. Modifications necessary for the design were developed and are subsequently presented in subsection A of this chapter. The computation of the control parameters and an analysis of the simulation results of the DC motor for both the classical and the modified all-digital controller systems are detailed in subsection B. Subsection C discusses methods for investigating the stability of the new-formed model.

A. Revisions to the All-Digital PID Compensator

The all-digital PID compensator detailed in chapter IV was interfaced to the model of the motor plant described in Table V-1 for initial analysis. The system was implemented for the purpose of tracking a velocity of 60 rpm. Initially, the model appeared to track the desired velocity set point. As the model was reconfigured to explore the control performance and the underlying system dynamics, the initial model revealed its limitations. For a fixed set of control parameters, as the desired tracking velocity was varied, the resulting system output settled to a steady state velocity that was highly sensitive to the commanded input. In some cases, small increases to the commanded input resulted in a decrease in the steady state velocity attained. An example of this is shown in Figure V-1. The red dashed line present in this figure represents the commanded velocity reference for the system simulation; throughout the remainder of this chapter, the presence of a red or dashed line in any figure will serve to indicate the relevant input velocity reference.

For the first setpoint reflected in the plot (top), the model converged to a steady state velocity with steady-state ripple about the desired output of 60 rpm. In the second plot, the system attained a steady state velocity of approximately 48 rpm, significantly less than the desired output of 120 rpm and the previous output of 60 rpm.

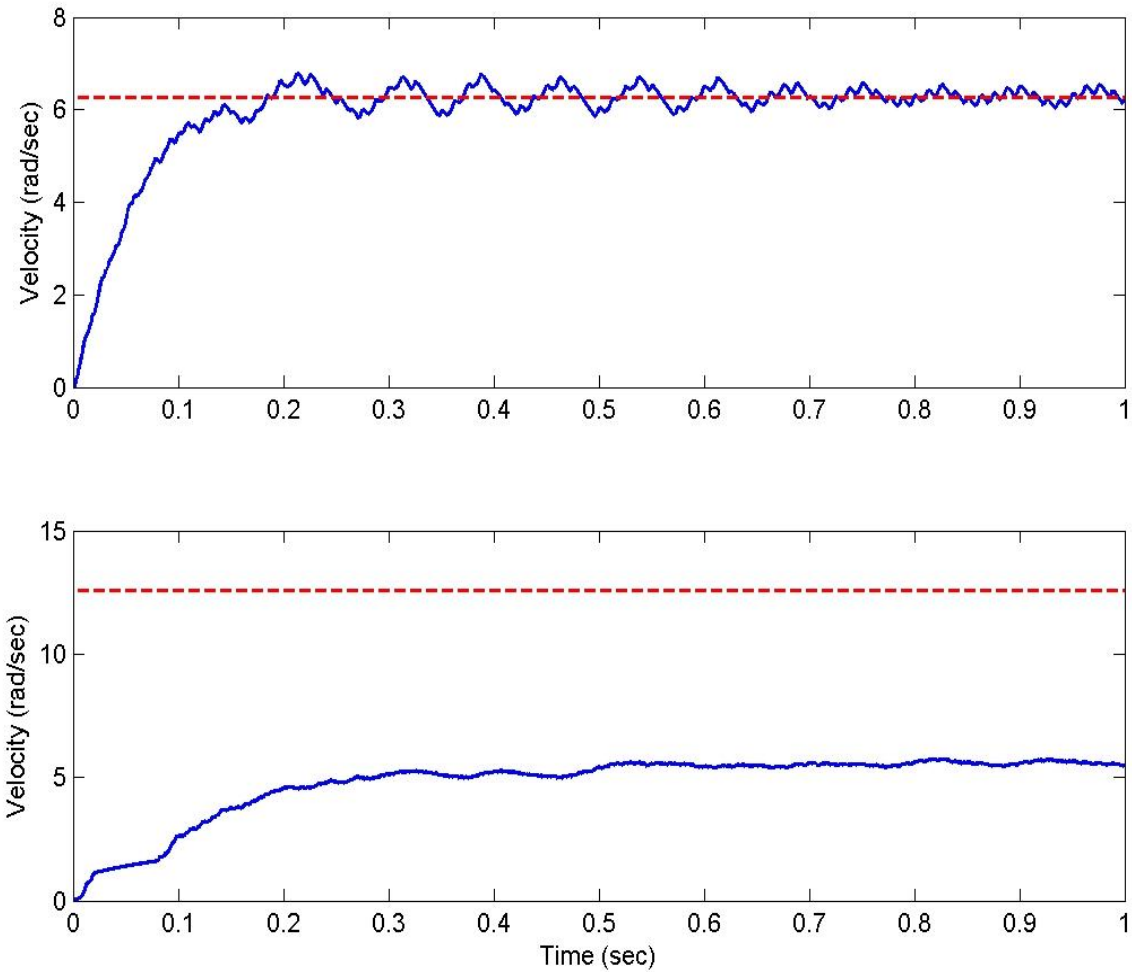


Figure V-1 – Initial all-digital PID results for a commanded tracking velocity of 60 rpm (top) and 120 rpm (bottom).

After tracing the signals internal to the controller and examining the affect of the various input and clock signal transitions on the counting circuitry, it became apparent that to accurately represent the error signal and actuate the counter circuitry appropriately, a more sophisticated implementation that allowed for some form of signal history or "previous state" information was needed.

The all-digital PID compensator was modified to include a new method of error formation. A next-state truth table mapping the appropriate direction and count occurrences was developed and implemented. The details of the implementation and the related next state table for the sequential logic device are shown in Appendix D. The device actuates the direction (U_D) and enable (EN_T) signals for the counters as described in chapter IV. In addition, the device actuates the directional signals for the counters in the proportional and derivative stages, eliminating the need for modifying the output from those stages to account for the sign of the error they represent. The control signals for the counters forming the all-digital PID compensator are a function of: the current and previous reference and feedback inputs as well as the previous count direction. The count direction and enable signals are representative of the sign and magnitude of the tracking error of the system within a small window of time. The device implementation could be realized with a combination of combinatorial logic gates and latches.

In addition to the error formation circuitry, a more detailed modeling of the memory storage and associated latching required for the system was developed. Two additional counters were used as memory storage registers to latch the computed difference output from the derivative stage and the accumulation of the three signals to address signal timing issues in the initial design. An updated *Simulink* model based on these additions is shown in Appendix D, immediately following the description of the modified error computation circuitry. This modified all-digital PID compensator was used for simulation and analysis in comparison to a classical PID compensator in the following subsection and throughout the remainder of this document.

B. *ALL-DIGITAL PID VERSUS TRADITIONAL PID*

i. ADPID comparison to a Baseline System

Consider the aforementioned model for a DC motor as a plant to be controlled. The objective is for a controller to be designed such that the motor tracks a velocity with the transient specifications required as follows:

1. Settling time, $t_s = 0.5$ seconds
2. Damping coefficient, $\zeta = 0.49$
3. Steady-state error, $\text{ess}|_{\text{parabola}} = 0.1$

Substituting the motor parameter values from Table V-1 into (17) yields the motor transfer function (18).

$$\frac{\theta(s)}{V(s)} = \frac{50/3}{s(.001s+1)(.1s+1)} \tag{18}$$

Given a DC motor system defined by (18), the open loop stability of the system can be determined from the poles of the transfer function. The DC motor system modeled by the transfer function is considered marginally stable because it has an open loop pole on the imaginary axis along with its two open loop poles in the left-half plane. A root locus plot of the system is shown depicting the open-loop poles, $s = \{0, -10, -1000\}$, and the trajectory of the resulting poles from a closed-loop system with a negative feedback gain, *K*, varying from 0 to ∞. The system becomes unstable with poles in the right-half plane for gains of *K* larger than approximately 53.

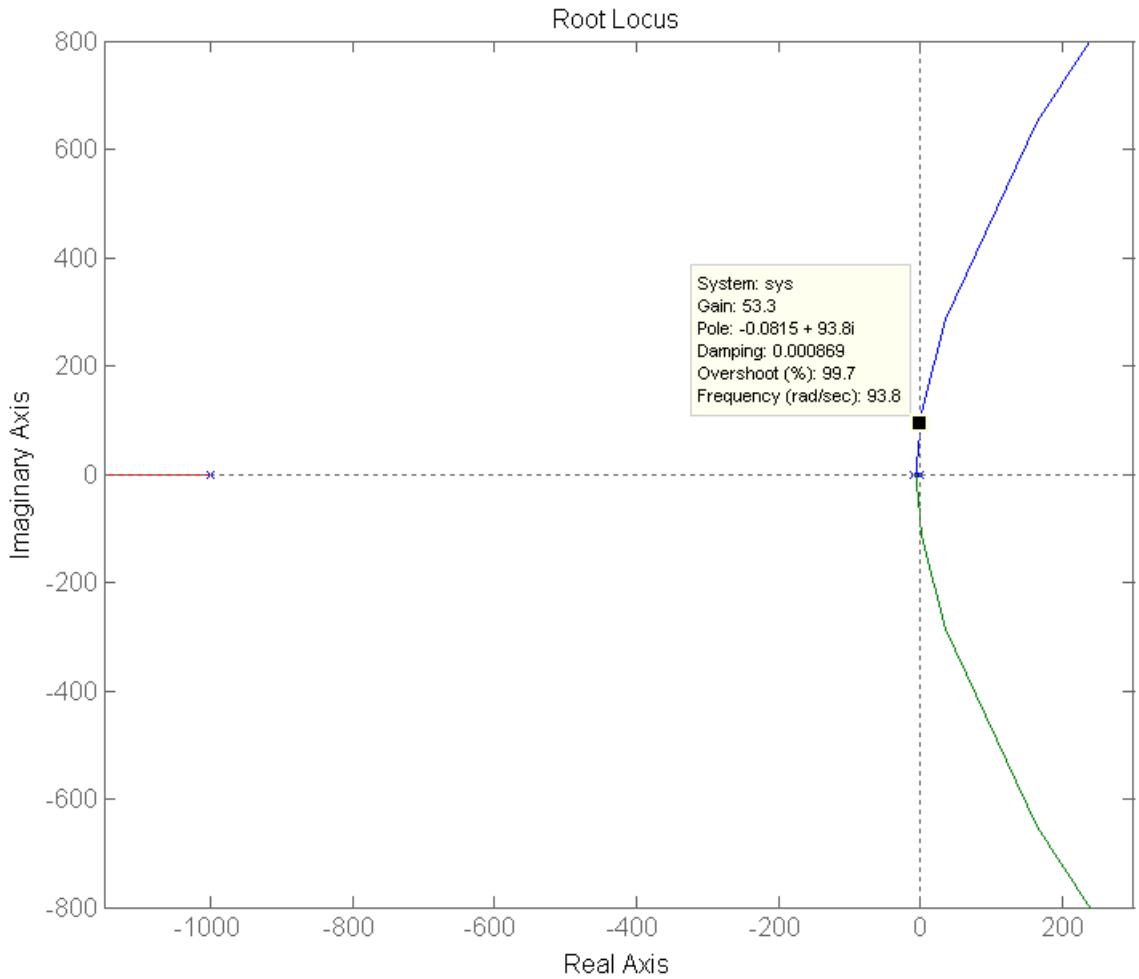


Figure V-2 – Root locus plot of the transfer function of a DC motor plant.

Given a system defined by the transfer function (18), a PID compensator can be designed using the criteria set out in section II.B.ii.

I. The desired settling time and damping coefficient for the compensated system can be used to solve for the desired dominant poles.

$$s = \zeta\omega_n \pm \omega_n\sqrt{\zeta^2 - 1} \tag{19}$$

$$t_s = -\frac{4}{\zeta\omega_n} = -\frac{4}{Re(s_{\max})} \tag{20}$$

$$\Rightarrow Re(s_{\max}) = -\frac{4}{0.5} = -8$$

$$\Rightarrow Im(s_{\max}) = \pm\left(\frac{8}{0.49}\right)\sqrt{(0.49)^2 - 1} = \pm j14.23$$

$$\Rightarrow s_{max} = -8 \pm j14.23$$

II. The compensated system is of type 2 due to the additional pole at s=0 introduced by the integrator term of the PID compensator. The desired steady state error condition can then be used to solve for the integrator gain, K_I.

$$ess|_{parab} = \lim_{s\to 0} s\frac{\frac{1}{s^3}}{\left(1 + G(s)G_{PID}(s)\right)} = \lim_{s\to 0}\frac{1}{s^2\left(1 + G(s)G_{PID}(s)\right)} \tag{21}$$

$$\Rightarrow ess|_{parab} = \lim_{s\to 0}\frac{1}{s^2\left(1 + \frac{50/3}{s(.001s+1)(.1s+1)}\left(K_P + K_D s + \frac{K_I}{s}\right)\right)}$$

$$= \frac{1}{50K_I/3} = 0.1$$

$$\Rightarrow K_I = 0.6$$

III. Using the value of K_I solved for in part II and selecting $s_1 = -8 + j14.23$ as one of the dominant poles from part I, the combined system transfer function is evaluated.

$$K_P + K_D s_1 = -\frac{1}{G(s_1)} - \frac{K_I}{s_1} \tag{22}$$

$$\Rightarrow (K_P - 8K_D) + j(14.23K_D) = -\frac{1}{-0.6632 + j0.2702} - \frac{0.6}{-8 + j14.23}$$

$$= 1.2932 + j0.5268 + 0.018 + j0.032 = 1.3112 + j0.5589$$

$$\Rightarrow K_D = \frac{0.5589}{14.23} = 0.0393 \approx 0.04$$

$$\Rightarrow K_P = 1.3112 + 8K_D = 1.6254 \approx 1.6$$

$$\Longrightarrow G_{PID}(s) = 1.6 + 0.04s + \frac{0.6}{s}$$

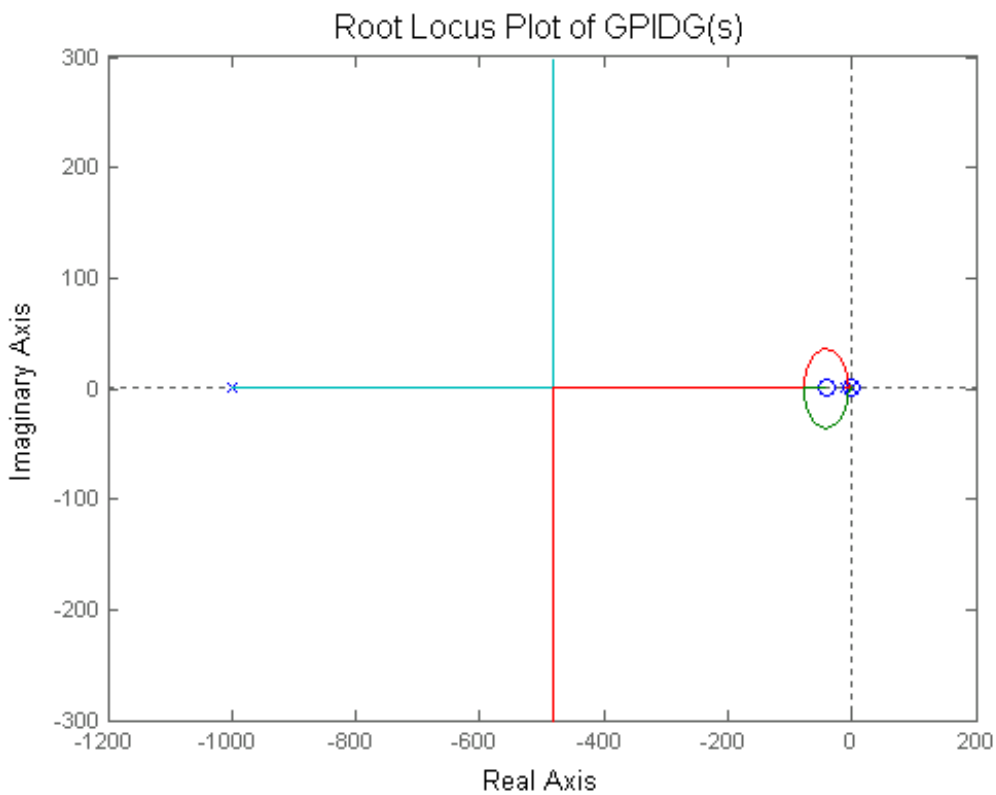


Figure V-3 – Root locus plot of the closed loop PID compensated DC motor system.

A root locus plot of the PID compensated closed loop DC motor system is shown in Figure V-3. With the classical design of the PID control parameters, the all-digital counting frequencies can be determined. For the simulation, the motor will track a velocity of 60rpm which equates to a 360 Hz square wave output from an optical encoder that produces 360 pulses per revolution. As a guide from (15), the base counting frequency, f_A, should be designed to be larger than $2 \cdot {}^{1}/_{0.04} \cdot 360\text{Hz}=18000\text{Hz}$. Selecting $f_A = 20000\text{Hz}$ satisfies this criteria. The subsequent frequency parameters can be calculated as follows:

$$f_P = f_A K_P = 32000Hz$$
$$f_I = f_A K_I = 12000Hz$$
$$f_D = f_A K_D = 800Hz$$

With the computed control parameters, the all-digital PID controller model can be implemented and simulated.

As a baseline for comparison and analysis of the novel, all-digital PID compensator, a classical PID feedback control system for the DC motor plant was implemented in *Simulink* and is shown in Figure V-4. The system implementation in Figure V-4 represents an ideal case where “true” position output is available from the system as a feedback input for the PID controller and the interface between the controller and the motor plant is both high resolution and analog.

A more realistic system modeling the effects of the plant-controller interface is depicted in Figure V-5; the interface takes into account the common implementation of the DC motor plant with an incremental encoder and associated decoder used for sensing and tachometry on the motor output. The controller-plant interface is also modified to reflect the more practical implementation of a voltage to PWM mechanism for actuating the motor or other plant under control.

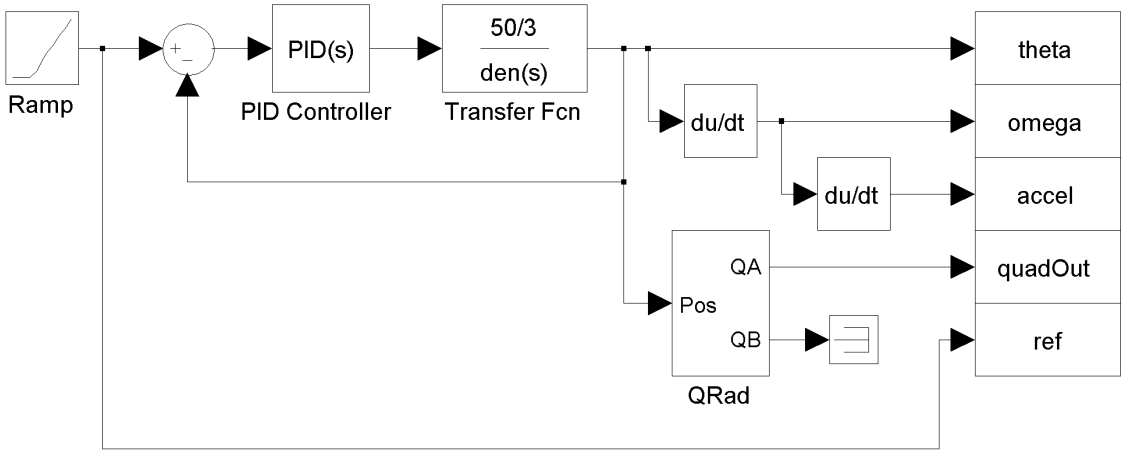


Figure V-4 – Velocity tracking classical PID compensated DC motor.

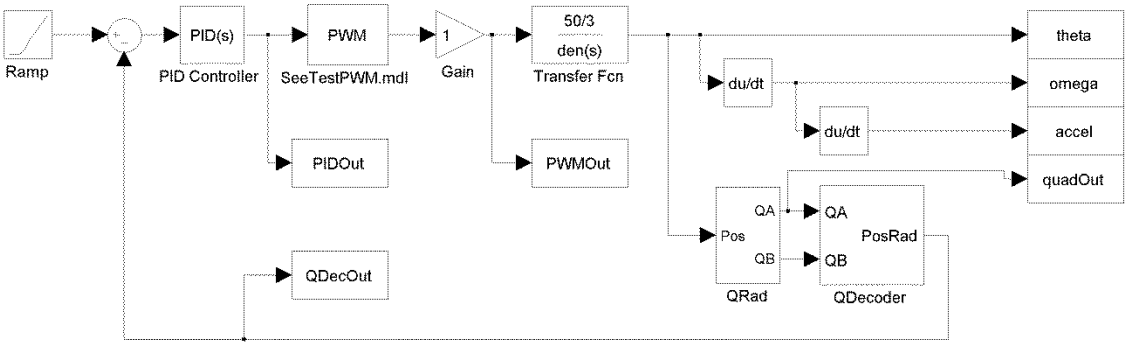


Figure V-5 – PID compensated DC motor plant model with typical controller to motor and motor to controller interfacing.

The simulated position as well as the estimated velocity and acceleration outputs of the motor under control are shown for the PID compensated theoretical system in Figure V-6. The first graph in Figure V-6 reflects the simulated position output, θ, of the compensated DC motor plant. The second and third plots in the figure depict estimates for the system velocity, $\omega = \dot{\theta}$, and acceleration of the system under control.

