AN ABSTRACT OF THE THESIS OF

_Dongtai_Liu__ for the degree of __Master__of__Science___ in _Electrical_and_Computer_Engineering_ presented on March_31, 1987_. Title:___A__Hierarchically_Organized_Microprocessor_System_for_Motor_Position_Control_____

Redacted for Privacy

Abstract approved:_____
James H. Herzog

Accurate Position Control is an important technique in robotic engineering. The working environment of robotic systems presents special requirements for position control. The scope of this study is to explore the control algorithms of a digital, hierarchical position control system. Detailed analysis and simulation show that the system can be compensated by linear functions to obtain the required characteristics. Various nonlinear approaches are also studied. Experiments show that with the nonlinear dynamic compensation (NDC) the response speed can be optimized and problems such as quantization noise and nonlinearity of the actuator can be better solved.

In order to test the design a high-level language digital simulator is developed. Real-time experiment is also conducted with a microprocessor controller. Experimental data from both the simulation and the real-time experiment demonstrate that the control algorithms are successful.

The experimental system is also hierarchical. The human interface is accomplished by a personal computer which receives the operator's command and reports the system status. The communication between the PC and the controller is managed by control-oriented operating system. The actual control activities are handled by the controller task routine in the operating system.

A Hierarchically Organized Microprocessor System for Motor Position Control

by

Dongtai Liu

A THESIS

submitted to

Oregon State University

in partial fulfilment of
the requirements for the
degree of

Master of Science

Completed March 23, 1987

Commencement June 1987

APPROVED:

Redacted for Privacy

Associate professor of E&CE in charge of major

Redacted for Privacy

Head of department of Electrical & Computer Engineering

Redacted for Privacy

Dean of Graduate School

Date thesis is presented March 31, 1987

ACKNOWLEDGEMENTS

I would like to extend my sincere thanks to Dr. James H. Herzog, my major professor, for his encouragement and guidance throughout this study. I appreciate the help of Professors L. C. Jensen, G. C. Alexander, A. K. Wallace, D. L. Amort and E. Fitcher who kindly provided parts, materials and information.

I warmly thank all of my friends and colleagues whose assistance, encouragements have been invaluable.

My wife, Yaqin Zhou, and my parents deserve special thanks for all of the support and encouragement provided during the time of the work.

TABLE OF CONTENTS

LIST OF FIGURES

A HIERARCHICALLY ORGANIZED MICROPROCESSOR SYSTEM FOR MOTOR POSITION CONTROL

1. INTRODUCTION

Accurate position control is an important technique in control engineering. Many working environments require that the position response be smooth and fast; overshoot must be minimized. The system is also required to be relatively insensitive to the variation of the load inertia which could change drastically with the motion and mass of the load. Typical position control applications cover a wide variety from mechanical arm control to satellite launch. The primary interest of this study will be focussed on systems using motors as actuators.

Stepping motors and d.c. servo motors are most frequently used actuating devices in position control systems. Accordingly there are two kinds of control mechanisms. A system employing a stepping motor is basically an open-loop system. Because the stepping motor has the build-in position following capability, the control scheme in such a system is very simple provided the torque capability of the motor is not exceeded. The stepping motor system is most suitably used where there is little mechanical load and external mechanical inertia. At large torque or power failure the controller can lose track of

position because there is no feedback.

A position control system using a d.c. servo motor differs from the stepping motor system in that the system is a closed-loop system. The controller is very much involved in the control activities. Since a position sensor is used in such a system, it is possible to achieve high accuracy and reliability. Another advantage of the d.c. motor system is that it can provide much greater mechanical power. Because of these features, d.c. motor systems are commonly used in robot arms. However, the control algorithms of such a system are more complicated.

The controller of a position control system could be a simple analog device, or a digital programmable device. Since a digital programmable controller (often a microprocessor) provides flexible control algorithms and easy communication with its host system (often a computer), it is more desirable in a robotic system.

The goal of this study is to explore the control algorithms of a microprocessor position control system. The research work is divided into four steps: the system modeling, the system simulation, the non-linear control experiment with the simulator and the real-time experiment.

The first step, described by Chapter 2, is the linear analysis of the position control system in which mathematical models of the system components are

established and the system parameters are related to the stability and performance of the system.

Chapter 3 describes the digital simulation of the position control system. A very useful design tool, the position control system simulator, is developed. The simulated system response curves are presented.

In Chapter 4, constraints of the linear control that limit the performance of the system are illustrated. Non-linear dynamic compensations for optimal control and further improvement are proposed and tested using the simulator.

Chapter 5 describes the real-time digital position control experiment using the system model and control algorithms developed in Chapter 2 thru 4. The system is organized in a hierarchical fashion. The Intel 8039 Micro-processor is used to implement the controller. An IBM PC is used as the host computer which sends position commands to and receives real-time position data from the microprocessor system. The various response curves recorded from the real-time experiment are presented and compared with the simulation curves.

2. SYSTEM MODELING AND CONTROLLER DESIGN

The position control system we are considering is a feedback control system. As in other control systems, there are delay links. It is these delays that limit the performance of the system. The number of the delay links is the order of the system. When the order of system is high the design can become complicated. It is necessary to specify the relevant conditions and to reach an acceptable compromise in performance indices. Therefore, in designing a high order system like the position control system, it is necessary to first understand the characteristics of the system.

In this chapter, mathematical models for the system components will be established and the root locus method will be used in the linear analysis of the system. A linear controller will then be specified.

2.1 Mathematical Models for the Position Control System

A position control system using a d.c. servo motor has 3 major components: a d.c. motor, a position transducer and a controller. It is most convenient to use the frequency-domain models or transfer functions to describe these system components. Linear analysis of the system can be done easily by using these models.

2.1.1 Transfer Function of the Motor

The transfer function of a d.c. motor can be derived from the structure diagram of the motor (Figure 2.1):

$$\frac{N(S)}{U(S)} = \frac{1/Ke}{TmTeS^2 + TmS + 1}$$

where U(S) is the Laplace transform of the motor's armature voltage, N(S) is the Laplace transform of the motor's rotary speed (revolutions per minute or rpm). Tm, Te and Ke are motor parameters. The definitions of these parameters can be found in Appendix I.