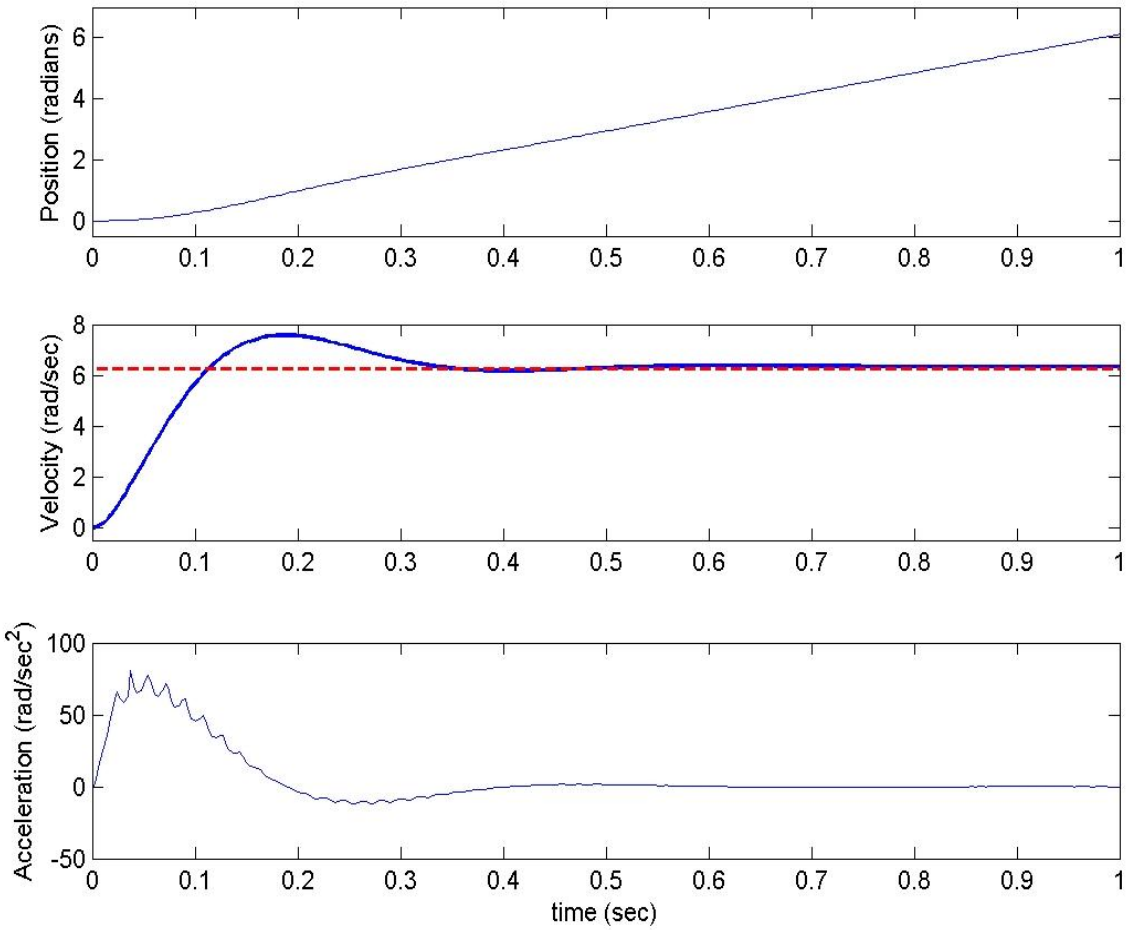


Figure V-6 – Position, velocity and acceleration outputs from the PID compensated DC motor system.

A motor tracking a constant velocity of 60 rpm, one revolution per second, should complete one revolution of $2\pi \approx 6.28$ radians in one second. One would expect the position of the motor in radians to be represented by a line with a slope of 2π. As shown by the simulated system output, the compensated DC motor plant completes one rotation in the first second of simulation (position reaches a value of approximately 2π) and the system settles to the steady state tracking velocity. By comparison, Figure V-7 shows the position, velocity and acceleration outputs for the classical PID controller interfaced to the model with typical interface technology.

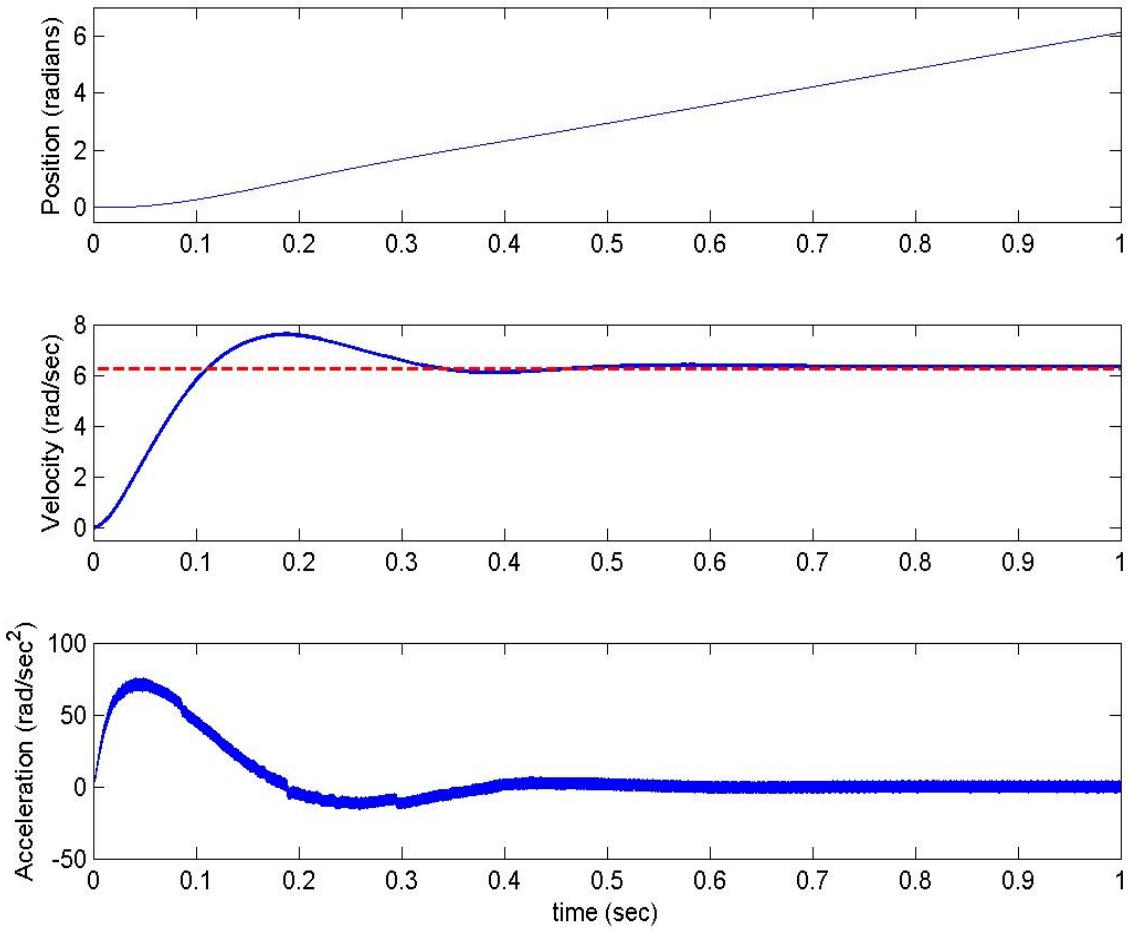


Figure V-7 - Position, velocity and acceleration outputs from the PID compensated DC motor system with typical controller-plant interfaces.

Table V-2 – Performance characteristics of the PID compensated plants in Figure V-4 and Figure V-5.

Description	**Symbol**	**System in Figure V-4**	**System in Figure V-5**
Peak Overshoot (%)	M_p	20.6	21.24
Time to Peak (sec)	t_p	0.186	0.189
Settling Time (sec)	t_s	0.326	0.423
Steady-State Ripple (%)			+2.0 to -1.88

In comparing the velocity outputs of the two systems, it is apparent that the addition of the PWM and encoder controller-plant-controller interfaces resulted in an increased settling time and increased overshoot of the output velocity. The PWM effects also produced a small amount of output velocity ripple about the desired steady state operating point. It is interesting to note that in a classical PID control framework, increasing the gain of the proportional and or integral tuning parameters would also result in increased overshoot, increased settling time and decreased output stability.

The incremental encoder measuring the system output and the corresponding decoding of the digital signal also contributed to the differences in the performance characteristics of the systems. In the “more realistic” system presented, the analog output of the quadrature decoder is subtracted from the reference and fed directly into the PID controller. In actuality, this analog signal would undergo analog to digital conversion (ADC) before interfacing with the control system, which would induce further imprecision and potential output ripple in the baseline system due to quantization and other related errors.

A summary of the measured transient performance characteristics of the simulated systems is shown in Table V-2. Defining settling time as the time after which the system remains within ±2% of the steady state operating point would yield a marginally defined value for the PWM system due to the ripple effects on the output. The settling time represented in the table for the PWM system is the time at which the system settled to within -2% of the desired steady state output after the time at which the output reached its peak.

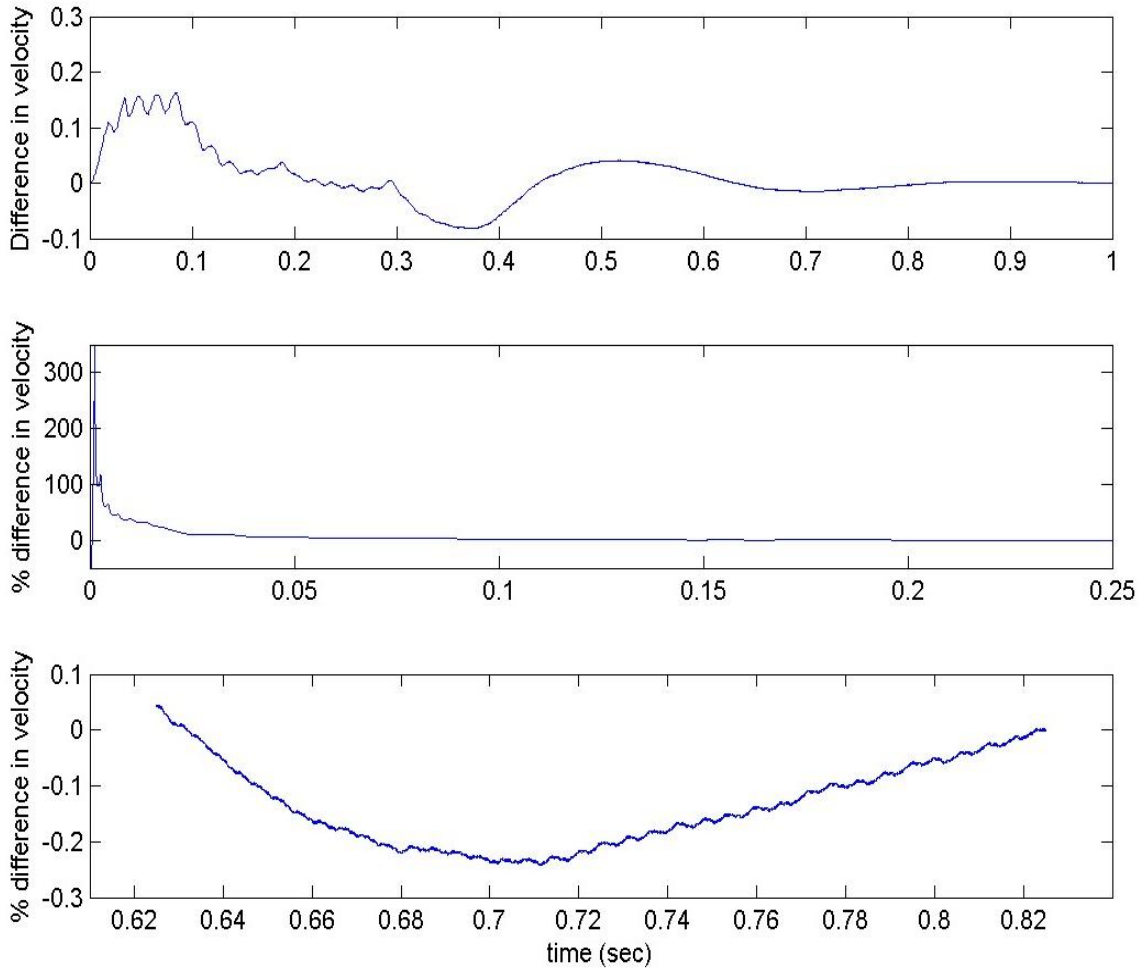


Figure V-8 – Absolute and percentage difference of the tracking velocities attained by the theoretical and modified systems.

For closer examination, Figure V-8 shows both the absolute difference and the percent difference between the velocity of the theoretical and modified systems under control. At steady state, the system oscillates within approximately ±0.5% of the velocity attained by the ideal theoretical system. Samples of the controller-plant-controller interface signals as the DC motor is starting up are depicted in Figure V-9. The PWM control output from the interface is depicted in the top graph and the output of the optical encoder sensor is illustrated in the bottom plot of the figure.

The aforementioned simulation results will serve as a baseline for performance analysis of the proposed all-digital PID compensator. The *Simulink* implementation of the proposed all-digital PID controlled system is shown in Figure V-10. The simulated position output along with velocity and acceleration estimates for the motor under control are subsequently depicted in Figure V-11.

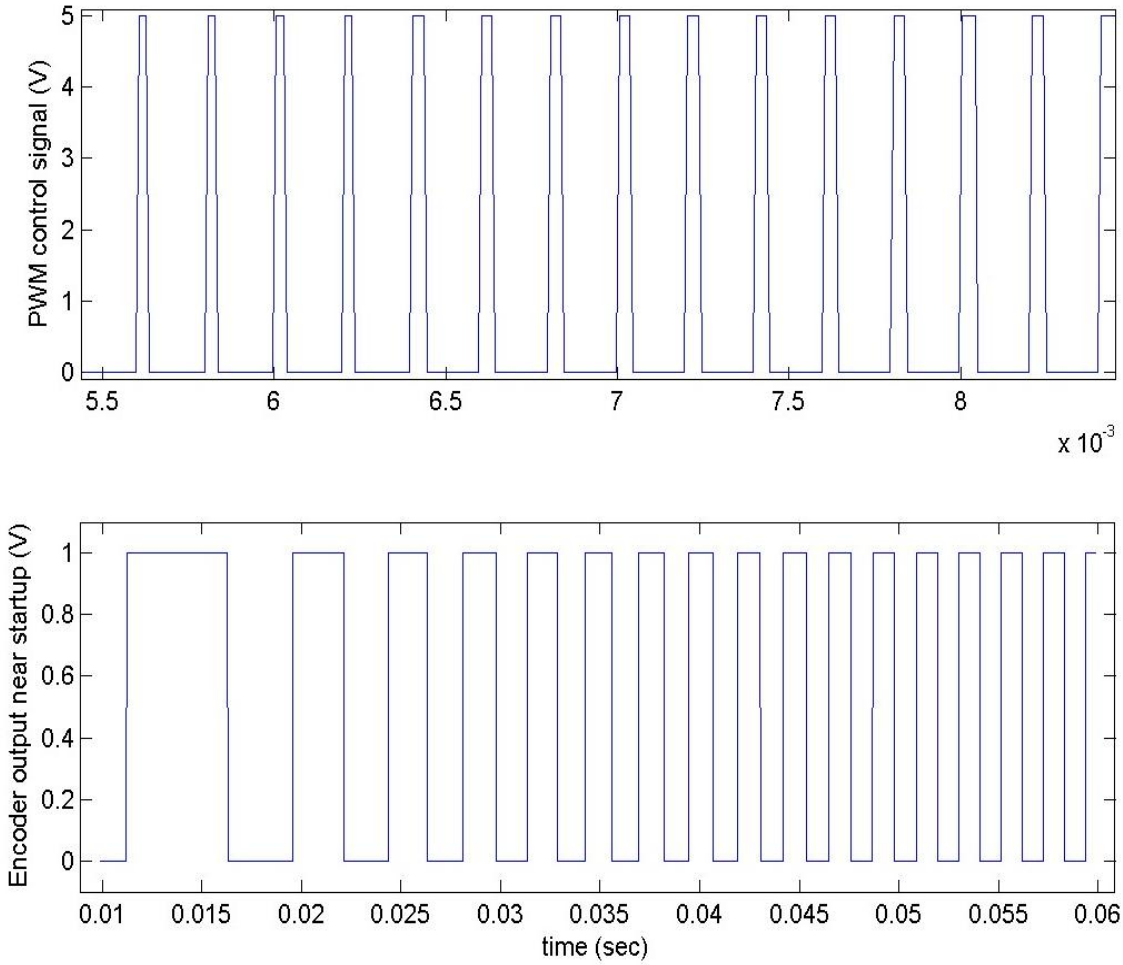


Figure V-9 – Signals representative of the PWM interface to the classical PID compensator and one channel of the quadrature encoder output of the system plant.

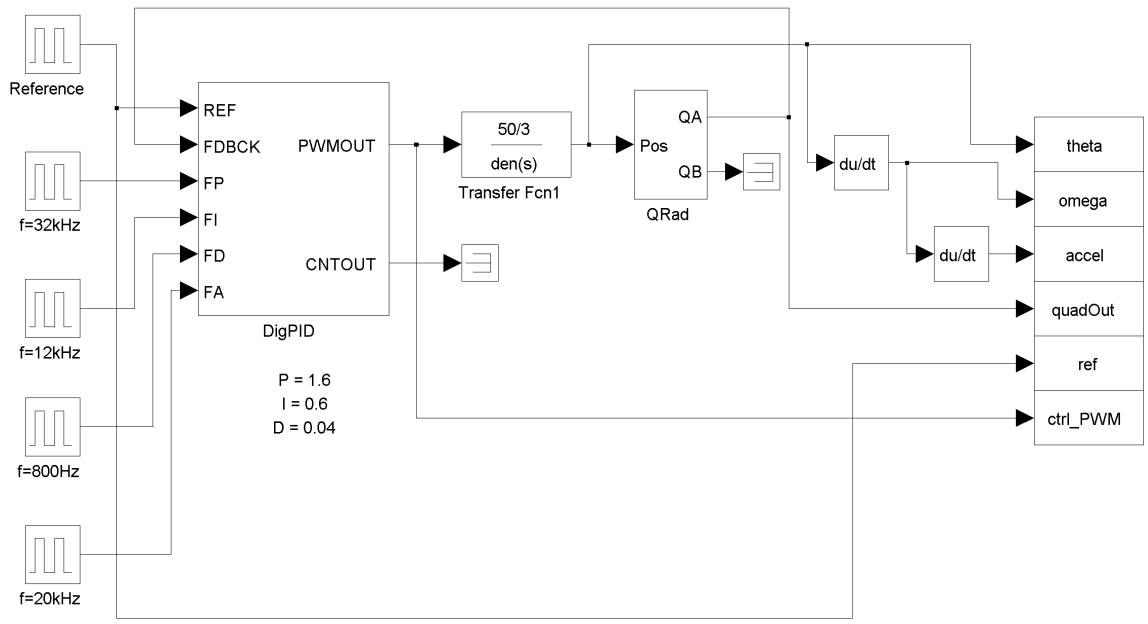


Figure V-10 – Velocity tracking all-digital PID compensated DC motor.

The initial problem statement requires that the system under control track a reference velocity. In the traditional PID system with controller-plant-controller interfaces modeled, Figure V-5, the feedback signal is the motor position output measured by an optical quadrature encoder / decoder pair. The result is an analog value representative of system position. The reference input required for the system is a ramp function with a slope of 2π, corresponding to a constant velocity of 1 revolution per second. In the all-digital system, Figure V-10, the feedback signal is a digital representation of the system position measured by a quadrature encoder. The square wave signal of variable frequency that results from the encoder is not decoded; it is fed directly into the all-digital compensator as an estimate of the system velocity. A reference square wave of constant velocity serves as the input to the compensated all-digital system. For an encoder with a resolution of 360 pulses per revolution, the equivalent reference input for a system tracking 60 rpm is a 360 Hz square wave.

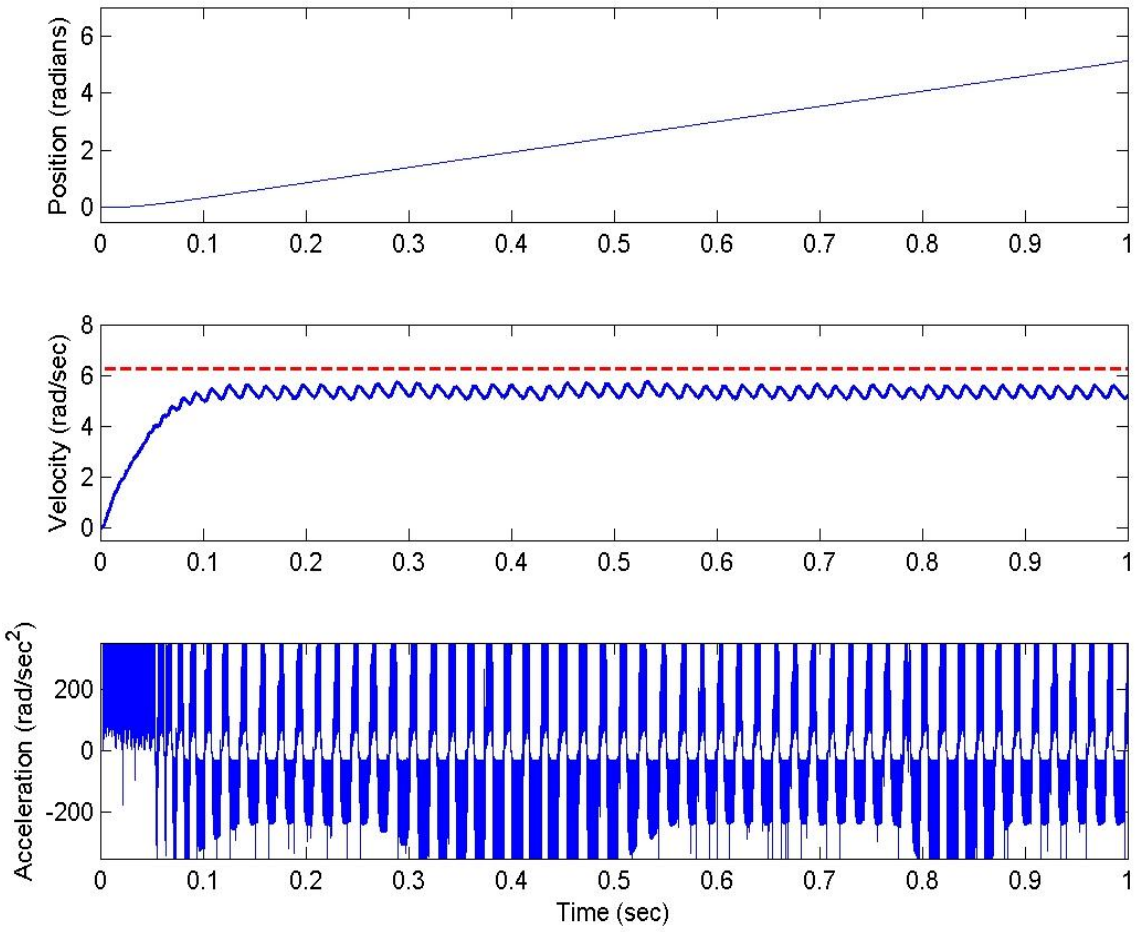


Figure V-11 – Position, velocity and acceleration outputs from the all-digital PID compensated DC motor system.

From the results shown in Figure V-7 and Figure V-11, it can be observed that at a time of one second, the classical system attained a position of approximately 6.28 radians. The all-digital system lagged that metric and only attained a position of approximately 5.2 radians, tracking an average steady state velocity of 5.33 rad/sec—approximately 15.2% less than the desired setpoint. The settling time for the classic and all-digital systems are approximately 0.42 seconds and 0.16 seconds respectively. The all-digital PID system did not present any overshoot. Due to the PWM nature of both systems, it is important to note that although the system converges to a “steady-state” velocity,

oscillation or ripple about the setpoint occurs. For the baseline system, the ripple was measured to be in the range +2.0 to -1.88% of the desired setpoint. For the all-digital PID compensator, the oscillation was found to be in the range of + 7.7% to -5.6% of the steady state velocity attained.

ii. Setpoint and Parameter Sensitivity in Velocity Tracking

For further analysis of the ADPID system versus the baseline, the systems were re-simulated to track velocities of 120 rpm, 240 rpm and 30 rpm. Thus far, the motor system has been actuated by a 1V PWM signal. With a maximum control output of 1V, the velocity saturated before the desired output of 240rpm could be reached. Figure V-12 depicts the simulation results with the rows of the plot displaying from top to bottom the plant outputs for the velocity tracking references of 60, 120 and 30 rpm. The left column plots the output of the baseline system and the all-digital system is depicted in the right column. Although the steady state velocity attained by the all-digital PID compensated system increases or decreases appropriately in correspondence to a change in the tracking setpoint, the error between the steady state value attained and the commanded setpoint reflects a potential sensitivity of the system to the control parameters and motivates the need for refining the method of selecting the parameters. It is also important to note that in the cases of tracking 60 and 120 rpm, the steady state error was negative with respect to the commanded velocity, and in the case of tracking 30 rpm, the error in the steady-state velocity attained was positive.

The sensitivity of the steady-state velocity attained by the closed-loop digital PID compensated system was investigated by implementing a proportional only controller with an accumulation counting frequency, f_A = 20 kHz, and varying the proportional gain K_P. For values of $K_P = \{0.8, 1.6, 4, 8, 16, 28, 32, 50\}$, the closed loop system remained stable and the system attained a steady state velocity that oscillated between approximately 6.1 and 5.15 rad/sec with an average value of approximately 5.6 rad/sec in all cases.

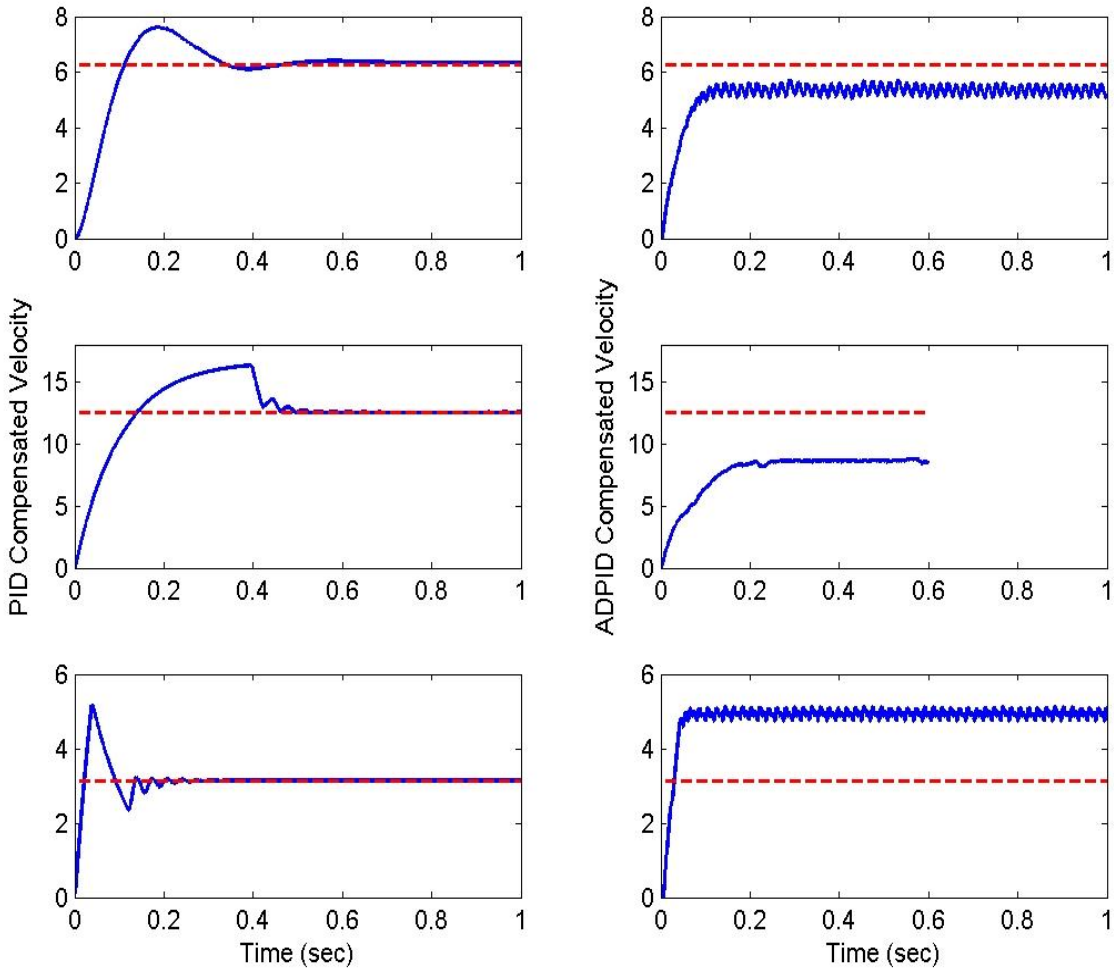


Figure V-12 – Velocity tracking comparison of the baseline and all-digital compensators.

The sensitivity of the system with respect to the accumulator tuning frequency was then investigated by reducing the parameter to 5 kHz and measuring the system output for a variety of proportional gains. For gains larger than 10, the closed loop system remained stable and the system attained a steady state velocity that oscillated between approximately 8.1 and 7.7 rad/sec with an average value of approximately 7.85 rad/sec. For a gain less than or equal to 10, the system oscillated in the neighborhood of the desired 6.28 rad/sec setpoint attaining an average steady state velocity of approximately 5.7 rad/sec.

In the first investigation, the response was not particularly sensitive to the gain *K*. In the latter case, although the system stabilized for large gains, the value of the gain *K* at which the response stabilized near the desired setpoint was reduced in the presence of a decreased base counting frequency. Reducing the base frequency by a factor of 4 increased the steady state velocity attained by roughly 25% for gains larger than 10, effectively limiting the range of valid proportional gains. For gains less than 10, the system tracked a velocity near the setpoint. It is also important to note that due to the binary output of the system, gains in excess of the approximate root locus stability bound of 53 did not cause the system to become unstable.

The sensitivity of the system about the integrator was investigated by implementing an integral-only control law and varying the integral tuning parameter for a base counting frequency of 5 kHz. The output of the closed-loop system from Figure V-10 for an integrator gain of 0.04 (left) and 0.06 (right) is depicted in Figure V-13. The change in the output velocity and the actuating control law based on a small change in the integrator gain as well as the small size of the integrator gain in terms of absolute magnitude indicates that the model is likely sensitive to the integrator gain. With the integrator term carrying significant weight, the selection of an integrator gain should likely be scaled down from what was originally suggested under the guidance of the rule-of-thumb in section IV-C. Additionally, the range limit inherent in a PWM–type output is commonly considered to contribute to “windup” or saturation of the integrator term in the control law if the effective controller output is operating near the range limit. A technique for addressing this effect should be considered for future development of this design methodology. One such possibility would be setting a range limit on the counter for the integrator term, capping the contribution it can make to the accumulated control law signal at the expense of attaining a minimal steady-state error.

Further investigation and increased simulation capacity is needed to characterize the sensitivity of the system to its selected gains and frequencies for the purpose of refining the proposed method of parameter tuning. Refinements to the method of parameter selection or tuning are essential to improving the steady-state error between the commanded and actual velocity attained by the system. In an attempt to improve the simulation times, particularly for system with large gains, the model was modified to include an edge detector that did not require an explicit clock signal for input signal latching in the hopes of reducing the number of signals and variables in the simulation. This model was slightly effective at reducing the simulation time in the case of fast counting frequencies resultant from large gains.

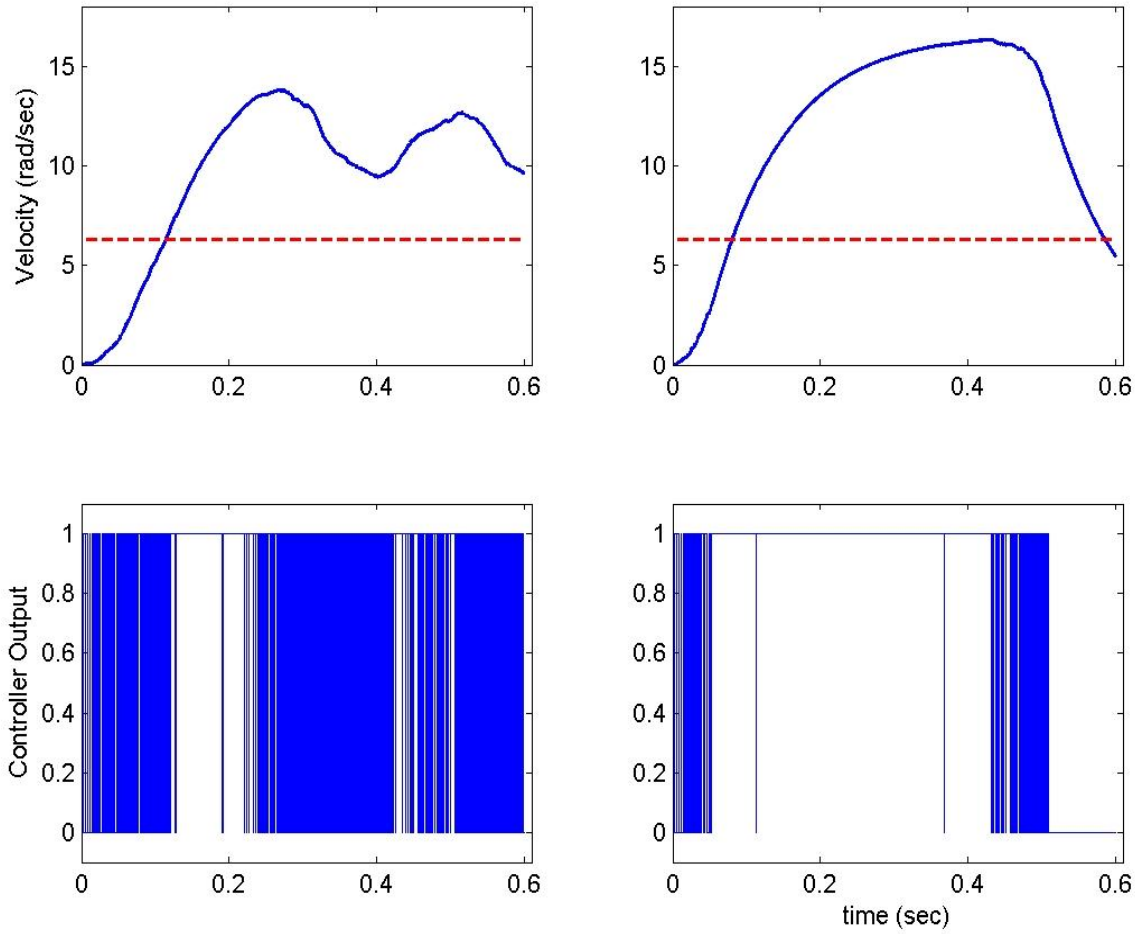


Figure V-13 – Velocity and control outputs of a closed-loop DC motor plant with an integral gain of 0.04 (left) and 0.06 (right).

iii. Baseline Comparison for an Additional DC Motor Plant

To continue validation of the all-digital PID compensator, the controller performance was again investigated for the purposes of a velocity tracking DC motor-based plant. An armature-controlled DC motor driving a rotational load was found in [9] to have a transfer function defining the relationship between rotor position and input voltage given by (23).

$$\frac{\theta_L(s)}{V(s)} = \frac{\left({}^{N_1}/_{N_2}\right) {}^{K_t}/_{R_a J_m}}{s\left[s + {}^{1}/_{J_m}\left(D_m + {}^{K_t K_b}/_{R_a}\right.\right]} \tag{23}$$

Model parameters for the motor from [9] and the resulting transfer function are shown in Table V-3 and (24) respectively.

$$\frac{\theta(s)}{V(s)} = \frac{{}^{5}/_{12}}{s(s + {}^{5}/_{3})} \tag{24}$$

Table V-3 – Armature-controlled DC motor with load specifications.

Symbol	**Value**	**Description**
${}^{K_t}/_{R_a}$	$5\ {}^{\mathrm{N{\cdot}m}}/_{\mathrm{V}}$	Electrical Constant
J_m	$12\ \mathrm{kg{\cdot}m^2}$	Total Inertia at the Armature
D_m	$10\ {}^{\mathrm{N{\cdot}m{\cdot}s}}/_{\mathrm{rad}}$	Total Damping at the Armature
K_b	$2\ {}^{\mathrm{V}}/_{\mathrm{rad/sec}}$	Electrical Constant

Consider the aforementioned model for a DC motor as a plant to be controlled. The objective is for a controller to be designed such that the motor tracks a velocity with the transient specifications required as follows:

1. Settling time, $t_s = 1.0$ second
2. Damping coefficient, $\zeta = 0.49$
3. Steady-state error, $ess|_{parabola} = 0.25$

Given a DC motor system defined by (24), the open loop stability of the system can be determined from the poles of the transfer function. The DC motor system modeled by the transfer function is considered marginally stable because it has an open loop pole on the imaginary axis along with one open loop pole in the left-half plane. A root locus plot of the system is shown in Figure V-14 depicting the open-loop poles ($s = [0, -1.67]$), and the trajectory of the resulting poles from a closed-loop system with a negative feedback gain, *K*, varying from 0 to ∞. The root locus reflects that the system is stable under negative feedback for positive values of K.

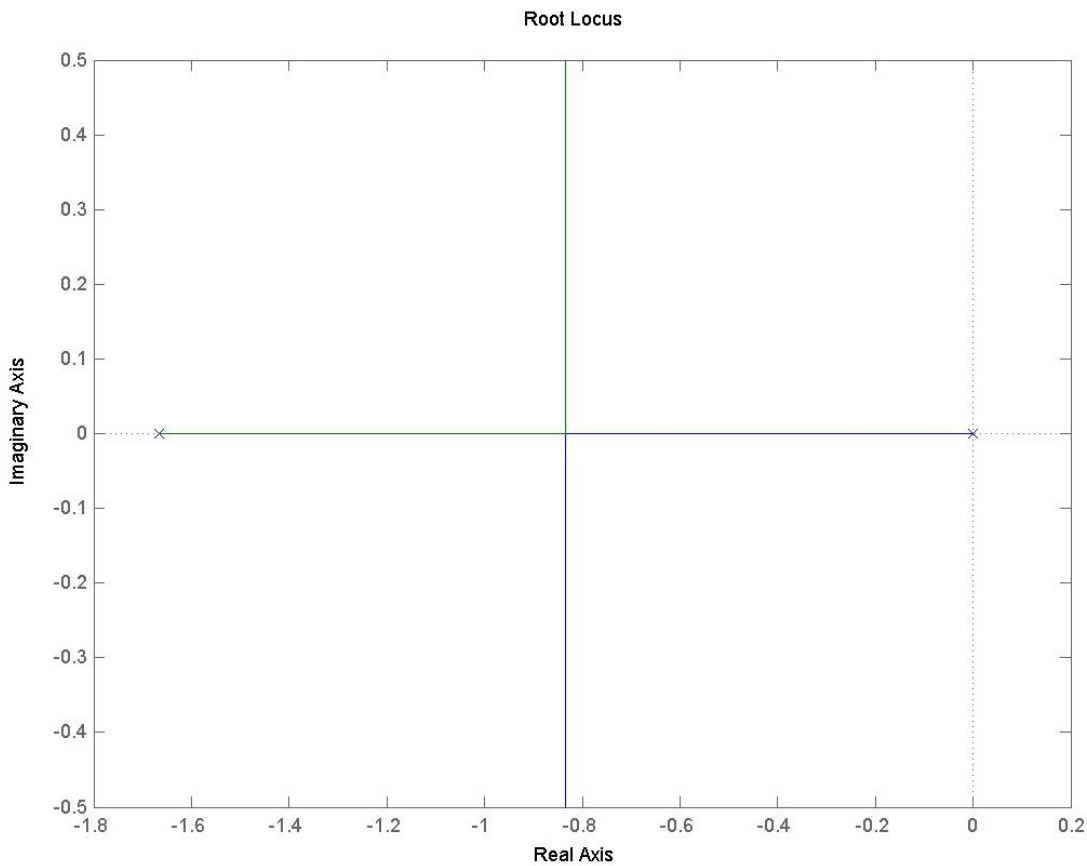


Figure V-14 – Root locus plot of the transfer function for a DC motor plant.

Given a system defined by the transfer function (24), a PID compensator can be designed using the criteria set out in section II.B.ii.

I. The desired settling time and damping coefficient for the compensated system can be used to solve for the desired dominant poles.

$$s = \zeta\omega_n \pm \omega_n\sqrt{\zeta^2 - 1} \tag{25}$$

$$t_s = -\frac{4}{\zeta\omega_n} = -\frac{4}{Re(s_{\max})} \tag{26}$$

$$\Rightarrow Re(s_{\max}) = -\frac{4}{1} = -4$$

$$\Rightarrow Im(s_{\max}) = \pm\left(\frac{4}{0.49}\right)\sqrt{(0.49)^2 - 1} = \pm j7.116$$

$$\Rightarrow s_{max} = -4 \pm j7.116$$

II. The compensated system is of type 2 due to the additional pole at s=0 introduced by the integrator term of the PID compensator. The desired steady state error condition can then be used to solve for the integrator gain, K_I.

$$ess|_{parab} = \lim_{s\to 0} s\frac{\frac{1}{s^3}}{\left(1+G(s)G_{PID}(s)\right)} = \lim_{s\to 0}\frac{1}{s^2\left(1+G(s)G_{PID}(s)\right)} \tag{27}$$

$$\Rightarrow ess|_{parab} = \lim_{s\to 0}\frac{1}{s^2\left(1+\frac{5/12}{s(s+5/3)}\left(K_P+K_D s+\frac{K_I}{s}\right)\right)}$$

$$= \frac{1}{0.25K_I} = 0.25$$

$$\Rightarrow K_I = 16$$

III. Using the value of K_I solved for in part II and selecting $s_1 = -4 \pm j7.116$ as one of the dominant poles from part I, the combined system transfer function is evaluated.

$$K_P + K_D s_1 = -\frac{1}{G(s_1)} - \frac{K_I}{s_1} \tag{28}$$

$$\Rightarrow (K_P - 4K_D) + j(7.116K_D) = 99.13 + j108.16 - \frac{16}{-4+j7.116}$$

$$= 99.13 + j108.16 + 0.9604 + j1.7086 = 100.09 + j109.87$$

$$\Rightarrow K_D = \frac{109.87}{7.116} = 15.44 \approx 15$$

$$\Rightarrow K_P = 100.09 + 4K_D = 161.85 \approx 160$$

$$\Longrightarrow G_{PID}(s) = 160 + 15s + \frac{16}{s}$$

A root locus plot of the PID compensated closed loop DC motor system is shown in Figure V-15.