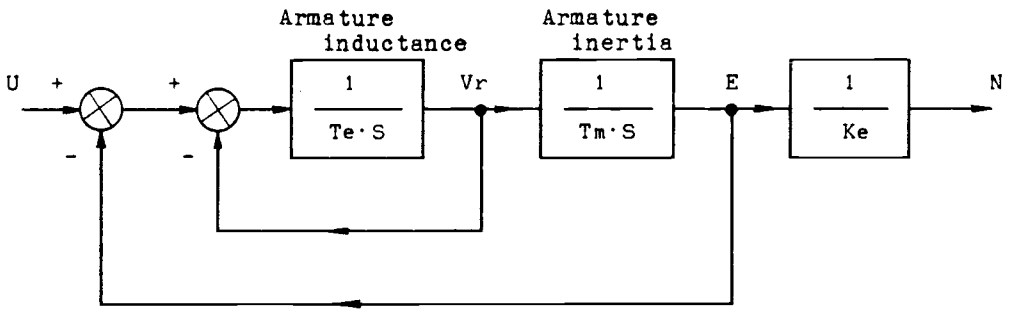


Figure 2-1. Structure of d.c. Motor

For most d.c. servo motors the inequality Tm>4Te holds. Therefore the transfer function of the motor can be written as

$$\frac{N(S)}{U(S)} = \frac{Km}{(S+A)(S+B)}$$

where A and B are the two real roots of the denominator polynomial and Km=1/KeTmTe. Root A corresponds to the motor's mechanical time constant Tm, hence, it is the dominant pole of the motor's transfer function.

2.1.2 Transfer Function of the Position Transducer

Since position is the time integral of the motor speed, the transfer function of the position transducer is

$$\frac{P(S)}{N(S)} = \frac{Kt}{S} \qquad (2\text{-}1)$$

where Kt is the transfer ratio or transducer gain.

2.1.3 Transfer Functions of the Controller

The task of the controller is to detect the position error and generate the control output. Usually the controller not only responds to the error itself but also to its derivative and/or integral.

Controllers are generally categorized into three types, according to the functions they perform: the P type, the PD type and the PID type. A P type controller generates an output proportional to the error. A PD type controller responds to both the error and its derivative. A PID

controller uses the error, its integral and its derivative to generate the control output. Figure 2-2 lists the transfer functions of these controllers.

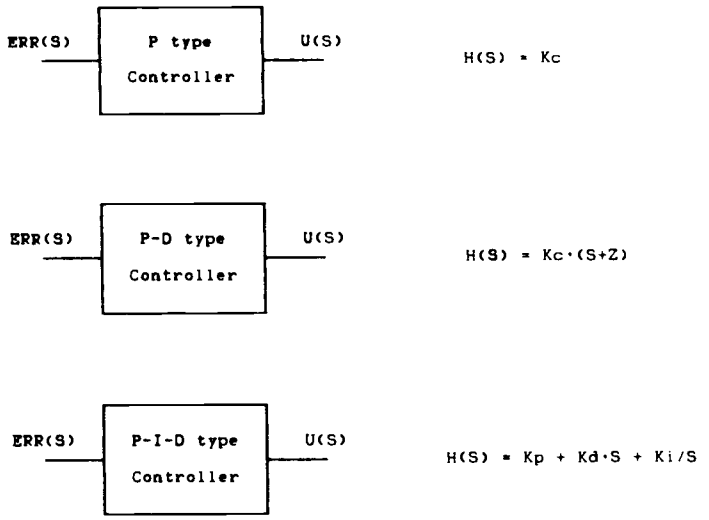


Figure 2-2. Frequently Used Linear Controllers

2.2 Linear Analysis of the System

The models developed in the previous sections can be put together to form the model of the position control system, as shown by Figure 2-3. The system's closed-loop characteristics can thus be predicted using the root-locus techniques. The root-locus diagrams of the system using the three types of controllers are shown in Figure 2-4 thru Figure 2-6.

Figure 2-4(a) shows the closed-loop root-locus of the system using a P type controller. As can be seen from the

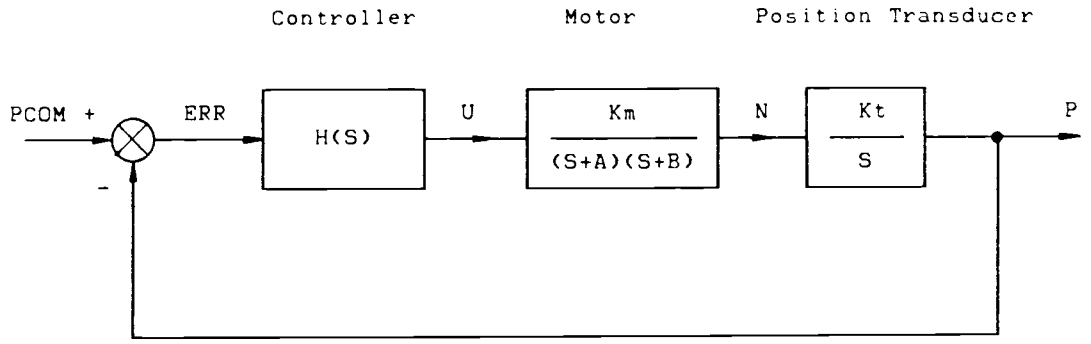
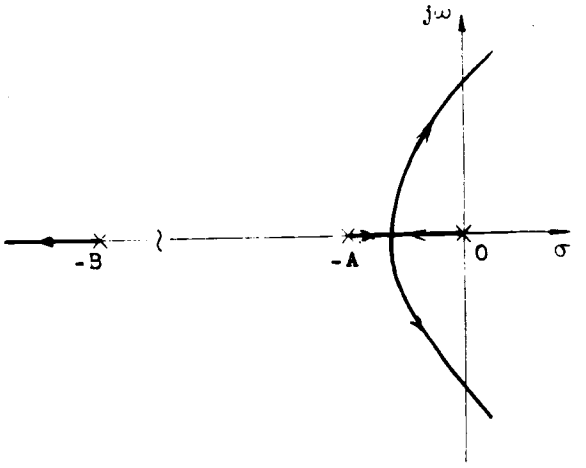


Figure 2-3. Block diagram of the system

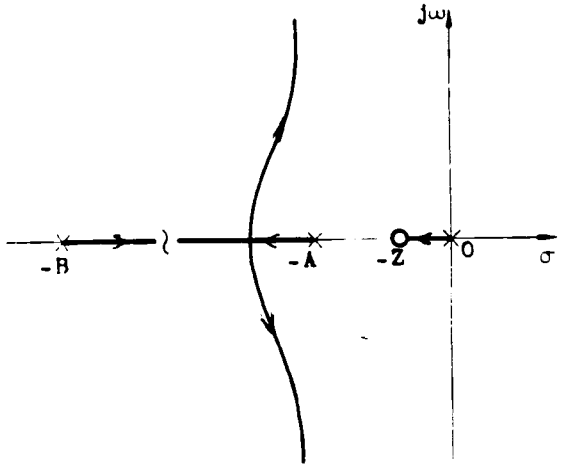
diagram, the locations of the two dominant closed-loop poles are affected by the location of the motor pole A. In other words, when the motor mechanical time constant changes, the system response will also change. This can cause the system to act in an undesirable manner when the load inertia changes. The system can also oscillate because the dominant roots can be complex.