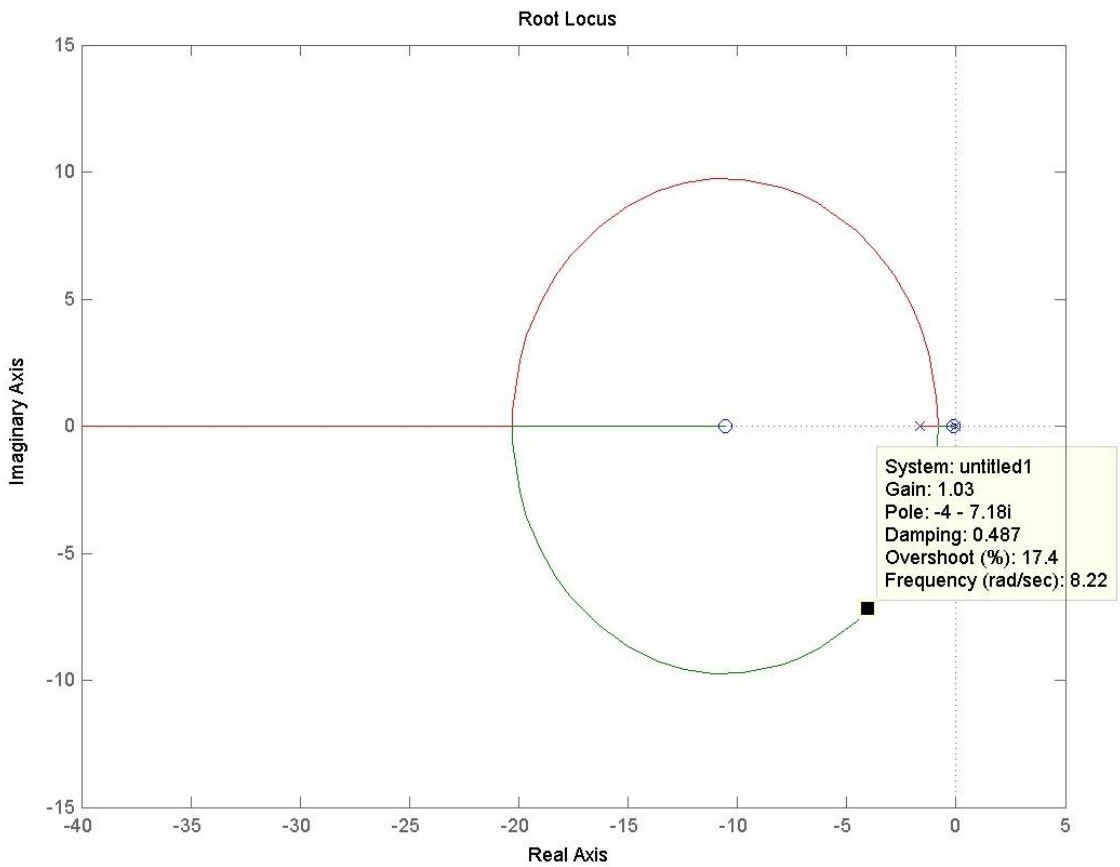


Figure V-15 – Root locus plot of the closed loop PID compensated DC motor system.

With the classical design of the PID control parameters, the all-digital counting frequencies can be determined. The system is modeled such that the drive actuator for the model has an inherent gain of 10. As a result, the controller gains are reduced by a factor of 10 for implementation. For the simulation, the loaded motor will track a velocity of 10 rpm which equates to a 60 Hz square wave output from an optical encoder that produces 360 pulses per revolution. As a guide from (15), the base counting frequency, f_A, should be designed to be larger than $2 \cdot 60\text{Hz}=120$ Hz. Selecting $f_A = 1200$ Hz satisfies this criteria. The subsequent frequency parameters can be calculated as follows:

$$f_P = f_A K_P = 19200 \text{ Hz}$$
$$f_I = f_A K_I = 1920 \text{ Hz}$$
$$f_D = f_A K_D = 1800 \text{ Hz}$$

The baseline and ADPID systems were re-simulated. The position and velocity outputs for the baseline system and the all-digital compensator are shown in Figure V-16 and Figure V-17 respectively. Significant steady-state error in the tracking velocity occurred in both the baseline and all-digital systems (17.7% and 14.2% respectively). The all-digital system also exhibited substantial overshoot (88.9%) indicating, as in the case with the previous simulation, sensitivity to the tuning parameters.

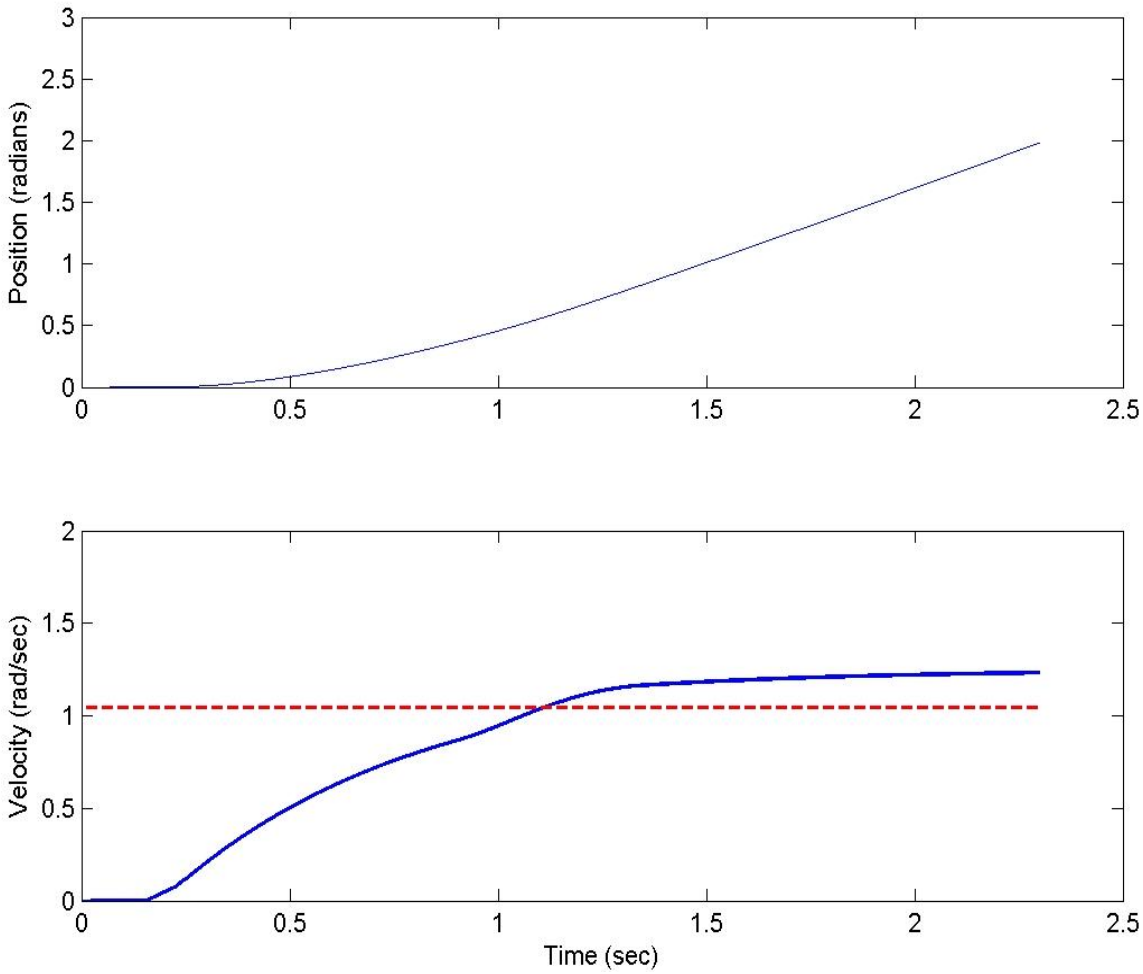


Figure V-16 – Position and velocity output of a PID compensated DC motor with load tracking a reference velocity of 10 rpm.

The all-digital system was re-implemented for proportional only control. The base frequency and the proportional gain for the controller were held at 1200Hz and 16 respectively, and the effects of integral and derivative action were removed. The steady state output and associated control actuation signal that resulted are shown below in Figure V-18. The system attained a steady-state value of approximately 1.21 rad/sec, settled within approximately 1.55 seconds and exhibited a steady state ripple of +5.4 to -6.4% about the mean steady state value.

Based on the known sensitivity of the integrator gain and the magnitude of overshoot exhibited in the initial simulation, the system was revised and re-simulated for values of $[K_P, K_I, K_D] = [7, 0.16, 4.5]$. The simulation results are shown in Figure V-19. The transient performance characteristics for the simulations discussed in this section are shown in Table V-4.

Table V-4 – Performance characteristics of the baseline and all-digital PID compensated armature-controlled DC motor plant with load.

Description	**Baseline**	**AD-PID**	**AD-P**	**Rev. ADPID**
Peak Overshoot (%)	17.7%	88.89%	21.75%	44.35%
Output Steady-State Ripple (%)		+6.4 to -7.4%	+5.4 to -6.4%	+7.0 to -6.1%
Steady-State Tracking Error (%)	17.7%	14.2%	15.6%	26.73%
Approximate S-S. Velocity ($\frac{\text{rad}}{\text{sec}}$)	1.23	1.20	1.21	1.41

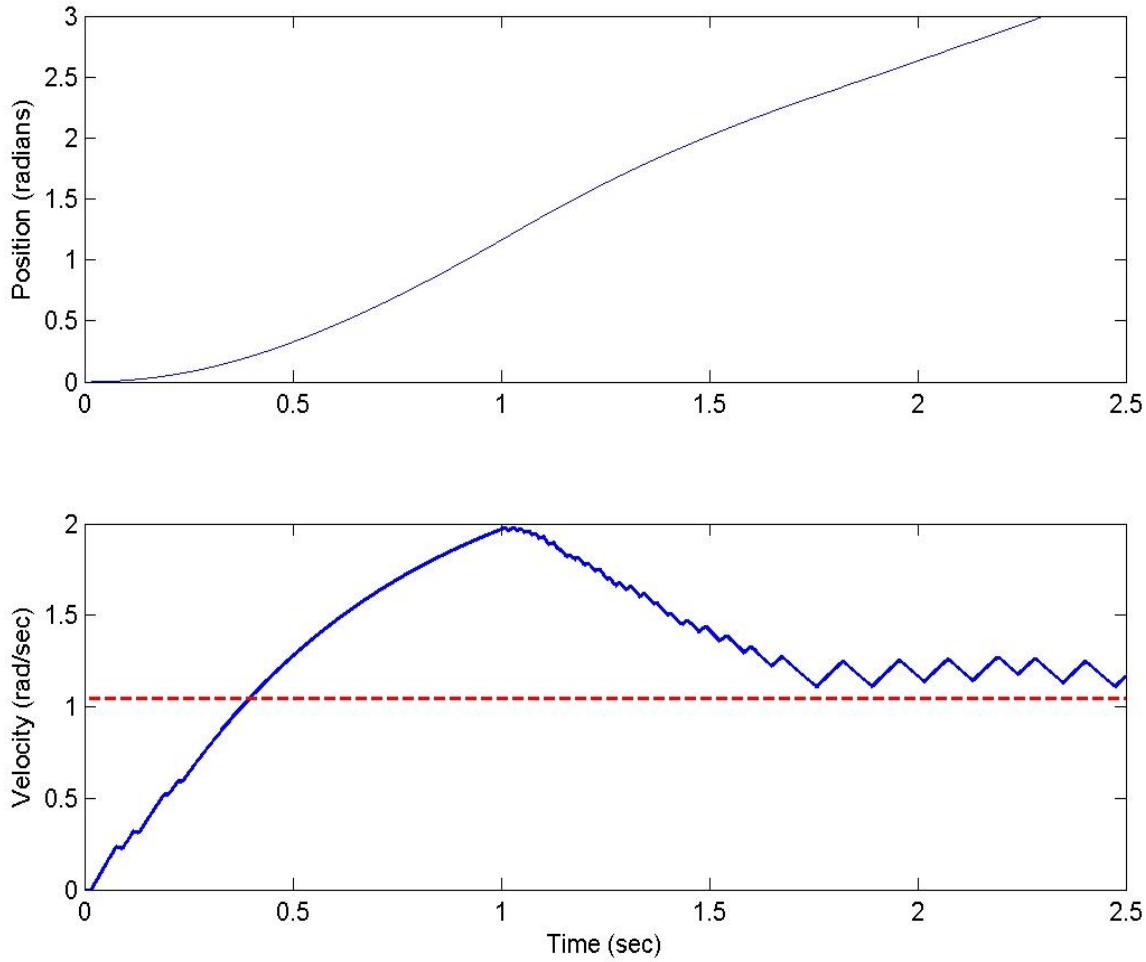


Figure V-17 – Position and velocity output of the ADPID compensated DC motor with inertial load tracking a reference velocity of 10 rpm.

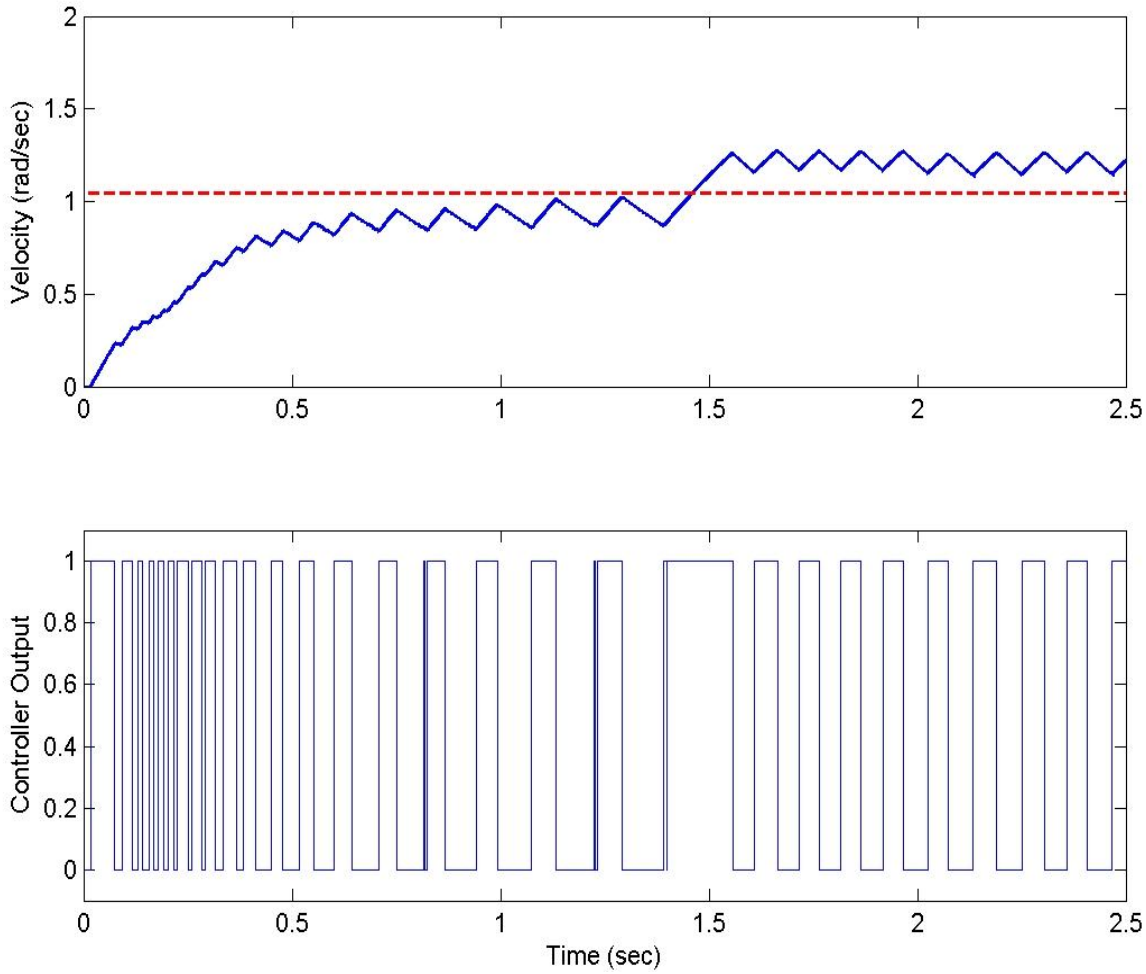


Figure V-18 – Proportional only control of a motor plant tracking a rotational velocity of 10 rpm.

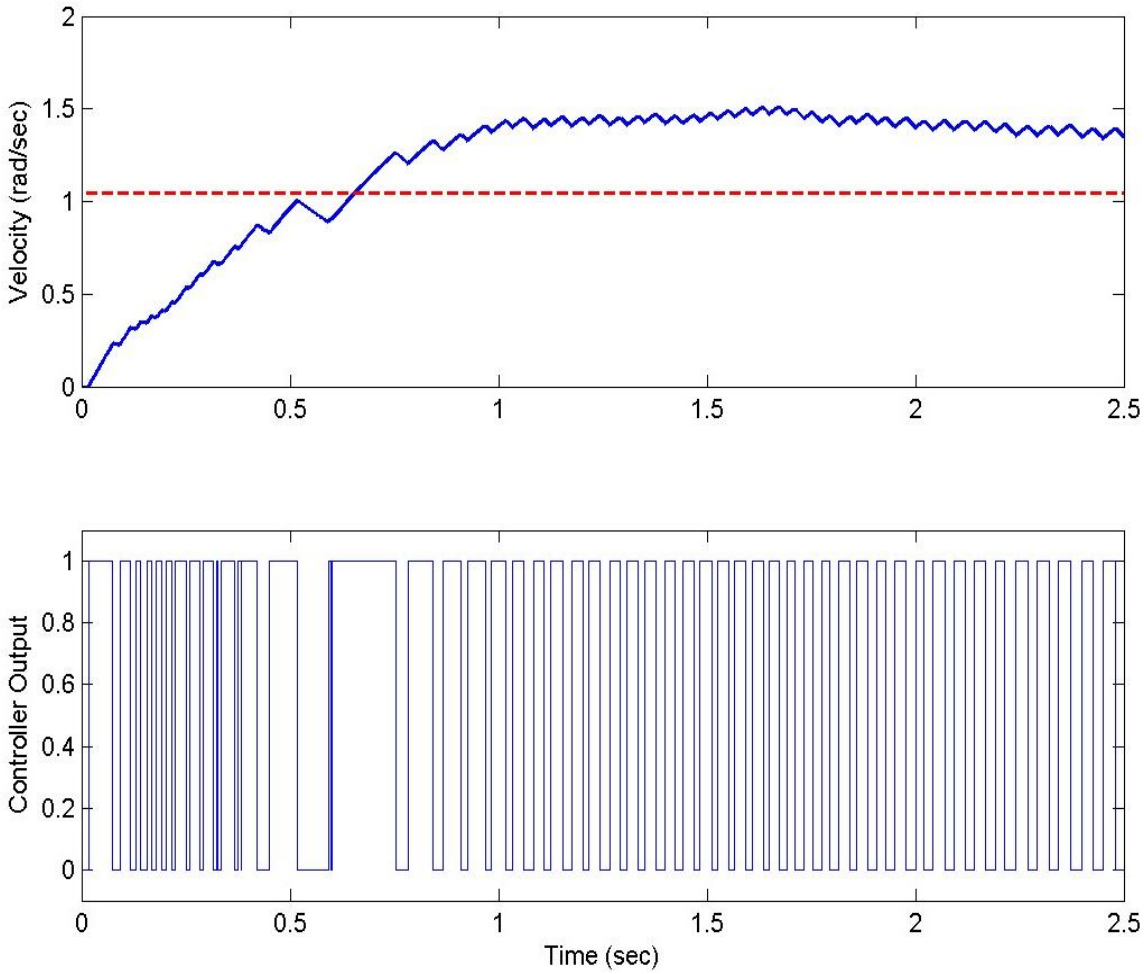


Figure V-19 – PID control of a motor tracking a velocity of 10 rpm with increased derivative gain, decreased proportional gain and significantly reduced integral gain.

C. STABILITY DISCUSSION

In examining the affect of the all-digital PID compensator and its operating modes on the stability of the underlying system under control, a number of parallels can be drawn between the all-digital system and systems currently existent within the realm of modern control theory.

The system produces a variable frequency, variable pulse-width rectangular wave output that is akin to a PWM device. The binary control output (on or off) effects a type of system compensation that is effectively similar to a “bang-off” control law. If the error or counting mechanism underlying the ADPID control system saturates or becomes “stuck”, the resultant input to the system under control would be a constant DC value of 1 scaled by the constant voltage gain (Vcc) of the electronics driving the interface. In this case, the stability of the controlled system would be equivalent to the stability of an open loop system with a gain of K=Vcc. In the case of saturation in the other direction, the resultant input to the system under control would be a value of 0. In this case the trajectory and relative stability of the previously excited system would be dependent on its previous state and internal dynamics.

The most interesting stability issue related to the underlying all-digital PID compensator involves investigation of the stability of the system excited by a variable frequency input. It is necessary to determine if frequencies of excitation exist that would “resonate” within the system driving it to a state of instability.

In [31], the effects of a pulse-width modulated actuation signal or input on systems that are both stationary and linear were investigated. The pulse-width modulation was parameterized in terms of the maximum amplitude of the actuation signal exciting the system, the sampling period of the modulated output and the input scaling factor, i.e. the inverse of the PWM input value that results in a 100% duty cycle output of the modulator. The stability analysis method proposed in [31] involves forming a piecewise description of the system operation under the possible actuation states of the modulator and linearizing it over the modulation period. A root-locus type analysis is conducted on the underlying linearized system and the bounds on the gain K serve as bounds on the product of the PWM scaling factor and amplitude, forming a sufficient condition for asymptotic stability. The author also notes that selecting a scaling factor such that the PWM signal is near saturation is a dominant factor in determining the sufficient

condition. The variable sampling period of the resultant signal from the proposed ADPID compensator requires an extension of this method for computation and analysis of the underlying system stability over the set of possible output periods achievable in the control law. The development of boundaries on the variable “scaling factor” implicit in the implementation as a result of the tuned control parameters is also required to extend this method. An extension in this manner would be reliant on significant simulation and statistical analysis of the proposed model for the specific class of systems for which its application is intended.

In [32], the stability of pulse-width modulated feedback systems is analyzed in the continuous time domain as opposed to their typical treatment in the discrete time domain due to their switching behavior. The extension introduces the effects of a “smoothing multiplier” and sampling on the system to achieve the continuous-time transformation. A restriction on the closed loop feedback gain that results in bounded input bounded output stability for a subset of systems is developed.

In assessing the performance of the closed-loop all-digital PID system under control, it is important for desired steady state transient specifications to be reachable. In the simulations presented, the closed-loop ADPID-DC motor plants converged to a steady state velocity output with periodic steady-state velocity ripple. In addition, at steady state the effective period of the control law output of the ADPID converged to a constant value. Methods for bounding output velocity ripple in PWM motor control [33] - [34] and for assuring steady-state system stability in the presence of ripple [35] remain active research areas in control systems theory. A proclivity for the use of pulse-width modulators in the class of digital control systems is generally attributed to the benefits afforded by their availability in low-cost, basic forms. The method of reducing steady-state ripple proposed in [34] relies on significant statistical simulations and Monte Carlo analysis with respect to the motor model and control parameters. The author works toward a more analytical method of analysis for a subset of systems in [33]. A sufficient

condition with respect to pulse-width modulator parameters for local stability of a second order system under PWM actuation based on Monte Carlo simulation is posed in [35]. It is the belief of the author that the “best” next step in further stability (and ultimately suitability) analysis for the use of the ADPID with the class of plants previously defined (those with first or second-order system dynamic models and plant-controller interfaces realizable in the form of incremental encoder technologies) is continued pursuit of extending the methods posed in [31] and [35] to include the variable sampling period affects and the relationship between the selection of the tunable control parameters and the effective pulse-width scaling factor that results in the all-digital PID compensator control law formed by the model presented herein.

Copyright © David Michael Feinauer 2011

VI. CONCLUSIONS / FUTURE WORK

A summary of the author's contributions and the ability for the model to address the current issues discussed in chapter III are presented in this chapter. Opportunities for future work related to: further development of the all-digital model, stability analysis of the resulting model, porting of the model to a hardware description language-based instantiation, simulation and testing of the resulting digital implementation and development of an auto-tuning routine are discussed.

A truly novel design methodology in the realm of digital control systems was presented in this work. The approach capitalizes on the inherent digital nature of many plants or systems to be controlled and accepts the output of a pulse train from a digital transducer, such as that of an incremental optical encoder, as direct input. The innovative design also implements a pulse-width modulated output in a digital manner to drive the digital plant. To achieve the original, all-digital implementation described herein, the concepts of proportional, integral and derivative control actions were re-envisioned around a counting method that provides a digital representation of the duration of the digital pulses and the frequency at which they occur.

The all-digital PID compensator model proposed herein contains a number of improvements from the preliminary model proposed in [28]. These contributions include:

- An all-digital formation of the error sign signal.
- A simplification of the counting methods and a reduction of the operating states required for implementing the proportional and derivative counting stages.
- The use of an edge detector and the inherent operating states of the binary up/down counter to map the finite state machine description to a causal block diagram-like description of the system for modeling and simulation.

- The implementation of modular, reusable models as described in Appendix A through Appendix F and the creation of library functions to facilitate reuse and version control as presented in Appendix F.
- The design and implementation of an improved method of determining the tracking error of the system based on current and previous signal and state information.
- Development of the model based on functional and operational descriptions of the constituent components with an eventual digital logic implementation in mind, allowing for a design that is independent of a particular hardware technology and portable to a variety of platforms including FPGAs, ASICs or a system of discrete digital logic components.
- Simulation of the model in The MathWorks *MATLAB* and *Simulink* Release 2010b packages—tools currently popular for modeling and simulation in the control systems engineering domain.
- The design and implementation of subcomponents independent of the proposed all-digital PID compensator, but necessary for interfacing the controller to plant models typical of what one would find in practice. These devices include an optical quadrature encoder, a voltage to PWM signal converter and a quadrature to analog position decoder described in Appendix E.
- The development of an initial heuristic rule regarding the selection of the control parameters.
- A presentation of results demonstrating the sensitivity of the proportional and integral gains to the selection of the base accumulator counting frequency.

Owing to the time complexity of numerically simulating and solving for the digital logic circuitry whose signals transition on a time scale orders of magnitude smaller than the time scale on which a response can be actuated in the physical system models, the performance of this system with respect to disturbance rejection in comparison to that of the classical PID compensated system is not present. Additionally, simulations for systems whose higher order poles are modeled are also not contained herein. Although not present in this work, investigations in this manner are possible and are being pursued.

The all-digital PID compensator developed for this work presents a number of opportunities for future research, the first of which relates to further development of the proposed model for improved stability and performance. The all-digital phase-locked loop controller proposed in [26] and [27] is a type of all-digital frequency tracking control like the all-digital PID implementation proposed herein. It was observed in [26] that when the PLL was in a “near-locked state”, increased ripple occurred in the system output due to the frequent switching resultant from non-zero digital count values representative of system error despite its proximity to the desired tracking frequency. The all-digital PID compensator appears to exhibit similar steady-state behavior when it is near-tracking the reference signal. Opportunity exists to develop a solution analogous to one proposed in [26] for PLLs. Additionally, the ADPLL methods were shown to achieve steady-state tracking at the expense of undesirable transient responses. The ADPID compensator implemented in this work exhibited undesirable error with respect to the commanded setpoint to be tracked, but could be tuned to achieve steady-state response with transient specs similar to those expected from classical PID control.

Initial ideas for the pursuit of this research involve defining a neighborhood of the desired reference signal in terms of the tracking error and rate of change of the tracking error (the outputs of the proportional and derivative stages). Once defined, any errors that occur when the system is operating in that neighborhood of the reference would result in a scaling or reduction of the impact of the error count, thus reducing the resultant

compensation from the overall controller. Additionally, the use of previous state information in the formation of the error signal, innovated throughout the course of this work and presented in section V.A, could be extended to a greater depth by increasing the "window" of signal values used to determine the relative error of the inputs. This increased information should result in an increased resolution with respect to the relative frequencies of the input signals which could affect a reduction in the rise time and ripple observed in the all-digital closed-loop system.

Other possible avenues for addressing the steady state ripple include adapting the model to an all-digital hybrid controller wherein the control law would switch to a reduced-ripple, all-digital phase-locked loop implementation once the PID based controller actuates a system velocity within a neighborhood of the desired reference. If a subset of potential applications for the controller and the associated plants to be controlled were defined, for example permanent magnet DC motors, then an extension of the Monte Carlo based methods proposed in [34] could also provide an avenue for addressing ripple about the attained steady-state velocity.

Another rich opportunity for future research direction involves the porting of the model to the domain of programmable logic devices by developing a high-level HDL-based description of the proposed controller. From this description, a number of target hardware implementation platforms could be identified and the performance could be simulated on four levels: pre-synthesis software simulation and verification based on the HDL model, co-simulation for stability analysis and performance verification through hardware vendor-provided *MATLAB e*xtensions, post-synthesis software simulation and verification based on the HDL model and the selected hardware platform and finally design verification of the model implemented on a reprogrammable logic device such as an FPGA.

It is important to note that the time scale on which signal transitions occur in the proposed all-digital PID compensator is many orders of magnitude smaller than the time scale on which a response in a physical system or plant under control can be actuated. This difference in time-scale greatly affects the ability for accurate or timely numerical simulation of the overall combined system performance. The increased adoption of digital logic devices in the realm of control systems continues to be met with new digital hardware vendor-supplied tools and extensions for numerical co-simulation. With such a software package, an HDL based description of the all-digital model could be simulated with the appropriate tools and time scale in the digital electronics realm and be interfaced to *MATLAB* and *Simulink* for co-simulation with models of plants to be controlled for which a different time scale is appropriate. Simulation in this manner allows for an overall reduction of simulation times and an increased number of simulation results necessary for the previously discussed stability and parameter tuning pursuits.

With a model simulated and verified with respect to its internal functionalities and suitability for application to physical systems, opportunities exist for developing an ASIC-based controller implementation. Engineering the system on this platform should result in a low cost, low-power, small footprint, commercial-off-the-shelf PID compensator targeting embedded systems applications. Additionally, development of the model as a “soft core” intellectual property–based control system would allow for it to be a stock model or entity available in an FPGA library. Development in this manner would allow the design to remain platform independent and avail it for use in reconfigurable system-on-a-chip devices.

The expansion of the proposed device to an HDL-based model combined with the improved possibility for numerical simulation improves the likelihood of defining the set of possible PWM output periods for a given set of control parameters. The improved ability for generating simulation data for a number of parameters also presents the possibility of characterizing the system based on Monte Carlo methods. The numerical

simulation results combined with extensions to prior art discussed in section V-C could assist in determining likely bounds on the tuning parameters of the all-digital PID system that will maintain closed-loop system stability and help in the pursuit of defining a sufficient condition for which a closed-loop all-digital PID compensated system would remain stable.

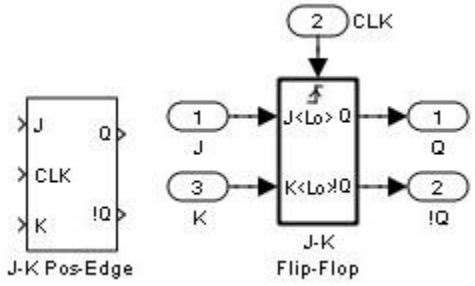
Lastly, it remains true that despite its simplicity and wide use, the manual tuning of PID control parameters by end users of commercial controllers is a challenge and often results in poor performing control loops. Research on a processor based start-up and auto-tuning routine appropriate for the all-digital PID compensator could be pursued. Once identified, the routine could be developed as a “soft-core” process that could become a part of the high-level model appropriate for a programmable logic device-based or FPGA-based implementation that is independent of a specific hardware platform.

Copyright © David Michael Feinauer 2011

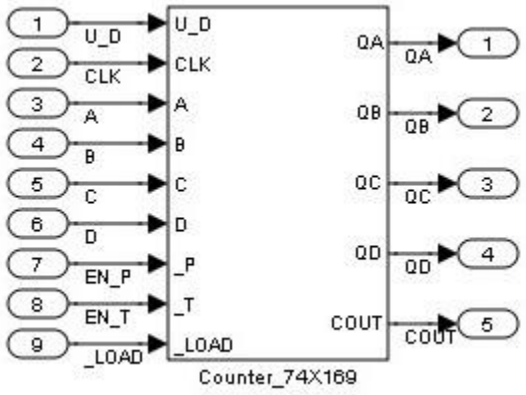
APPENDIX A

The 16-bit counter (Counter16Bit.mdl) referenced in this document was constructed from the *Simulink* models depicted below (JKPosEdge.mdl and Counter74X169.mdl). The graphical implementation of Counter16Bit.mdl follows.

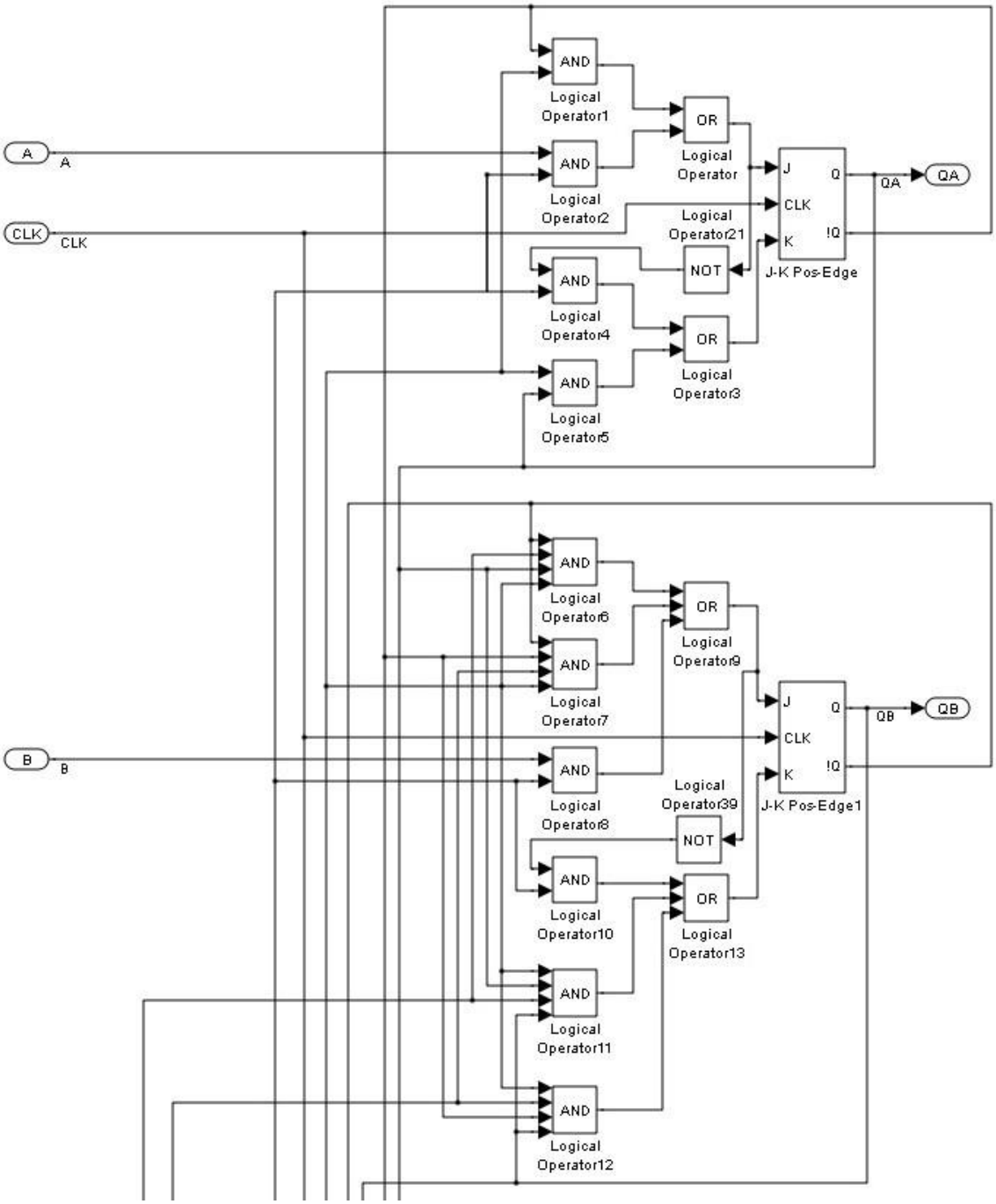
JKPosEdge.mdl

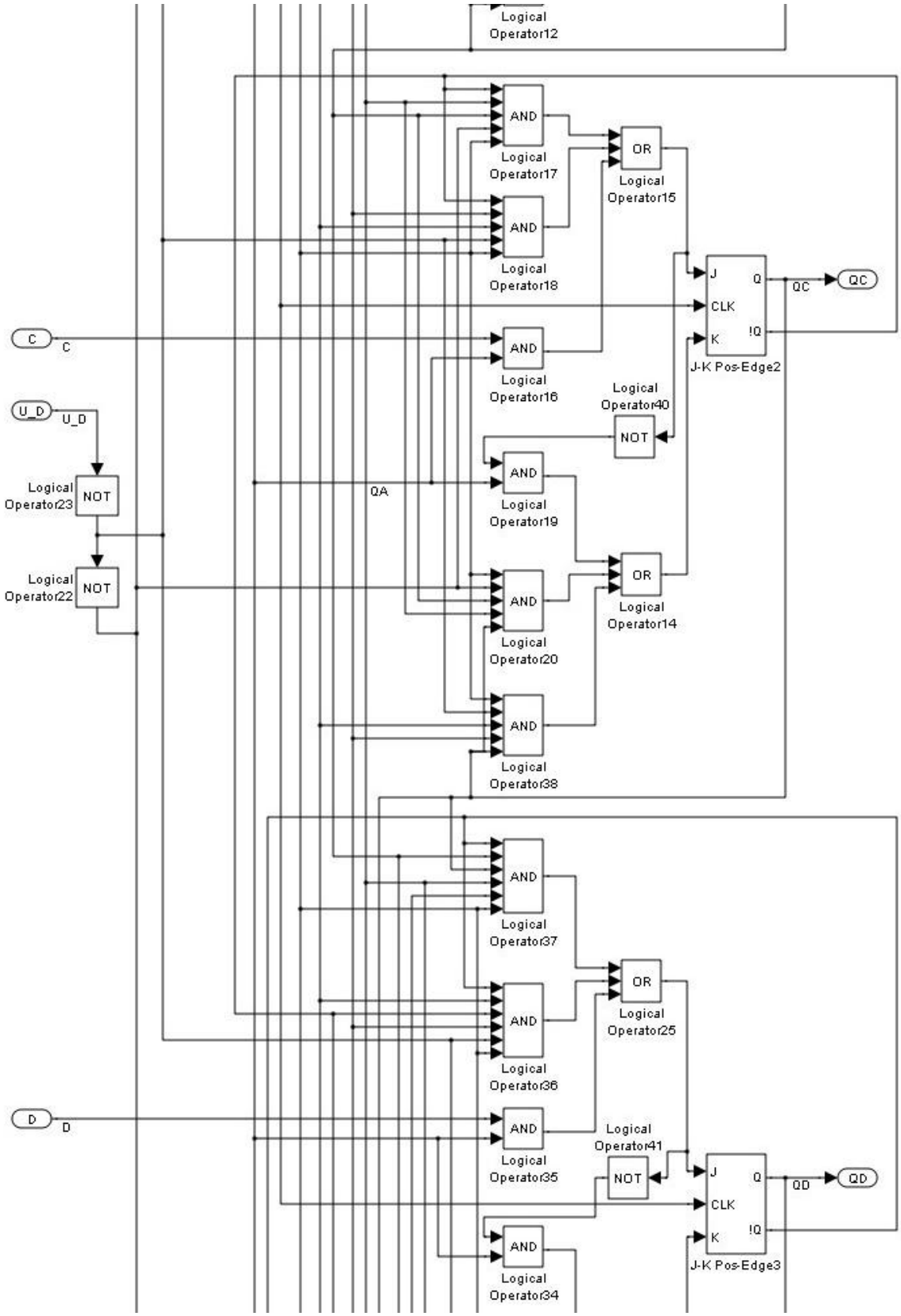


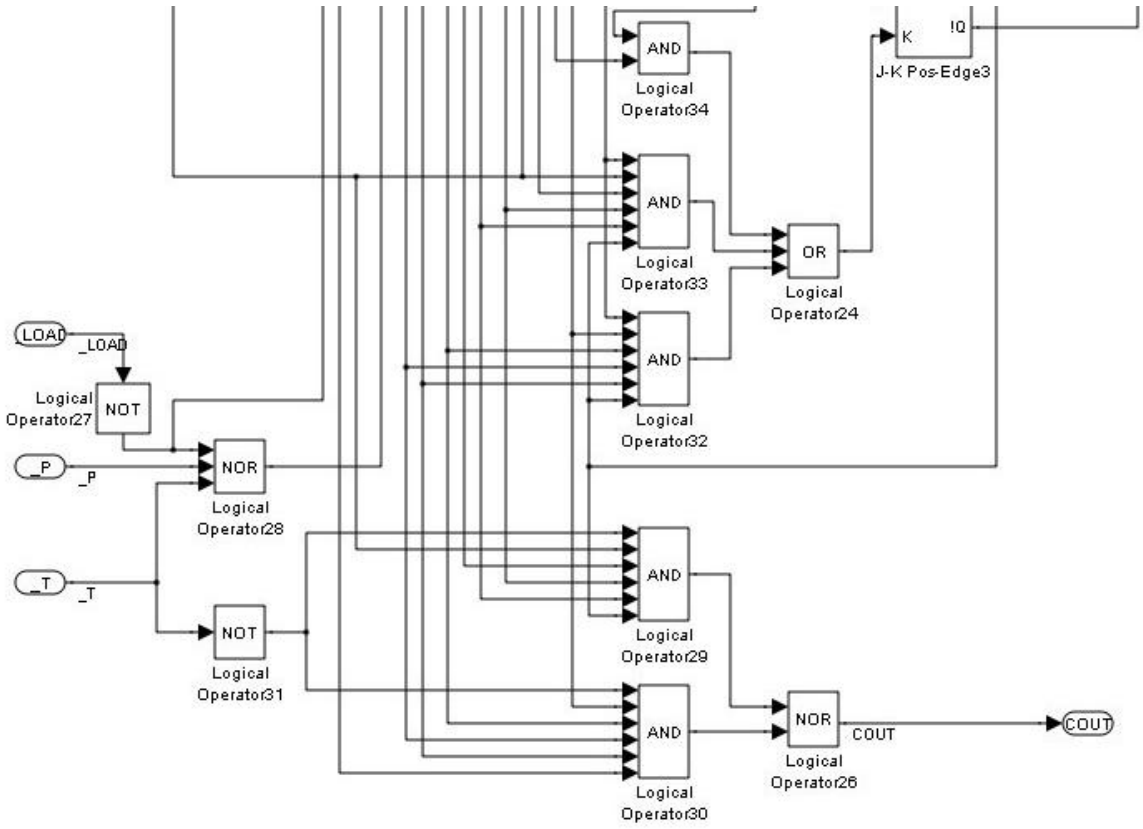
Counter74X169.mdl



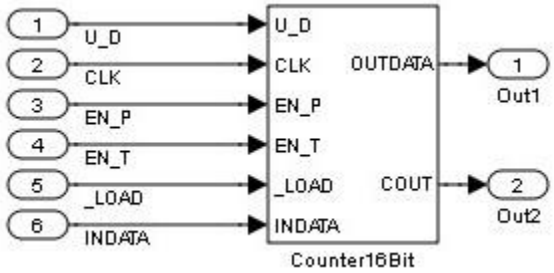
Please note that an exploded view of the Counter74X169 module follows—it spans three pages.