Figure 2-4(b) shows the root locus of the system using a PD controller. If we choose the location of the zero such that it is on the negative real axis between the motor pole and the origin, the dominant closed-loop roots will always reside on the real axis, the system's step response takes the form of an exponential curve. There is no overshooting or oscillation. Because the dominant poles are located between the origin and the zero Z of the controller, the motor pole has very little influence on them. Therefore, the



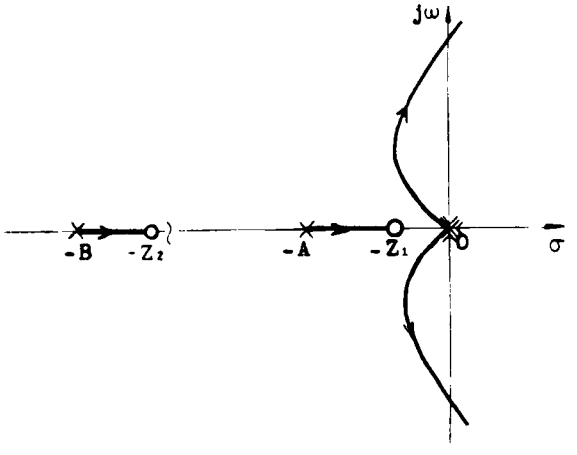
$$G(S) = \frac{K}{S \cdot (S+A) \cdot (S+B)}$$

(a)



$$G(S) = \frac{K \cdot (S+Z)}{S \cdot (S+A) \cdot (S+B)}$$

(b)



$$G(S) = \frac{K \cdot (S+Z_1) \cdot (S+Z_2)}{S^2 \cdot (S+A) \cdot (S+B)}$$

(c)

Figure 2-4. Root locus of the system (a) With P controller; (b) With P-D controller; (c) With P-I-D controller.

system behavior is not greatly affected by the change of the total motor inertia.

The root locus of the system using a PID controller is shown by Figure 2-4(c). The system becomes a 4th order system with two poles at the origin. The advantage of PID control over the PD and P control is that there is no steady-state error with constant torque output. However, because the dominant close-loop poles are always complex, the step response is slow and overshooting.

Generally, the PD and PID controllers are more preferable. The advantage of the PD controller is that the system is most stable. With properly chosen parameters the system has no overshoot, even at changing inertia. The defeciency of such a system is that there can be steady-state error when a constant load torque greater than the mechanical friction is applied. The magnitude of the error is proportional to the applied torque. The error analysis is given in Appendix II.

The PID controller, on the other hand, can overcome this steady-state error by integrating (or accumulating) small error to generate an output large enough to provide the torque. However, since the integration function brings another delay into the system, the compromise between the stability and response speed is more difficult. Such a system is often associated with overshoot.

It should be noted that there are other approaches to implement the controller's linear functions. The previously discussed control functions belong to "serial compensation" of the system. Another frequently used method is "parallel compensation" which could also perform the control task. Parallel approach uses more than one system state transducer (such as a speed transducer) and simpler controller function.

In this study the PD controller is chosen because it satisfies most of the requirements for robot control. Also, the relative simplicity of the control function is more suitable to be implemented with the available 8-bit single chip microprocessor.

Figure 2-5 shows the block diagram of the linear position control system with a PD controller. We will see in the later chapters that other functions including some non-

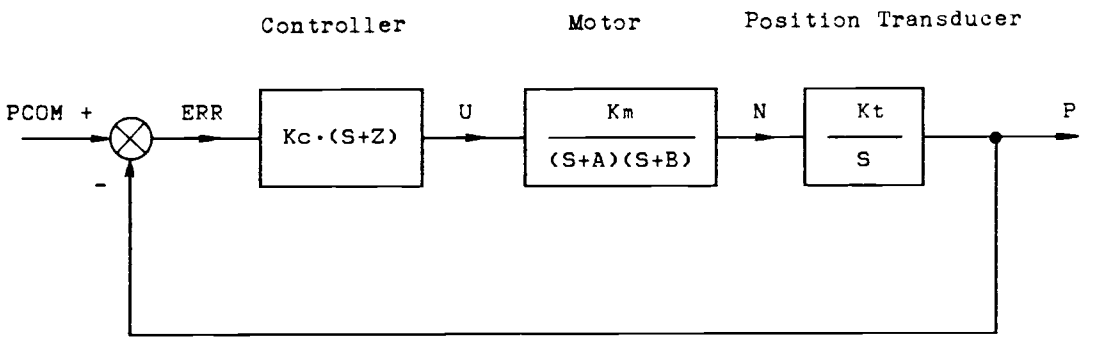


Figure 2-5. Block diagram of the system with P-D controller

linear functions will be added to the controller to solve the special problems which arise in the discrete, quantized system and to further improve the system's performance.

2.3 Summary

Position control systems are high-order systems. In order to design a position control system with desired characteristics, linear analysis is required.

The mathematical model of the system, required for the linear analysis, can be established using the transfer functions of the system components.

A linear analysis for the system shows that the PD and PID type controllers are appropriate candidates for the controller. The system using the PD controller has the maximum stability. There is no overshoot in the response. Such a system is also able to resist the inertia change. However, steady-state position error can occur if the constant external torque is greater than the friction. The system using the PID controller has no steady state error. The response is usually slow and overshooting.

3. DIGITAL SIMULATION OF THE SYSTEM

Although the analysis introduced in the last chapter is for the continuous or analog system, the resulting conclusions will still hold for a sampled-data/discrete-control system with the provision that the sample/control cycle is small compared with the major time constants in the system. For example, a discrete control system with a sample cycle of 1 millisecond controls a conventional industrial servo motor in much the same way an analog controller does. The delay introduced by the motor is usually more than 10 milliseconds. Generally a discrete digital controller is implemented by a microprocessor. The sample/control cycle can be made small compared to the delays caused by the system components.