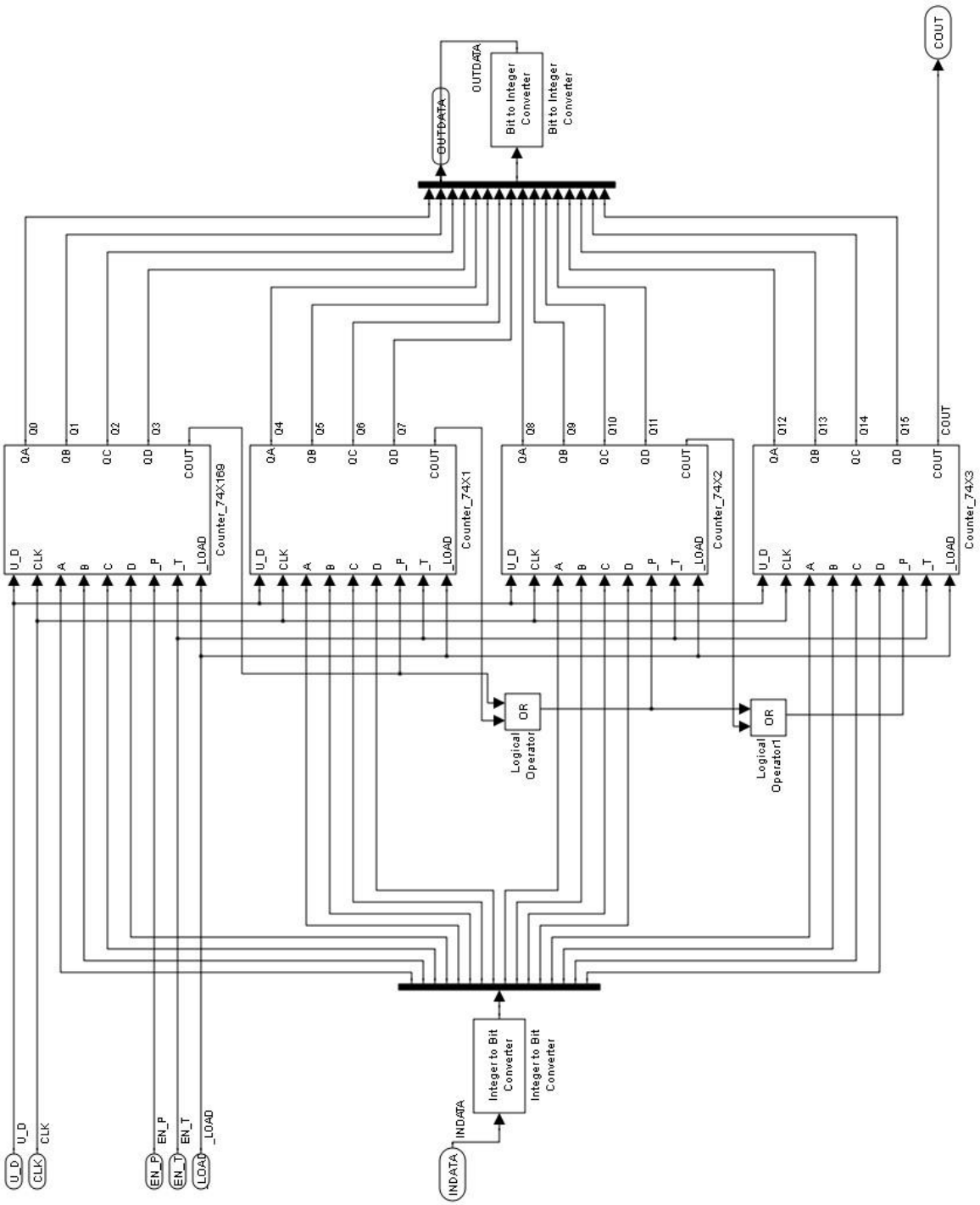

Logical Operator12
AND
Logical Operator17
OR
Logical Operator15
AND
Logical Operator18
J
Q
QC
QC
CLK
K
!Q
J-K Pos-Edge2
C
C
AND
Logical Operator16
Logical Operator40
NOT
U_D
U_D
AND
Logical Operator19
QA
Logical Operator23
NOT
OR
Logical Operator14
Logical Operator22
NOT
AND
Logical Operator20
AND
Logical Operator38
AND
Logical Operator37
OR
Logical Operator25
AND
Logical Operator36
D
D
AND
Logical Operator35
Logical Operator41
NOT
J
Q
QD
QD
CLK
K
!Q
J-K Pos-Edge3
AND
Logical Operator34



Counter16Bit.mdl



An exploded view of the Counter16Bit.mdl comprised of 4 cascaded Counter74X169 modules follows on the next page.


U_D
CLK
EN_P
EN_T
_LOAD
INDATA
Integer to Bit Converter
Integer to Bit Converter
Counter_74X169
Counter_74X1
Counter_74X2
Counter_74X3
Logical Operator
Logical Operator1
OR
OR
OUTDATA
Bit to Integer Converter
Bit to Integer Converter
COUT

APPENDIX B

The *Simulink* S-function implementation of a generic counter is below. The counter requires one parameter specifying the size of the counter. For a parameter value that is a strictly positive integer, N, the result is a positive-edge triggered, synchronous, N-bit binary up/down counter with asynchronous load, active low enable and carry out ripple along with the associated active low carry enable for cascading multiple stages. The maximum value that can be represented by the counter is 2^N – 1. The counter wraps or overflows about this point. Selecting N=4 results in a module with a functionality similar to the 74LS169 digital logic device previously discussed. Specifying a model with N = 'Inf' (infinity) results in an integer-based positive-edge triggered counter without an explicit limit or maximum count. Operating the counter in this manner proved useful at addressing some of the variable type requirements of the core *Simulink* functional blocks.

```
function Counter2Nb(block)
%Counter2Nb
%   A MATLAB S-function implementing an N-bit binary up/down counter with
%   active low enable, asynchronous load, and carry out ripple
%   functionality. The allows for an input argument to function as an
%   integer counter or a binary counter of 2^n bits.
%
%   The block requires a parameter to indicate the size of the counter. The
%   parameter can contain one of two different types of arguments:
%       1) A positive integer, N, representing the bit length of a counter
%       that will count to a max of 2^N - 1.
%       2) The number Inf (infinity) is used to indicate an integer based
%       counting mode with no overflow.
%
%   $Revision: 2.0 $
%
% The setup method is used to setup the basic attributes of the
% S-function such as ports, parameters, etc.
%
setup(block);

%endfunction
```

```
%% Function: setup ====================================================
% Abstract:
%   Set up the S-function block's basic characteristics such as:
%   - Input ports
%   - Output ports
%   - Dialog parameters
%   - Options
%
function setup(block)

% Register number of ports
block.NumInputPorts  = 6;
block.NumOutputPorts = 2;

% Setup port properties to be dynamic (inherited) or defaults
block.SetPreCompInpPortInfoToDynamic;
block.SetPreCompOutPortInfoToDynamic;

% Override input port properties
block.InputPort(6).Dimensions  = 1;
block.InputPort(6).DatatypeID  = 0;  % double
block.InputPort(6).Complexity  = 'Real';

block.InputPort(1).DirectFeedthrough = true;
block.InputPort(2).DirectFeedthrough = true;
block.InputPort(3).DirectFeedthrough = true;
block.InputPort(4).DirectFeedthrough = true;
block.InputPort(5).DirectFeedthrough = true;
block.InputPort(6).DirectFeedthrough = true;

block.InputPort(1).SamplingMode = 'Sample';
block.InputPort(2).SamplingMode = 'Sample';
block.InputPort(3).SamplingMode = 'Sample';
block.InputPort(4).SamplingMode = 'Sample';
block.InputPort(5).SamplingMode = 'Sample';
block.InputPort(6).SamplingMode = 'Sample';

% Override output port properties
block.OutputPort(1).Dimensions  = 1;
block.OutputPort(1).DatatypeID  = 0; % double
block.OutputPort(1).Complexity  = 'Real';
block.OutputPort(1).SamplingMode = 'Sample';

block.OutputPort(2).Dimensions  = 1;
block.OutputPort(2).DatatypeID  = 0; % double
block.OutputPort(2).Complexity  = 'Real';
block.OutputPort(2).SamplingMode = 'Sample';
```

```
% Register parameters
block.NumDialogPrms = 1;
block.DialogPrmsTunable = {'Tunable'};

% Register sample times
%  [0 offset]            : Continuous sample time
%  [positive_num offset] : Discrete sample time
%
%  [-1, 0]               : Inherited sample time
%  [-2, 0]               : Variable sample time
block.SampleTimes = [-1 0];

% Specify the block simStateCompliance. The allowed values are:
%    'UnknownSimState', < The default setting; warn and assume
%    DefaultSimState
%    'DefaultSimState', < Same sim state as a built-in block
%    'HasNoSimState',   < No sim state
%    'CustomSimState',  < Has GetSimState and SetSimState methods
%    'DisallowSimState' < Error out when saving or restoring the model sim
%    state
block.SimStateCompliance = 'DefaultSimState';

% -----------------------------------------------------------------
% The MATLAB S-function uses an internal registry for all
% block methods. You should register all relevant methods
% (optional and required) as illustrated below. You may choose
% any suitable name for the methods and implement these methods
% as local functions within the same file. See comments
% provided for each function for more information.
% -----------------------------------------------------------------

block.RegBlockMethod('CheckParameters', @CheckPrms);
block.RegBlockMethod('ProcessParameters', @ProcessPrms);
block.RegBlockMethod('PostPropagationSetup', @DoPostPropSetup);
block.RegBlockMethod('InitializeConditions', @InitializeConditions);
block.RegBlockMethod('Start', @Start);
block.RegBlockMethod('Outputs', @Outputs);     % Required
block.RegBlockMethod('Update', @Update);
%block.RegBlockMethod('Derivatives', @Derivatives);
block.RegBlockMethod('Terminate', @Terminate); % Required

%end setup
```

```
%% CheckParameters:
  %   Functionality    : Called in order to allow validation of the
  %                      block dialog parameters. You are
  %                      responsible for calling this method
  %                      explicitly at the start of the setup method.
  %   C-Mex counterpart: mdlCheckParameters
  %
function CheckPrms(block)

size = block.DialogPrm(1).Data;

if (size ~= Inf)
    Nint = int32(size);
    if (Nint < 1)
    error('The dialog parameter must be a positive integer or Inf.')
    end
end

%endfunction

%% ProcessParameters:
  %   Functionality    : Call to allow an update of run-time parameters.
  %   C-Mex counterpart: mdlProcessParameters
  %
function ProcessPrms(block)
  block.AutoUpdateRuntimePrms;

%endfunction

%% PostPropagationSetup:
%   Functionality    : Setup work areas and state variables. Can
%                      also register run-time methods here
%   Required         : No
%   C-Mex counterpart: mdlSetWorkWidths
%
function DoPostPropSetup(block)
block.NumDworks = 3;

  block.Dwork(1).Name            = 'xClk';
  block.Dwork(1).Dimensions      = 1;
  block.Dwork(1).DatatypeID      = 0;      % double
  block.Dwork(1).Complexity      = 'Real'; % real
  block.Dwork(1).UsedAsDiscState = true;

  block.Dwork(2).Name            = 'xOut';
  block.Dwork(2).Dimensions      = 1;
  block.Dwork(2).DatatypeID      = 0;      % double
  block.Dwork(2).Complexity      = 'Real'; % real
  block.Dwork(2).UsedAsDiscState = true;
```

```
  block.Dwork(3).Name            = 'xCarry';
  block.Dwork(3).Dimensions      = 1;
  block.Dwork(3).DatatypeID      = 0;      % double
  block.Dwork(3).Complexity      = 'Real'; % real
  block.Dwork(3).UsedAsDiscState = true;


%end DoPostPropSetup


%% InitializeConditions:
%   Functionality    : Called at the start of simulation and if it is
%                      present in an enabled subsystem configured to reset
%                      states, it will be called when the enabled subsystem
%                      restarts execution to reset the states.
%   Required         : No
%   C-MEX counterpart: mdlInitializeConditions
%
function InitializeConditions(block)

block.Dwork(1).Data = 0;
block.Dwork(2).Data = 0;
block.Dwork(3).Data = 1;

%end InitializeConditions

%% Start:
%   Functionality    : Called once at start of model execution. If you
%                      have states that should be initialized once, this
%                      is the place to do it.
%   Required         : No
%   C-MEX counterpart: mdlStart
%
function Start(block)

block.Dwork(1).Data = 0;
block.Dwork(2).Data = 0;
block.Dwork(3).Data = 1;

%end Start
```

```
%% Outputs:
%   Functionality    : Called to generate block outputs in
%                      simulation step
%   Required         : Yes
%   C-MEX counterpart: mdlOutputs
%
function Outputs(block)
% Inputs
% 1 - UpDwn';
% 2 - Clk
% 3 - EnP
% 4 - EnT
% 5 - Load
% 6 - DIn

N = block.DialogPrm(1).Data;

UpDwn = block.InputPort(1).Data;
Clk = block.InputPort(2).Data;
EnP = block.InputPort(3).Data;
EnT = block.InputPort(4).Data;
Load = block.InputPort(5).Data;
DIn = block.InputPort(6).Data;

%Get previous output
tempCnt = block.Dwork(2).Data;
tempCarry = block.Dwork(3).Data;

if ~Load                                                    %Load Data
    if N == inf
        tempCnt = DIn;
        tempCarry = 1;
    else
        tempCnt = max(0,min(DIn,2^N-1));
        tempCarry = 1;
    end
elseif ~(EnP || EnT)                                        %Check enabled
       if (Clk && block.Dwork(1).Data == 0)                 %Cnt Pos Edge
           tempCnt = tempCnt + 2*UpDwn-1;
           tempCarry = 1;
           %Check Overflow
            if  N ~= inf
                if (UpDwn && (tempCnt == 2^N))              %Up Cnt Overflow
                    tempCnt = 0;
                    tempCarry = 0;
                elseif ~UpDwn && ~tempCnt                  %Dwn Cnt Overflow
                    tempCnt = 2^N - 1;
                    tempCarry = 0;
                end
            end
```

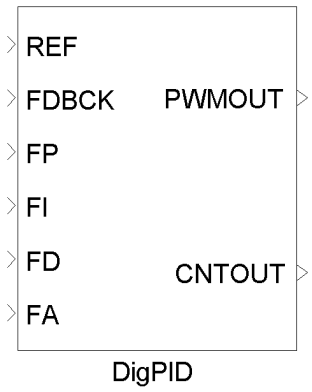
```
        end
end

block.OutputPort(1).Data = tempCnt;
block.OutputPort(2).Data = tempCarry;

%end Outputs

%% Update:
%   Functionality    : Called to update discrete states
%                      during simulation step
%   Required         : No
%   C-MEX counterpart: mdlUpdate
%
function Update(block)
block.Dwork(1).Data = block.InputPort(2).Data;
block.Dwork(2).Data = block.OutputPort(1).Data;
block.Dwork(3).Data = block.OutputPort(2).Data;

%end Update


%% Terminate:
%   Functionality    : Called at the end of simulation for cleanup
%   Required         : Yes
%   C-MEX counterpart: mdlTerminate
%
function Terminate(block)

%end Terminate
```

APPENDIX C

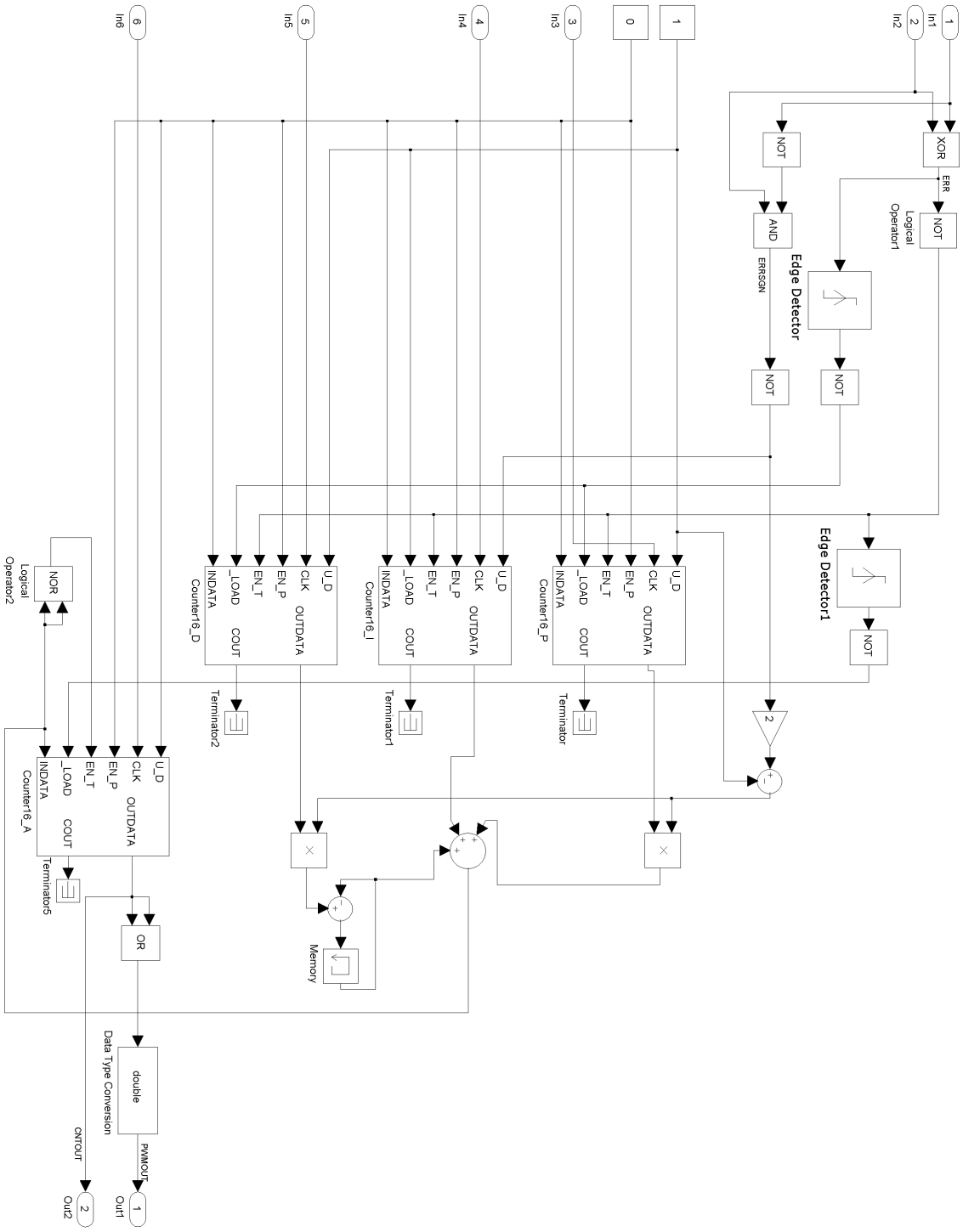
The *Simulink* model file for the proposed all-digital PID controller is detailed below. The controller is primarily constructed from the counter model described in the appendix above along with core blocks and functions from the *Simulink* library.

The initial all-digital PID controller model (DigPID.mdl) implementation is shown below

DigPID.mdl



An exploded view of the module and its core components follows on the next page.


In1
In2
In3
In4
In5
In6
XOR
ERR
NOT
Logical Operator1
AND
ERRSGN
Edge Detector
Edge Detector1
Counter16_P
Counter16_I
Counter16_D
Counter16_A
U_D
CLK
EN_P
EN_T
_LOAD
INDATA
OUTDATA
COUT
Terminator
Terminator1
Terminator2
Terminator5
Memory
OR
NOR
Logical Operator2
double
Data Type Conversion
PWMOUT
CNTOUT
Out1
Out2

APPENDIX D

The “next state” truth table for a digital logic function that determines the magnitude and sign of the error between the reference and feedback signals is given below. The magnitude and sign outputs map directly to the required counter control logic—the sign output drives the direction control signals (U_D) for the counters and the magnitude output drives the counter enable (EN_T) circuitry, determining when a count occurs.

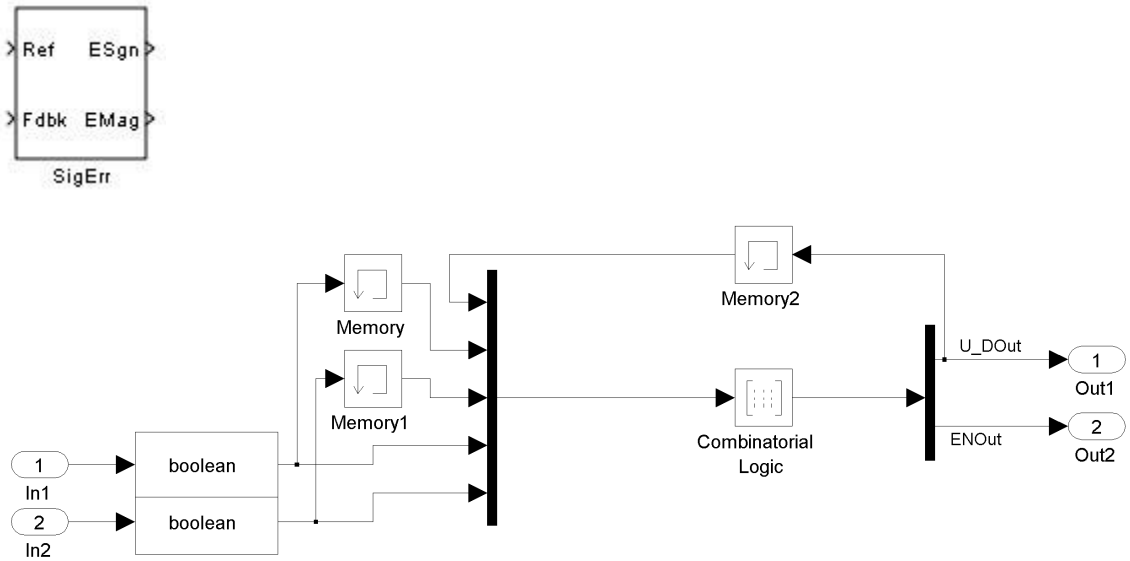
The active low device enable interface was taken into account when listing the binary word output representation [$ErrSgn_{n+1}$ $ErrMag_{n+1}$] in the truth table; the functional operation modes related to the outputs are described in the subsequent table. The color coding of the truth table indicates the operational sequence of the device as it relates to the underlying error signal that is being decoded. The states highlighted in red represent the transitions that occur when the feedback signal leads the reference and the device counts down as a result. Green is used to indicate the sequence of possible transitions when the reference leads the feedback signal and the device counts up. The aqua states represent cases where no transitions are occurring and the previous modes of operation are held. The orange highlighting designates the transitions that occur when the underlying relationship between the two input signals changes and the counting changes direction as a result. This operational description is also represented in the “operation” column where the green, red, aqua and orange rows of the table are designated by the operational descriptions Ref. Leads, Ref. Lags, Hold and Chg. Direction, respectively.