In this chapter computer simulations and their graphical results for the discrete system with a PD controller will be presented. Due to the high frequency noise caused by the quantization, a digital lowpass filter is added to the controller in addition to the PD function.

3.1 The Mathematics of the Simulator

In order to use computer languages to simulate a real-time, discrete control process, it is usually necessary to derive a set of difference equations from the system's mathematical model and then apply repeated computations to

obtain values of system variables. In the position control system, the system variables could be the position, the motor speed and the motor torque or current.

For a frequency-domain function, there exists a corresponding time-domain function by inverse Laplace transform. For example, S-domain function $Y(S)=X(S)/S$ is an integral in continuous time-domain. It can be approximated by a discrete time function or an accumulation. Likewise, function $Y(S)=S\ X(S)$ is the S-domain expression for the time-domain derivative or differentiation. Several transforms from S-domain functions to time-domain functions that are useful the simulation are listed in Table 3.1.

Table 3-1.

	S-domain	Discrete Approximation
Proportional	$Y(S) = K \cdot X(S)$	$y(n) = K \cdot x(n)$
Derivation	$Y(S) = K \cdot S \cdot X(S)$	$y(n) = K \cdot \frac{x(n) - x(n-1)}{\tau}$
Integration	$Y(S) = K \cdot \frac{X(S)}{S}$	$y(n) = y(0) + K \cdot \tau \cdot \sum_{i=0}^{n} x(i)$

* τ is the sampling interval.

Applying these relationships to the model of the motor (Figure 2-1), we obtain a set of discrete functions which discribes the motor variables as functions of time:

$$Vr(n) = Vr(0) + \frac{T}{Te} \cdot \sum_{i=0}^{n} [U(i) - E(i) - V_f(i)]$$

$$E(n) = E(0) + \frac{T}{Tm} \cdot \sum_{i=0}^{n} Vr(i) \qquad (3\text{-}1)$$

$$N(n) = \frac{1}{Ke} \cdot E(n)$$

These functions are good at high motor speed. When the motor operates at low speed, the effect of mechanical friction has to be considered. Because of the brushes the mechanical

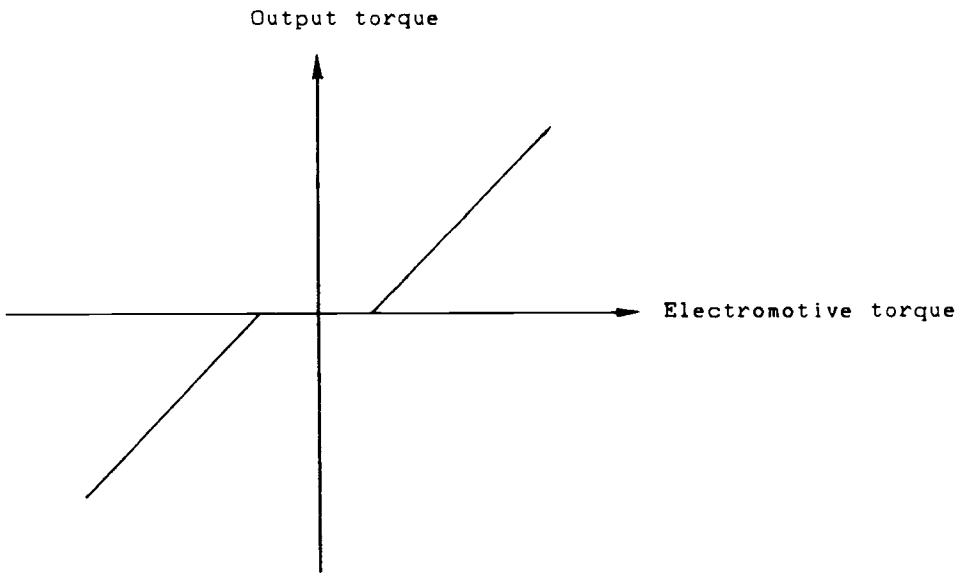


Figure 3-1. Dead Zone in Motor's Output Torque

friction in a d.c. servo motor is considerable. Figure 3-1 shows the output torque versus the electromotive torque in an actual d.c. servo motor. Equation 3-1 can be modified to include the effect of mechanical friction. This was done in the simulation program in Appendix III.

Similarly the discrete functions for the position transducer can be derived from Equation 2-1:

$$P(n) = P(0) + Kt \cdot \tau \cdot \sum_{i=0}^{n} N(i)$$

Using the S-domain model for the PD controller (Figure 2-2), the discrete time-domain equation for the PD controller is

$$U(n) = Kc \cdot Z \cdot ERR(n) + \frac{Kc}{\tau} \cdot [\ ERR(n) - ERR(n-1)\ ]$$

where variable ERR(n) is the position error at the moment of the nth sampling.

At this point, we have obtained the complete set of discrete time-domain functions which is sufficient for the system simulation.

3.2 Simulator Algorithms

The simulation process is essentially a process of repeated computations using the numerical functions that describe the system. It can be accomplished by a higher level language computer program. In this study the system

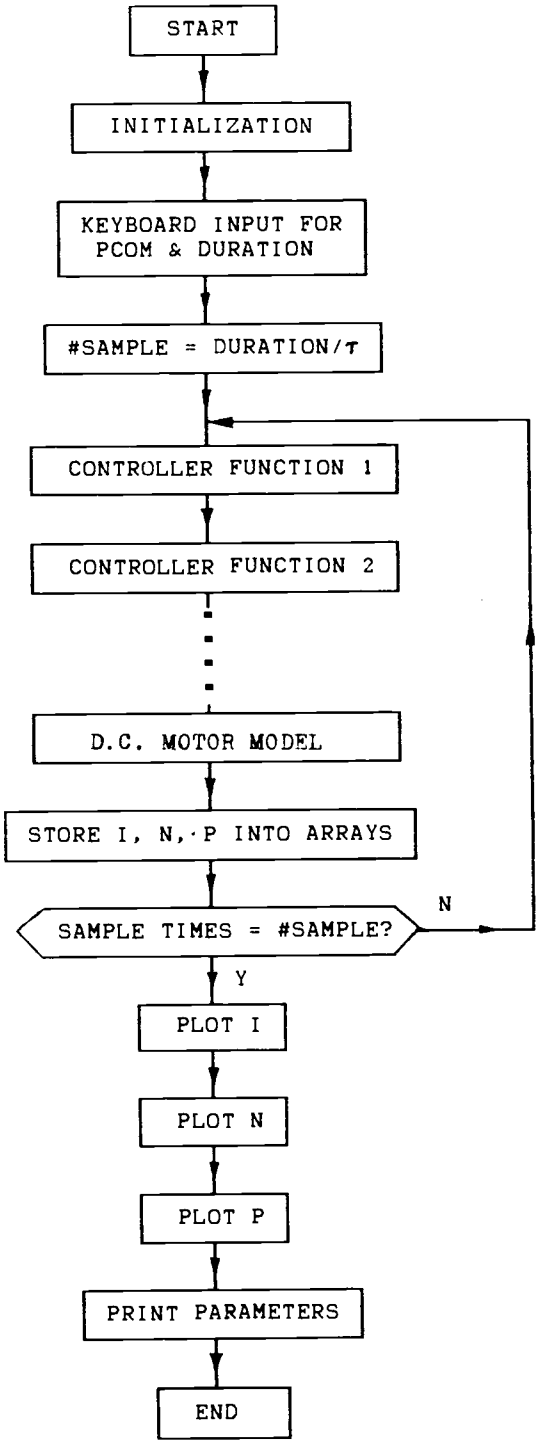


Figure 3-2. Simulator Program Flow Chart

simulation is done with a BASIC program. Discrete functions derived in the last section are used. The program simulates three important system variables: the motor's armature current, the motor speed and the position. The quantization of variables is realized by defining these variables as integers in the BASIC program. Quantization of variables is necessary since in a microprocessor controlled system most variables are integers or quantized analog quantities.

Figure 3-2 shows the control flow of the simulator. Detailed information about the simulator can be found in Appendix III.

3.3 Simulation Results with a Linear PD Controller

In this research a small d.c. servo motor E522-MG (Electo-Craft) was chosen for simulation. From the manufacturer's specifications the three useful motor parameters are:

$Tm = 0.041$ sec

$Te = 0.002$ sec

$Ke = 0.0042$ v/rpm

The transfer function of the motor is

$$\frac{N(S)}{U(S)} = \frac{1/Ke}{TmTeS^2+TmS+1}$$

$$= \frac{2.9\times10^8}{(S+26.5)(S+450)}$$

The position sensor used was a shaft encoder mounted on the motor's shaft. There are 12 holes on the encoder disc. The approximated transfer function for the position transducer is

$$\frac{P(S)}{N(S)} = \frac{Kt}{S}$$

$$= \frac{0.2}{S}$$

where Kt = (number of holes)/(seconds in a minute).

The location of the controller zero, Z, was chosen to be -8.33 on the real axis of the S-plane. Controller gain Kc is experimentally chosen to be 0.06. With these parameters the controller's transfer function is

$$\frac{U(S)}{ERR(S)} = KcS+KcZ$$

$$= 0.06S+0.5 \qquad (3\text{-}1)$$

The sampling interval used in the simulation was 0.002 second.

Figure 3-3 illustrates the simulated position step reponse. Three system variables of interest are plotted. As can be seen from the figure, the position response is approximately an exponential curve; There is no overshoot or oscillation. These characteristics agree with what