The digital circuit that implements the function in the table is shown in the SigErr3.mdl figure below the table. The circuit could be implemented with a combination of combinational and sequential logic elements. The table and circuit represent a system where the output is a function of the current and previous inputs as well as one of the previous outputs.

Operation	$ESgn_n$	REF_{n-1}	$FDBK_{n-1}$	REF_n	$FDBK_n$	$ESgn_{n+1}$	$EMag_{n+1}$
Hold	0	0	0	0	0	0	1
Ref. Lags	0	0	0	0	1	0	0
Chg. Direction	0	0	0	1	0	1	0
Ref. Lags	0	0	0	1	1	0	1
Ref. Lags	0	0	1	0	0	0	1
Hold	0	0	1	0	1	0	0
Ref. Lags	0	0	1	1	0	0	0
Chg. Direction	0	0	1	1	1	1	1
Chg. Direction	0	1	0	0	0	1	1
Ref. Lags	0	1	0	0	1	0	0
Hold	0	1	0	1	0	0	0
Ref. Lags	0	1	0	1	1	0	1
Ref. Lags	0	1	1	0	0	0	1
Chg. Direction	0	1	1	0	1	1	0
Ref. Lags	0	1	1	1	0	0	0
Hold	0	1	1	1	1	0	1
Hold	1	0	0	0	0	1	1
Chg. Direction	1	0	0	0	1	0	0
Ref. Leads	1	0	0	1	0	1	0
Ref. Leads	1	0	0	1	1	1	1
Chg. Direction	1	0	1	0	0	0	1
Hold	1	0	1	0	1	1	0
Ref. Leads	1	0	1	1	0	1	0
Ref. Leads	1	0	1	1	1	1	1
Ref. Leads	1	1	0	0	0	1	1
Ref. Leads	1	1	0	0	1	1	0
Hold	1	1	0	1	0	1	0
Chg. Direction	1	1	0	1	1	0	1
Ref. Leads	1	1	1	0	0	1	1
Ref. Leads	1	1	1	0	1	1	0
Chg. Direction	1	1	1	1	0	0	0
Hold	1	1	1	1	1	1	1

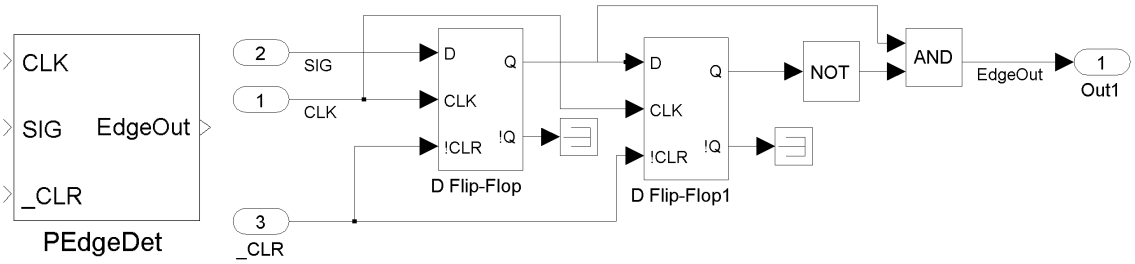
Output	Function
0 0	Count Down *(feedback leads reference)*
0 1	Hold Previous Output *(previously counting down)*
1 0	Count Up *(reference leads feedback)*
1 1	Hold Previous Output *(previously counting down)*

sigErr3.mdl



The edge detector below (PosEdgeDet.mdl) was also implemented for this model. It produces an impulse of at least one clock period when a rising edge is detected on the signal (SIG) input.

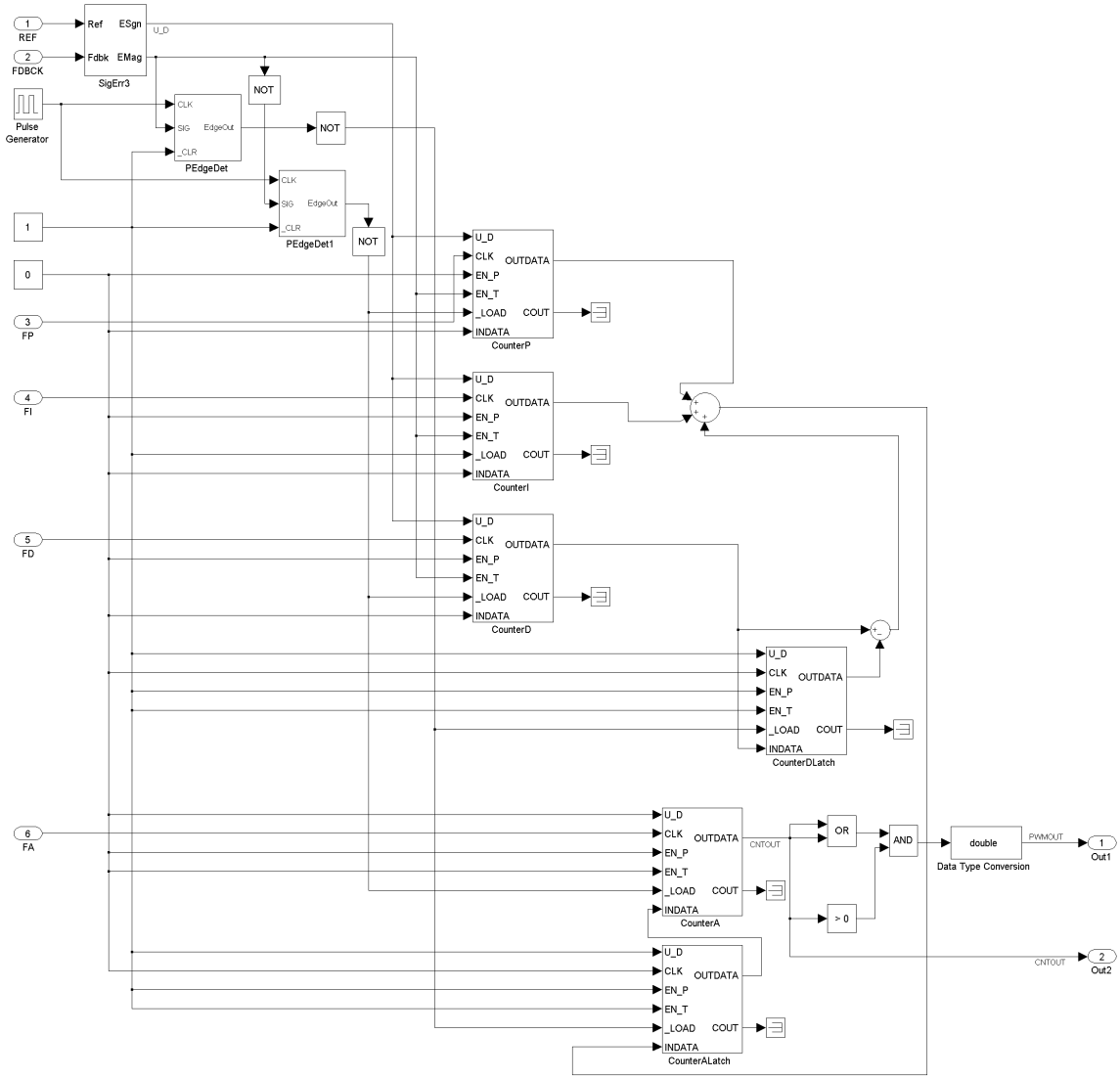
PosEdgeDet.mdl



The modified all-digital PID compensator shown below contains the following additions:

1. A new method for determining tracking error (SigErr3) as described above.
2. Explicit latching of the derivative and accumulator outputs on the falling edge of the error magnitude signal (Counter DLatch and Counter ALatch).
3. A modified edge detector (PEdgeDet) that holds the edge detection pulse for a duration determined by the clocked input.

DigHM3h.mdl



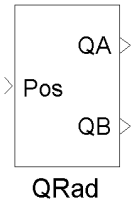
APPENDIX E

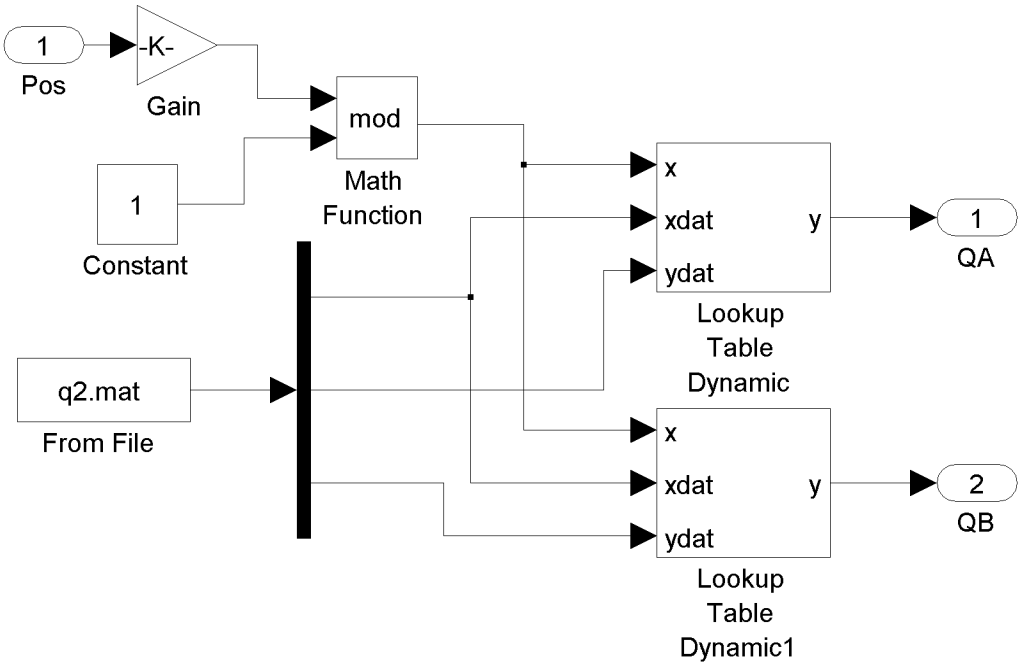
In addition to the models created to construct the proposed all-digital PID compensator, a number of custom models were created to simulate the physical interfaces typical of a real world system application. The details of three such models: a quadrature encoder, a voltage controlled pulse-width-modulator (PWM) and a quadrature decoder are contained in this appendix.

A *Simulink* model of an optical quadrature encoder that generates two square waves ninety degrees out of phase with a pulse generation frequency of one pulse per degree rotation, 360 pulses per revolution is presented below. The model assumes an input in radians and produces an output with a magnitude of 1. A lookup table of values, q2.mat, relating the outputs of the encoder to the position input is shown below. The relationship is based in degrees. The "gain" block present in the model is used to convert the input from radians to degrees before lookup.

Position (deg.)	***QA***	***QB***
0.000	0	0
0.249	0	0
0.250	1	0
0.251	1	0
0.499	1	0
0.500	1	1
0.501	1	1
0.749	1	1
0.750	0	1
0.751	0	1
0.999	0	1
1.000	0	0
1.001	0	0

QRad.mdl


QA
Pos
QB
QRad


1
Pos
-K-
Gain
mod
Math
Function
1
Constant
q2.mat
From File
x
xdat
ydat
y
Lookup
Table
Dynamic
1
QA
x
xdat
ydat
y
Lookup
Table
Dynamic1
2
QB

PWM.mdl

The *Simulink* S-function implementation of a parameterized pulse-width modulated source is below. The function requires specification of the PWM output frequency, the PWM scaling factor (the maximum input value that will result in full duty cycle output), and the name of the *Simulink* model that will call the function. The implementation provides a scalar to PWM function with a fixed frequency output of amplitude 1 and variable duty cycle.

```
function PWM(block)
%PWM
%   A MATLAB S-function implementing a PWM output based on a scalar input.
%   The block requires a comma separated list of three parameters:
%       1) The PWM output period
%       2) The maximum value of the input for full range PWM output
%       3) The name of the Simulink mdl file that calls the block, as a
%          string.
%   The maximum simulation step time of the simulink model must be much
%   smaller than the output period specified as a parameter. The
%   relationship of the step size to the PWM output period can be thought
%   of in a similar manner as the bit resolution specifications of a PWM
%   signal.
%
%   $Revision: 1.0 $
%
% The setup method is used to setup the basic attributes of the
% S-function such as ports, parameters, etc.
%
setup(block);

%endfunction
```

```
%% Function: setup ====================================================
% Abstract:
%   Set up the S-function block's basic characteristics such as:
%   - Input ports
%   - Output ports
%   - Dialog parameters
%   - Options
%
function setup(block)

% Register number of ports
block.NumInputPorts  = 1;
block.NumOutputPorts = 1;

% Setup port properties to be dynamic (inherited) or defaults
block.SetPreCompInpPortInfoToDynamic;
block.SetPreCompOutPortInfoToDynamic;
% Override input port properties
%block.InputPort(1).Dimensions        = 1;   %Removed for inheritance
%block.InputPort(1).DatatypeID  = 0;  % double
%block.InputPort(1).Complexity  = 'Real';
%block.InputPort(1).DirectFeedthrough = true;
block.InputPort(1).DirectFeedthrough = 1;
block.InputPort(1).SamplingMode = 'Sample';

% Override output port properties
block.OutputPort(1).Dimensions  = 1;
block.OutputPort(1).DatatypeID  = 0; % double
block.OutputPort(1).Complexity  = 'Real';
block.OutputPort(1).SamplingMode = 'Sample';

%Additional Outputs Used for Troubleshooting
%block.OutputPort(2).Dimensions  = 1;
%block.OutputPort(2).DatatypeID  = 0; % double
%block.OutputPort(2).Complexity  = 'Real';
%block.OutputPort(2).SamplingMode = 'Sample';

%block.OutputPort(3).Dimensions  = 1;
%block.OutputPort(3).DatatypeID  = 0; % double
%block.OutputPort(3).Complexity  = 'Real';
%block.OutputPort(3).SamplingMode = 'Sample';

%block.OutputPort(4).Dimensions  = 1;
%block.OutputPort(4).DatatypeID  = 0; % double
%block.OutputPort(4).Complexity  = 'Real';
%block.OutputPort(4).SamplingMode = 'Sample';

% Register parameters
block.NumDialogPrms = 3;
block.DialogPrmsTunable = {'Tunable','Tunable','Tunable'};
```

```
% Register sample times
%  [0 offset]            : Continuous sample time
%  [positive_num offset] : Discrete sample time
%
%  [-1, 0]               : Inherited sample time
%  [-2, 0]               : Variable sample time
block.SampleTimes = [-1 0];

% Specify the block simStateCompliance. The allowed values are:
%    'UnknownSimState', < The default setting; warn and assume DefaultSimState
%    'DefaultSimState', < Same sim state as a built-in block
%    'HasNoSimState',   < No sim state
%    'CustomSimState',  < Has GetSimState and SetSimState methods
%   'DisallowSimState' < Error out when saving or restoring the model sim state
block.SimStateCompliance = 'DefaultSimState';

% -----------------------------------------------------------------
% The MATLAB S-function uses an internal registry for all
% block methods. You should register all relevant methods
% (optional and required) as illustrated below. You may choose
% any suitable name for the methods and implement these methods
% as local functions within the same file. See comments
% provided for each function for more information.
% -----------------------------------------------------------------

block.RegBlockMethod('CheckParameters', @CheckPrms);
block.RegBlockMethod('ProcessParameters', @ProcessPrms);
block.RegBlockMethod('InitializeConditions', @InitializeConditions);
block.RegBlockMethod('Outputs', @Outputs);     % Required
block.RegBlockMethod('Terminate', @Terminate); % Required

%end setup

%% CheckParameters:
  %   Functionality    : Called in order to allow validation of the
  %                      block dialog parameters. You are
  %                      responsible for calling this method
  %                      explicitly at the start of the setup method.
  %   C-Mex counterpart: mdlCheckParameters
  %
function CheckPrms(block)

tPeriod = block.DialogPrm(1).Data;
PWMRange = block.DialogPrm(2).Data;
modelName = block.DialogPrm(3).Data;

if (tPeriod <= 0)
    error('The first dialog parameter must be a positive real number.')
end
```

```
if (PWMRange <= 0)
    error('The second dialog parameter must be a positive real number.')
end

if ~isstr(modelName)
    error('The second dialog parameter must be a positive real number.')
end

%endfunction

%% ProcessParameters:
  %   Functionality    : Call to allow an update of run-time parameters.
  %   C-Mex counterpart: mdlProcessParameters
  %
function ProcessPrms(block)
  block.AutoUpdateRuntimePrms;

%endfunction

%% InitializeConditions:
%   Functionality    : Called at the start of simulation and if it is
%                      present in an enabled subsystem configured to reset
%                      states, it will be called when the enabled subsystem
%                      restarts execution to reset the states.
%   Required         : No
%   C-MEX counterpart: mdlInitializeConditions
%
function InitializeConditions(block)
block.OutputPort(1).Data = 0.0;

%end InitializeConditions

%% Outputs:
%   Functionality    : Called to generate block outputs in
%                      simulation step
%   Required         : Yes
%   C-MEX counterpart: mdlOutputs
%
function Outputs(block)
tPeriod = block.DialogPrm(1).Data;
PWMRange = block.DialogPrm(2).Data;
modelName = block.DialogPrm(3).Data;
pulseWidth = min(block.InputPort(1).Data/PWMRange,1.0);
currTime = get_param(modelName,'SimulationTime');
relTime = mod(currTime,tPeriod)/tPeriod;
```

```
if (relTime <= pulseWidth)
    block.OutputPort(1).Data = 1;
else
    block.OutputPort(1).Data = 0;
end

%block.OutputPort(2).Data = currTime;
%block.OutputPort(3).Data = pulseWidth;
%block.OutputPort(4).Data = relTime;

%end Outputs

%% Terminate:
%   Functionality    : Called at the end of simulation for cleanup
%   Required         : Yes
%   C-MEX counterpart: mdlTerminate
%
function Terminate(block)

%end Terminate
```

QDecoder.mdl

The *Simulink* S-function implementation of a quadrature to analog position decoder that resolves the direction of motion based on which input is leading and computes the numeric position in radians from two inputs where 360 pulses represent one revolution is below.

```
function QDecode(block)
%QDecode
%   A MATLAB S-function implementing a decoder of the quadrature output
%   from a 360 pulses per revolution optical encoder. From the quadrature
%   signal inputs, position data in radians is output.
%
%   $Revision: 1.0 $
%
% The setup method is used to setup the basic attributes of the
% S-function such as ports, parameters, etc.
%
setup(block);

%endfunction

%% Function: setup ====================================================
% Abstract:
%   Set up the S-function block's basic characteristics such as:
%   - Input ports
%   - Output ports
%   - Dialog parameters
%   - Options
%
function setup(block)

% Register number of ports
block.NumInputPorts  = 2;
block.NumOutputPorts = 1;

% Setup port properties to be dynamic (inherited) or defaults
block.SetPreCompInpPortInfoToDynamic;
block.SetPreCompOutPortInfoToDynamic;

% Override input port properties
%block.InputPort(1).Dimensions        = 1;      %Removed for inheritance
%block.InputPort(1).DatatypeID  = 0;  % double
%block.InputPort(1).Complexity  = 'Real';
%block.InputPort(1).DirectFeedthrough = true;
block.InputPort(1).DirectFeedthrough = 1;
block.InputPort(2).DirectFeedthrough = 1;
```

```
block.InputPort(1).SamplingMode = 'Sample';
block.InputPort(2).SamplingMode = 'Sample';

% Override output port properties
block.OutputPort(1).Dimensions  = 1;
block.OutputPort(1).DatatypeID  = 0; % double
block.OutputPort(1).Complexity  = 'Real';
block.OutputPort(1).SamplingMode = 'Sample';

% Register parameters
block.NumDialogPrms = 0;

% Register sample times
%  [0 offset]            : Continuous sample time
%  [positive_num offset] : Discrete sample time
%
%  [-1, 0]               : Inherited sample time
%  [-2, 0]               : Variable sample time
block.SampleTimes = [-1 0];

% Specify the block simStateCompliance. The allowed values are:
%    'UnknownSimState', < The default setting; warn and assume
%    DefaultSimState
%    'DefaultSimState', < Same sim state as a built-in block
%    'HasNoSimState',   < No sim state
%    'CustomSimState',  < Has GetSimState and SetSimState methods
%    'DisallowSimState' < Error out when saving or restoring the model sim
%    state
block.SimStateCompliance = 'DefaultSimState';

% -----------------------------------------------------------------
% The MATLAB S-function uses an internal registry for all
% block methods. You should register all relevant methods
% (optional and required) as illustrated below. You may choose
% any suitable name for the methods and implement these methods
% as local functions within the same file. See comments
% provided for each function for more information.
% -----------------------------------------------------------------

block.RegBlockMethod('PostPropagationSetup', @DoPostPropSetup);
block.RegBlockMethod('Start', @Start);
block.RegBlockMethod('Outputs', @Outputs);     % Required
block.RegBlockMethod('Update', @Update);
%block.RegBlockMethod('Derivatives', @Derivatives);
block.RegBlockMethod('Terminate', @Terminate); % Required

%end setup
```