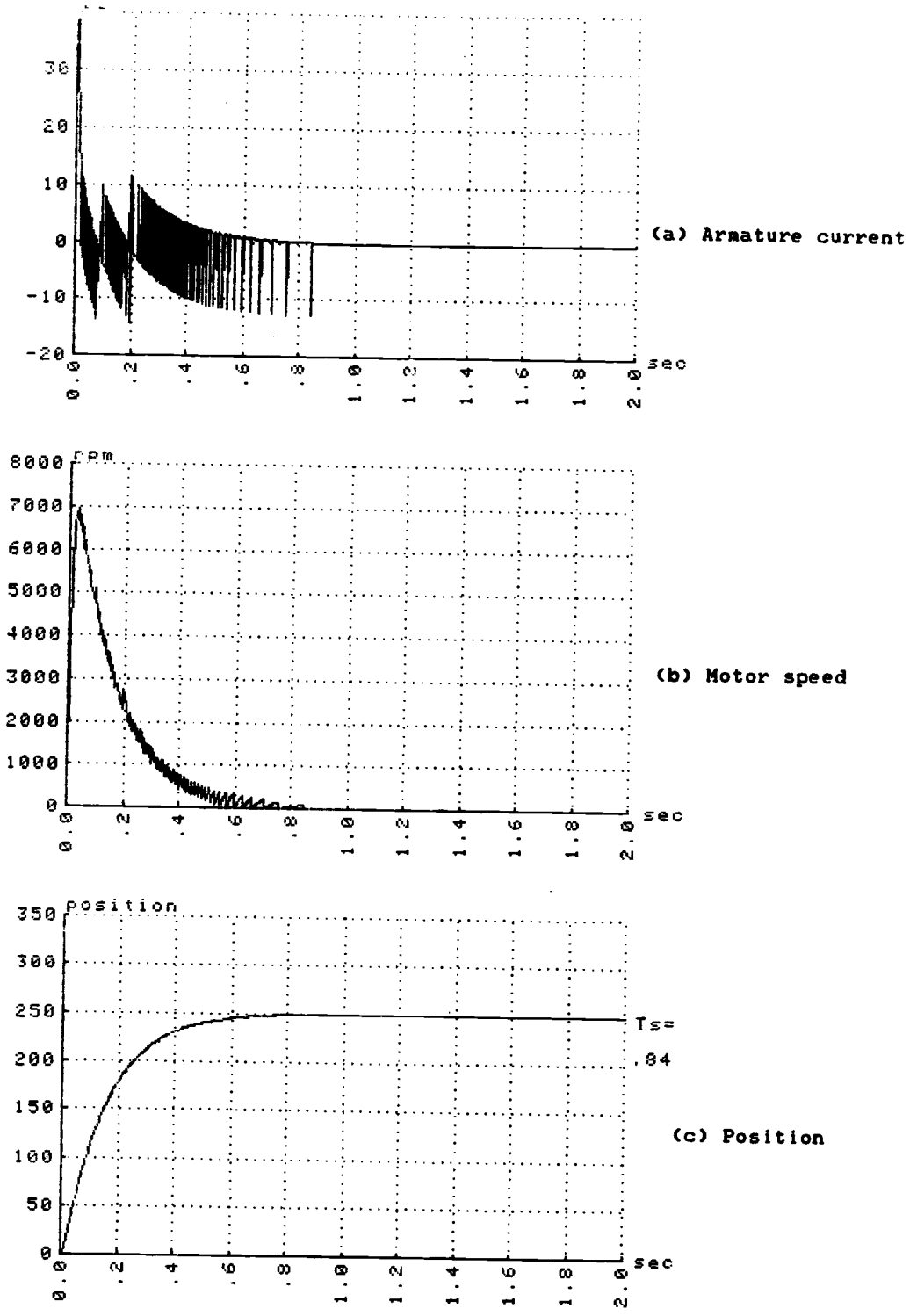


Figure 3-3. Simulated step position response

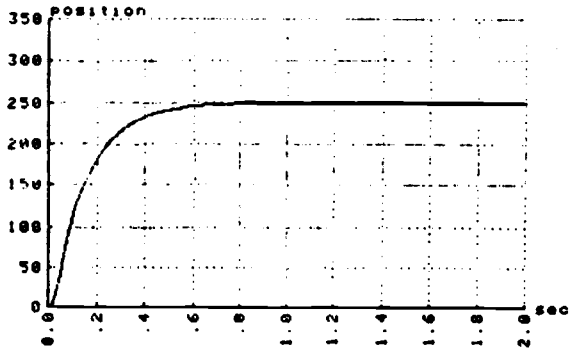
predicted by the linear analysis in Chapter 1: the dominant poles are real and negative.

On the other hand, undesirable high frequency noise occured in the motor current and speed. We leave this problem to Section 3.5 where quantization noise is discussed and solved.

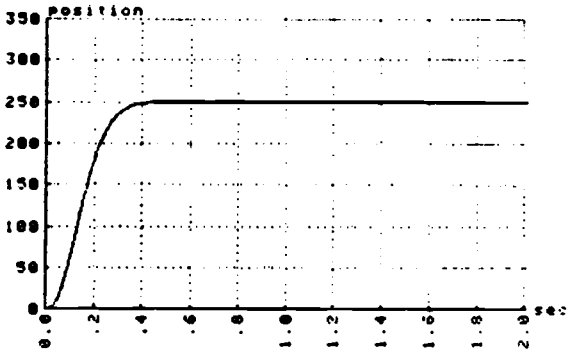
3.4 Experiment With Different Motor Inertia

Tests show that the controller zero, Z, plays an important role in the system's ability to resist the change of total motor inertia. The total motor inertia is the motor armature's inertia plus the reflected external mechanical inertia. The greatest delay factor, Tm, is proportional to the total motor inertia. When there is no external mechanical mass connected to the motor shaft, Tm is the same as specified by the motor parameter. When external inertia is added Tm becomes larger. Since Tm is the major delay factor in the system, variation of Tm could stongly affect the system's performance.

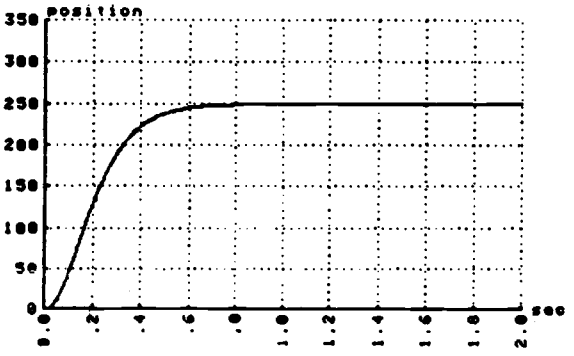
By introducing an appropriate zero into the system's transfer function (this is equivelant to employing the differential control) the influence of the variation of Tm can be eliminated to some extent. Figure 3-4(a) thru 3-4(d) illustrate a contrast of position response with different Tm from two systems. System 1 employs the PD controller. System 2 uses proportional control only. The performance of system



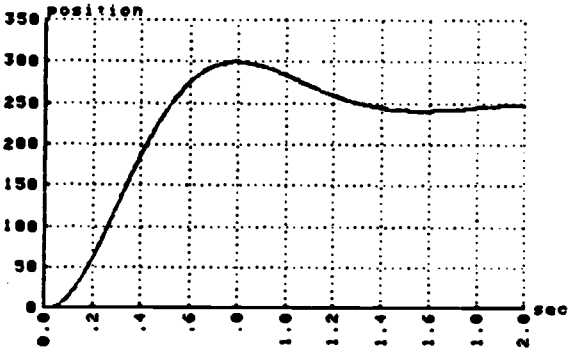
(a) System 1



(b) System 1



(c) System 2



(d) System 2

Figure 3-4. (a) System 1 with original Tm; (b) System 1 with 5Tm; (c) System 2 with original Tm; (d) System 2 with 5Tm

2 strongly depends on the value of Tm: The greater the Tm, the slower the response. The system can even oscillate with large Tm. On the contrary system 1 seems to be relatively insensitive to the variation of Tm.

Experiments also show that the open-loop gain and the controller's zero jointly determine the performance of the system. Figure 3-5 plots the controller gain Kc versus Tm with different Z (refer to Equation 3-1), under critical damping condition. The controller zero Z affects the system performance in the way that the smaller the value of Z, the wider the damping zone (refer to Figure 3-5). In other words, a smaller Z allows the system to endure a larger motor inertia variation. However, since the setting time of the response is proportional to 1/Z, a smaller Z also results in a longer response time. The choice will eventually be a compromise between the adaptability of the system to inertia change and the quickness of the response. This is an important criteria for the design of the system.

3.5 Lowpass Filtering

An undesirable effect of the quantization is the existance of the quantization error in the controller output. As a function of time, the quantization error is a random process. Although the time average of the error is zero, the error process does add an a.c. noise to the motor input voltage. After being scaled (enlarged) by the