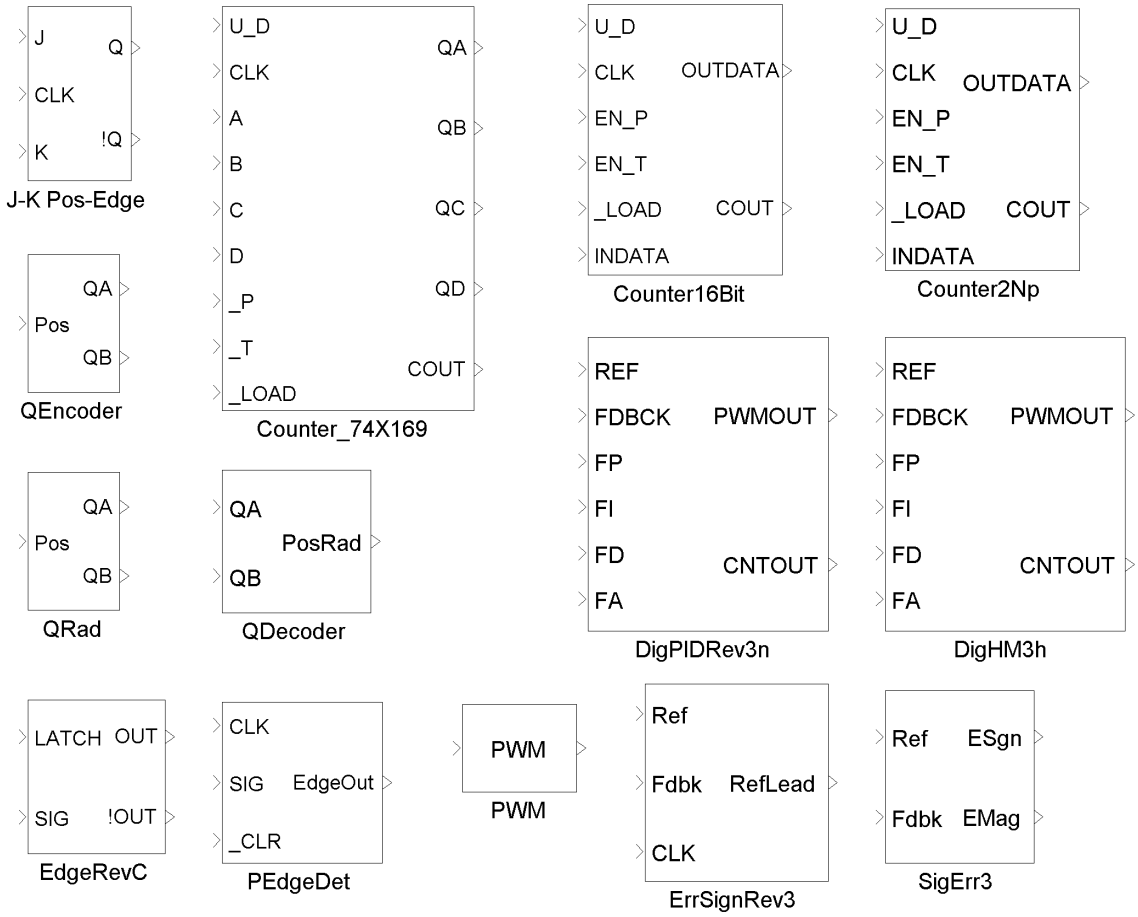
```
%% PostPropagationSetup:
%   Functionality    : Setup work areas and state variables. Can
%                      also register run-time methods here
%   Required         : No
%   C-Mex counterpart: mdlSetWorkWidths
%
function DoPostPropSetup(block)
block.NumDworks = 3;

  block.Dwork(1).Name            = 'Current';
  block.Dwork(1).Dimensions      = 1;
  block.Dwork(1).DatatypeID      = 0;      % double
  block.Dwork(1).Complexity      = 'Real'; % real
  block.Dwork(1).UsedAsDiscState = true;

  block.Dwork(2).Name            = 'Previous';
  block.Dwork(2).Dimensions      = 1;
  block.Dwork(2).DatatypeID      = 0;      % double
  block.Dwork(2).Complexity      = 'Real'; % real
  block.Dwork(2).UsedAsDiscState = true;

  block.Dwork(3).Name            = 'Out';
  block.Dwork(3).Dimensions      = 1;
  block.Dwork(3).DatatypeID      = 0;      % double
  block.Dwork(3).Complexity      = 'Real'; % real
  block.Dwork(3).UsedAsDiscState = true;

%end DoPostPropSetup

%% Start:
%   Functionality    : Called once at start of model execution. If you
%                      have states that should be initialized once, this
%                      is the place to do it.
%   Required         : No
%   C-MEX counterpart: mdlStart
%
function Start(block)

block.Dwork(1).Data = 4;
block.Dwork(2).Data = 4;
block.Dwork(3).Data = 0;

%end Start
```

```
%% Outputs:
%   Functionality    : Called to generate block outputs in
%                      simulation step
%   Required         : Yes
%   C-MEX counterpart: mdlOutputs
%
function Outputs(block)
% Inputs
% 1 - Q1
% 2 - Q2
%
change = 0.0;
P = 3.142;

    if ((block.InputPort(1).Data == 1) && (block.InputPort(2).Data == 0))
        block.Dwork(2).Data = block.Dwork(1).Data;
        block.Dwork(1).Data = 1;
    elseif ((block.InputPort(1).Data == 1) && (block.InputPort(2).Data == 1))
            block.Dwork(2).Data = block.Dwork(1).Data;
            block.Dwork(1).Data = 2;
    elseif ((block.InputPort(1).Data == 0) && (block.InputPort(2).Data == 1))
            block.Dwork(2).Data = block.Dwork(1).Data;
            block.Dwork(1).Data = 3;
    elseif ((block.InputPort(1).Data == 0) && (block.InputPort(2).Data == 0))
            block.Dwork(2).Data = block.Dwork(1).Data;
            block.Dwork(1).Data = 4;
    end

    if block.Dwork(1).Data ~= block.Dwork(2).Data
        switch block.Dwork(1).Data
            case 1
                if block.Dwork(2).Data == 4
                    change = P/720;
                elseif block.Dwork(2).Data == 2
                    change = -P/720;
                end
            case 2
                if block.Dwork(2).Data == 1
                    change = P/720;
                elseif block.Dwork(2).Data == 3
                    change = -P/720;
                end
            case 3
                if block.Dwork(2).Data == 2
                    change = P/720;
                elseif block.Dwork(2).Data == 4
                    change = -P/720;
                end
```

```
            case 4
                if block.Dwork(2).Data == 3
                    change = P/720;
                elseif block.Dwork(2).Data == 1
                    change = -P/720;
                end
        end
        block.OutputPort(1).Data = block.Dwork(3).Data + change;
    else
        block.OutputPort(1).Data = block.Dwork(3).Data;
    end

%end Outputs

%% Update:
%   Functionality    : Called to update discrete states
%                      during simulation step
%   Required         : No
%   C-MEX counterpart: mdlUpdate
%
function Update(block)
block.Dwork(3).Data = block.OutputPort(1).Data;

%end Update

%% Terminate:
%   Functionality    : Called at the end of simulation for cleanup
%   Required         : Yes
%   C-MEX counterpart: mdlTerminate
%
function Terminate(block)

%end Terminate
```

APPENDIX F

A library file that contains the core functional components described and developed in this work for use in creating future systems and simulations is comprised of the *Simulink* model and *MATLAB* m-file presented in this appendix.



```
function blkStruct = slblocks
  % Creates Simulink library of all-digital components. Useful for
  % model re-use, and version control.

  % Must modify the MATLAB path to include the specific subdirectory
  % where this file resides.

  % Specify that the product should appear in the library browser
  % and be cached in its repository

  % blkStruct.Name = ['Feinauer' sprintf('\n') 'Dissertation'];
  % blkStruct.OpenFcn = 'FeinDiss';
  % blkStruct.MaskDisplay = '';

  Browser.Library = 'FeinDiss';
  Browser.Name    = 'Feinauer Dissertation';
  blkStruct.Browser = Browser;
  clear Browser;
```

REFERENCES

[1] V.J. VanDoren, "PID: still the one," *Control Engineering*, vol. 50, no. 10, pp. 32-37, Oct. 2003.

[2] S. Bennet, "The past of PID controllers," *Annual Reviews in Control*, vol. 25, pp. 43-53, 2001.

[3] K.J. Aström and T. Hägglund, "Auto-tuners for PID controllers," in *The Impact of Control Technology*, T. Samad and A.M. Annaswamy (eds.), IEEE Control Systems Society, 2011. [Online]. Available: http://ieeecss.org/general/impact-control-technology

[4] G. Stewart and T. Samad, "Cross-application perspectives: application and market requirements," in *The Impact of Control Technology*, T. Samad and A.M. Annaswamy (eds.), IEEE Control Systems Society, 2011. [Online]. Available: http://ieeecss.org/general/impact-control-technology

[5] K.H. Ang and G. Chong, "PID control system analysis, design and technology," *IEEE Trans. On Control Systems Tech.*, vol. 13, no. 4, pp. 559-576, July 2005.

[6] J.G. Ziegler and N.B. Nichols, "Optimum settings for automatic controllers," *Trans. ASME*, vol. 64, pp. 759-768, 1942.

[7] V.J. VanDoren, "Auto-tuning control using Ziegler-Nichols," *Control Engineering*, vol. 53, no. 10, pp. 66-70, Oct. 2006.

[8] J.C. Basilio and S.R. Matos, "Design of PI and PID controllers with transient performance, specification, *IEEE Trans. On Education*, vol. 45, no. 4, pp. 364-370, Nov. 2002.

[9] N.S. Nise, *Control Systems Engineering*, 4th ed. Hoboken, NJ: John Wiley & Sons, 2004, pp. 87-93, 329-330, 635-645.

[10] R.C. Dorf and R.H. Bishop, *Modern Control Systems*, 9th ed. Upper Saddle River, NJ: Prentice Hall, 2001, pp. 52-56, 295-303.

[11] M.A. Hersh and M.A. Johnson, "A study of advanced control systems in the work place," *Control Engineering Practice*, vol. 5, no. 6, pp. 771-778, 1997.

[12] H. Takatsu and T. Itoh, "Future needs for control theory in industry—report of the control technology survey in Japanese industry," *IEEE Trans. Control Systems Technology*, vol. 7, no. 3, pp. 298-305, May 1999.

[13] A. O'Dwyer, "PI and PID controller tuning rules: an overview and personal perspective," in *The IET Irish Signals and Systems Conference*, Dublin, Jun. 28-30, 2006, pp. 161-166.

[14] K. Butts and A. Varga, "Cross-application perspectives: tools and platforms for control systems," in *The Impact of Control Technology*, T. Samad and A.M. Annaswamy (eds.), IEEE Control Systems Society, 2011. [Online]. Available: http://ieeecss.org/general/impact-control-technology

[15] Bespoke chips for the common man. (2002, Dec. 14). *The Economist*, no. 8303, pp. 31-32.

[16] K. Parnell, "Could microprocessor obsolescence be history?" *Xcell Journal*, no. 45, pp. 21-23, Spring 2003.

[17] R. Dubey, P. Agarwal and M.K. Vasantha , "Programmable logic devices for motion control—a review," *IEEE Trans. on Industrial Electronics*, vol. 54, no. 1, pp. 559-566, Feb. 2007.

[18] E. Monmasson and M.N. Cirstea, "FPGA design methodology for industrial control systems—a review," *IEEE Trans. Industrial Electronics*, vol. 54, no. 4, pp. 1824-1842, Aug. 2007.

[19] A. Aounis, S.E. Cirstea and M.N. Cirstea, "Reusable VHDL architectures for induction motor PWM vector control, targeting FPGAs," in *32nd Annual Conference on IEEE Industrial Electronics*, 2006, pp. 4923-4928.

[20] H.J. Guo, M. Takahashi. T. Watanabe and O. Ichinokura, "A new sensorless drive method of switched reluctance motors based on motor's magnetic characteristics," *IEEE Tran. on Magnetics*, vol. 37, no. 4, pp. 2831-2833, Jul. 2001.

[21] Y-S. Kung, C-S. Chen, K-I. Wong, and M-H. Tsai, "Development of a FPGA-based control IC for PMSM drive with adaptive fuzzy control," *31st Annual Conference of IEEE Industrial Electronics Society*, 2005, pp. 1544-1549.

[22] Y.S. Sun and Y. Fang, "Field programmable gate array (FPGA) based embedded system design for AFM real-time control," in *2010 IEEE International Conf. on Control Applications*, Yokohama, Japan, pp. 245-250.

[23] K.K. Tan and A.S. Putra, "Trends in motion control," in *Drives and Control for Industrial Automation*, Advances in Industrial Control. London, England: Springer-Verlag, 2011, ch. 7, pp. 163-170.

[24] A.W. Moore, "Phase-locked loops for motor-speed control," *IEEE Spectrum*, vol. 10, no. 4, pp. 61-67, Apr. 1973.

[25] G-C. Hsich and J.C. Hung, "Phase-locked loop techniques—a survey," *IEEE Trans. on Industrial Electronics*, vol. 43, no. 6, Dec. 1996.

[26] C.A. Adkins, M.A. Marra and B.L. Walcott, "Modified phase-frequency detector for improved response of PLL servo-controller," in *Proc. of the American Control Conference*, Anchorage, AK, 2002, pp. 19-20.

[27] C.A. Adkins, T.J. Eade, M.A. Marra, III, B.L. Walcott, "Apparatus and method for electronic control of DC motor using an all-digital phase-locked loop," U.S. Patent 6 823 133 B1, Nov. 23, 2004.

[28] H.H. Chin, "All digital design and implementation of proportional-integral-derivative (PID) controller," M.S. Thesis, Dept. of Electrical and Computer Eng., Univ. of Kentucky, 2006.

[29] W.J. Fleming, "Overview of Automotive Sensors," *IEEE Sensors Journal*, vol. 1, no. 4, pp. 296-308, Dec. 2001.

[30] National Semiconductor Corp. (June 1989). 54LS169 DM54LS169A DM74LS169A Synchronous 4-Bit Up/Down Binary Counter. National Semiconductor, Santa Clara, CA. [Online]. Available: http://www.national.com/ds/54/54LS169.pdf

[31] G.J. Murphy and S.H. Wu, "A stability criterion for pulse-width-modulated feedback control systems," *IEEE Trans. Automatic Control*, vol. AC-9, pp. 434-441, Oct. 1964.

[32] R.A. Skoog, G.L. Blankenship, "Generalized pulse-modulated feedback system: norms, gains, Lipschitz constants, and stability," *IEEE Trans. Automatic Control*, vol. AC-15, no. 3, June 1970.

[33] W. Zhang and W. Zhan, "Sensitivity analysis of motor PWM control," in *Proc. Of the World Congress on Engineering and Computer Science*, San Francisco, CA, 2008.

[34] W. Zhan, "Robust design of motor PWM control using modeling and simulation," in *Advances in Computational Algorithms and Data Analysis*, vol. 14, *Lecture Notes in Electrical Engineering*, S-I. Ao, B. Rieger and S-S. Chen Eds. Netherlands: Springer, 2009, pp. 439-449.

[35] A. Eisinberg, G. Fedele and D. Frascino, "Kepler's equation and limit cycles in a class of PWM feedback control systems," *Nonlinear Dynamics*, vol. 62, no. 1, pp. 215-227, Oct. 2010.

VITA

David Michael Feinauer

Date of Birth: May 1981

State of Birth: Kentucky

EDUCATIONAL INSTITUTIONS

University of Kentucky

Bachelors of Science Degree, 2003

Major: Electrical Engineering

Minors: Mathematics and Computer Science

PROFESSIONAL POSITIONS

Instructor, College of Engineering, University of Kentucky, Lexington, Kentucky *(Fall 2011, Fall 2010, Fall 2009*)

Graduate Research Assistant, Department of Electrical and Computer Engineering, University of Kentucky, Lexington, Kentucky *(Fall 2006 – Fall 2010*)

Electrical Hardware Engineer, Lexmark International, Inc., Lexington, Kentucky *(July 2005 – December 2006*)

Co-Director, UK Engineering Summer Program, College of Engineering, University of Kentucky, Lexington, Kentucky *(Summers of 2003 – 2007, Summer 2009*)

Teaching Assistant, College of Engineering, University of Kentucky, Lexington, Kentucky *(Fall 2003)*

Undergraduate Research Assistant, Department of Electrical and Computer Engineering, ONR STTR Grant, University of Kentucky, Lexington, Ky *(Fall 2002 – Spring 2003*)

Resident Advisor, Office of Residence Life, University of Kentucky, Lexington, Kentucky *(Fall 2001 – Spring 2003*)

Engineering and Business Processes Intern, Holland Metals Ltd. and RamKat Enterprises LLC., Dayton, Kentucky (*Summers of 1999 – 2003*)

Undergraduate Research Assistant, Dr. Kevin Donohue, Lexington, Ky (*Spring 2001*)

SCHOLASTIC AND PROFESSIONAL HONORS

Named Inventor, US Patent Number 7,593,653 B2

Best Conference Paper, ASEE Zone II, 2007

First Place, Best Paper Award, 2007 ASEE North Central Section Conference

Lexmark Fellowship 2005 – 2006

President's Fellowship 2004 – 2005

Charles T. Wethington Fellowship 2003 – 2004

Second place, Student Software Competition, IEEE SoutheastCON Conference, 2003

Outstanding Senior in Electrical and Computer Engineering, University of Ky, 2003

Outstanding Junior in Electrical and Computer Engineering, University of Ky, 2002

Eta Kappa **N**u Outstanding Junior in Electrical Engineering, University of Ky, 2002

General Chemistry Excellence Award, University of Kentucky, 2000

Second Place, UK IEEE Student Paper Contest, 2000

JCPenney / United Way Golden Rule Community Service Award, 1999

PROFESSIONAL PUBLICATIONS

[1] O. Rawashdeh, D. Feinauer, C. Harr, G. Chandler, D. Jackson, A. Groves, and J. Lumpp, "A Dynamically Reconfiguring Avionics Architecture for UAVs," *AIAA Infotech@Aerospace Conference*, Arlington, Virginia, September 26-29, 2005. AIAA-2005-7050.

[2] G. Chandler, C. Harr, O. Rawashdeh, D. Feinauer, D. Jackson, A. Groves, and J. Lumpp, "Wireless Extension of an Avionics Bus for Prototyping and Testing Reconfigurable UAVs," *AIAA Infotech@Aerospace Conference*, Arlington, Virginia, September 26-29, 2005. AIAA-2005-7151.

[3] D. Feinauer, B. Walcott, "Lessons Learned From a Summer Engineering Outreach Program," *Proceedings of the Spring 2007 American Society for Engineering Education North Central Section Conference at West Virginia Institute of Technology (WVUTech)*, Charleston, West Virginia, March 30-31, 2007.

[4] D. Feinauer, B. Walcott, "Immersing High School Students in Engineering and Entrepreneurship," *ASEE Annual Conference and Exposition*, Honolulu, Hawaii, June 24-27, 2007. AC 2007-1226.

[5] E. Seagraves, B. Walcott, D. Feinauer, "Equalization of Volterra Systems Using a Multilinear SVD Based Pseudo p^{th} Order Inverse," *50th IEEE International Midwest Symposium on Circuits and Systems, MWSCAS 2007*, August 5-8, 2007, Montreal, QC, Canada, 10.1109/MWSCAS.2007.

[6] E. Seagraves, B. Walcott, D. Feinauer, "Efficient Implementation of Volterra Systems Using a Multilinear SVD," *International Symposium on Intelligent Signal Processing and Communication Systems, ISPACS 2007,* November 28 - December 1, 2007, pp. 762-765. 10.1109/ISPACS.2007.4445999.

[7] D. Feinauer, D. Hewett, B. Walcott, "The KNEED Program: A Novel Model for Small High-Tech Business and University Cooperation," *Proceedings of the 2008 ASEE Southeast Section Conference,* Memphis, Tennessee, April 6-8, 2008.

PROFESSIONAL PRESENTATIONS

[1] "Lessons Learned From a Summer Engineering Outreach Program," *ASEE North Central Section Conference,* Charleston, West Virginia, March 30, 2007.

[2] "Immersing High School Students in Engineering and Entrepreneurship," *ASEE Annual Conference and Exposition*, Honolulu, Hawaii, June 25, 2007.

[3] "Ethical Behavior and Responsibility – Response Moderator Report," *2007 National Conference on Graduate Student Leadership*, Lexington, Kentucky, November 11, 2007.

[4] "The KNEED Program: A Novel Model for Small High-Tech Business and University Cooperation," *ASEE Southeast Section Conference,* Memphis, Tennessee, April 7, 2008.

[5] "Lessons Learned From a Summer Engineering Outreach Program," *ASEE Annual Conference and Exposition* (Invited Presentation), Pittsburgh, Pennsylvania, June 24, 2008.

[6] "E-STEM: A Panel Discussion of Entrepreneurialism and the Role of Enterprising Thought in Science, Technology, Engineering, and Mathematics Education," *First Annual STEM Education Conference*, University of Kentucky, Lexington, Kentucky, February 19, 2010.