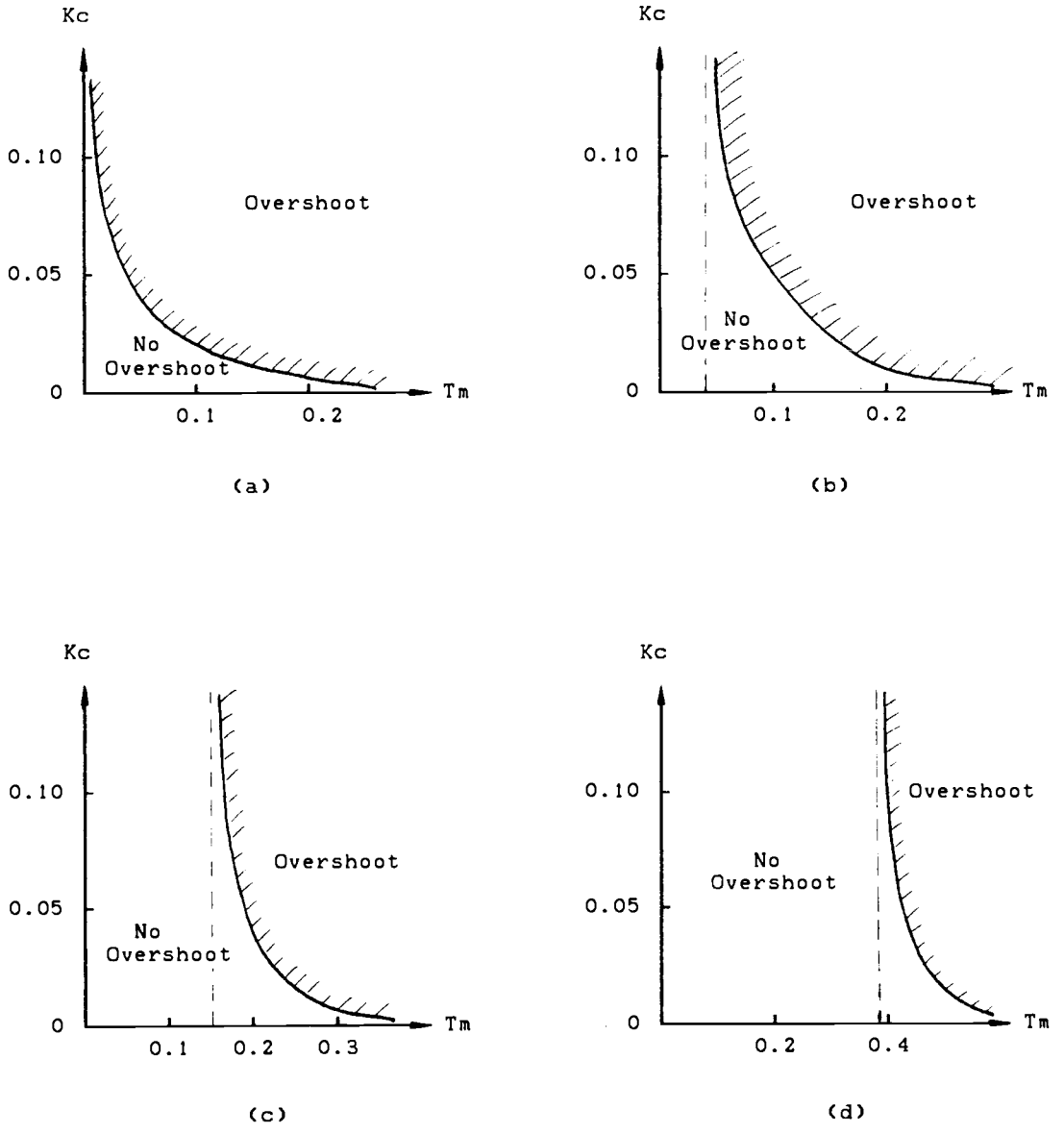


Figure 3-5. Kc vs Tm Under Critical Damping. (a) Without S+Z; (b) Z=26.5; (c) Z=10; (d) Z=5.3

controller the quantization noise becomes quite noticeable. As a result there are spikes in the motor's armature current (see Fig. 3-3). This high frequency noise contains no d.c. component, hence it does not affect the position because of the mechanical inertia. However, the effective a.c. current increases the power dissipations of the motor and the driving circuit, so it is harmful.

To eliminate the quantization noise, a lowpass filter can be used to limit the bandwidth of the output voltage, shown by Figure 3-6. The pole of the filter M should be greater than the controller zero so that the filtering does not destroy the effect of the zero compensation. A lowpass filter pole value several times as large as the value of Z will give a satisfactory result. The discrete function for the lowpass filter is:

$$U(n) = U(0) + \tau \cdot M \sum_{i=0}^{n} [U1(i) - U(i)]$$

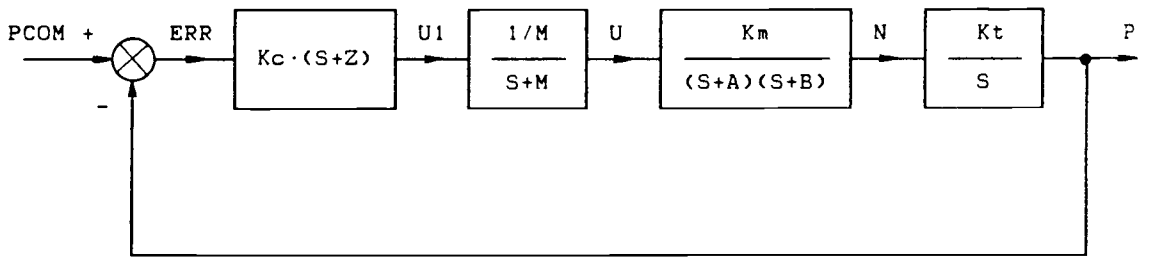
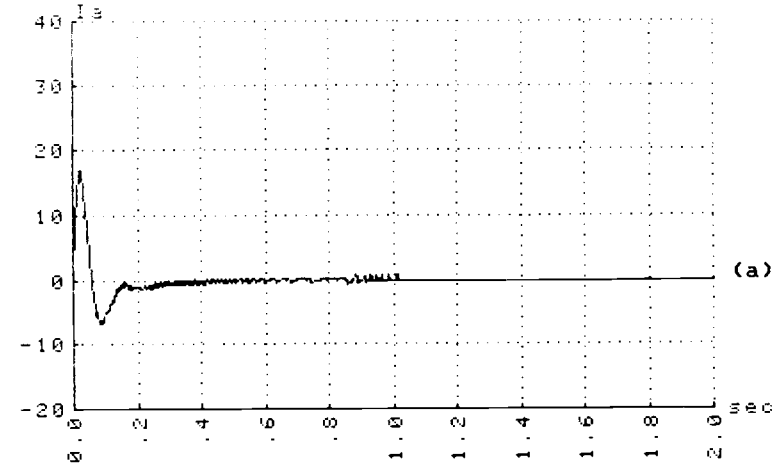
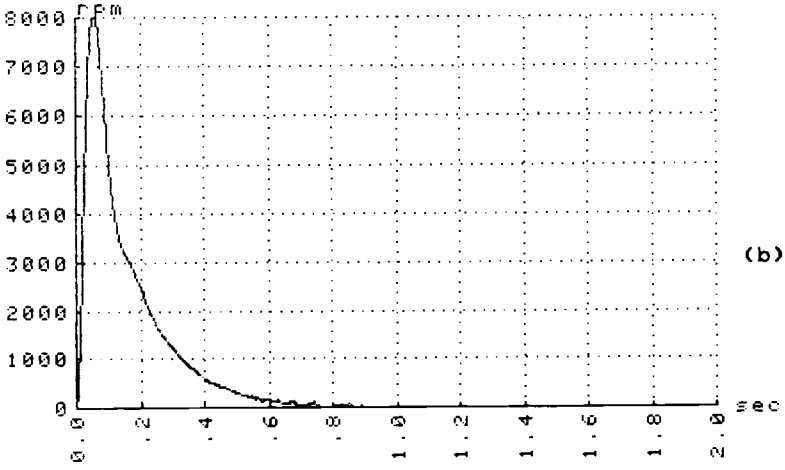


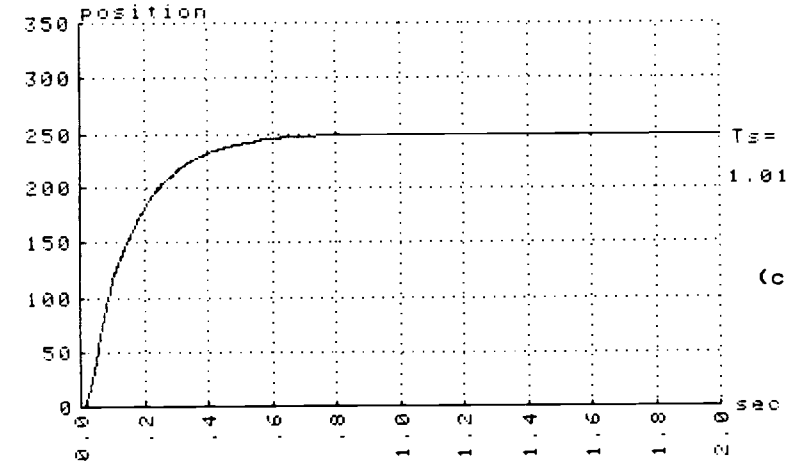
Figure 3-6. Block diagram of system with Lowpass Filter



(a) Armature current



(b) Motor speed



(c) Position

Figure 3-7. Removal of the Quantization Noise

Figure 3-7 illustrates the step response with the lowpass filter; the filter pole is 31.25 rad/sec. Compared with Figure 3-3, both the current and the speed are much smoother.

3.6 Summary

Conclusions from the linear analysis of the continuous system are also valid for the discrete system if the sample rate is high enough.

A set of discrete functions can be derived from the system's transfer functions. These functions completely describe the discrete system in time domain.

The simulation process is essentially a process of repeated computation using the discrete functions. By setting the relevant system variables as integers in the simulation program the quantization in the real digital system can also be simulated.

The simulation results exhibit the same characteristics that are predicted by the linear analysis; there is no overshoot or oscillation in the position response. The system also proves to be insensitive to the inertia variation if the controller parameters are properly chosen.

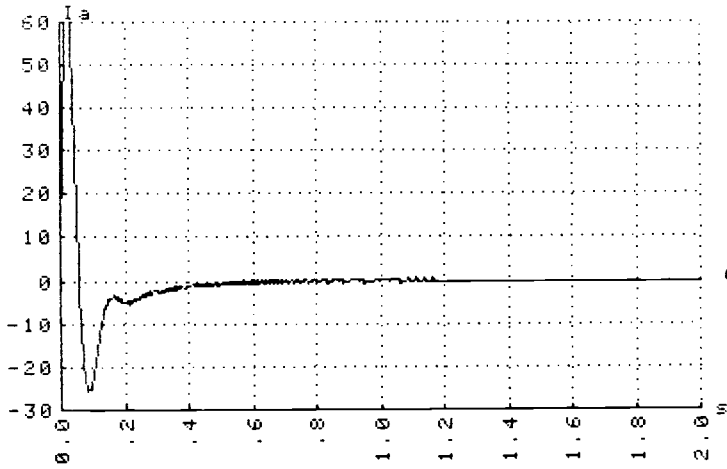
Quantization noise is harmful to the motor and the circuits. A lowpass filter can be used to eliminate the noise from the controller output.

4. NONLINEAR DYNAMIC COMPENSATION

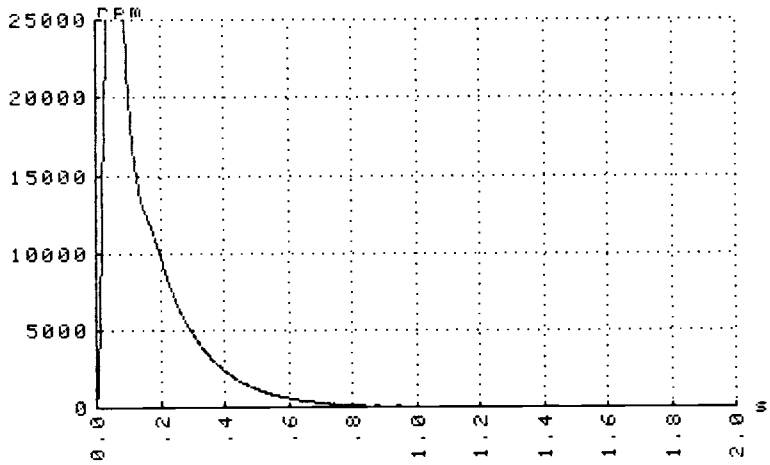
The position control system developed in the previous chapter is a linear system. By linearity it is meant that the magnitude of any system variable is proportional to the magnitude of the system input. System variables of physical significance, however, usually have certain limitations. For example, the motor used in the simulation has a speed rating of 7000 rpm and an armature impulse current rating of 20 amperes. Neither of the two is supposed to exceed their ratings. This problem is illustrated by Figure 4-1 in which a step position input of magnitude 1000 is applied to the linear system. Both the motor speed and the current exceed their rated values. Although a small system gain can help lower the speed and current profile, the resultant system will be very slow.

Another way to look at the problem involves the utilization of motor power which is directly related to the response speed. In a linear response the motor speed curve is peak-like. The motor speed is low for most of the time during the response process and results in a long response time. The solution to this kind of problem is beyond the capabilities of linear compensations.

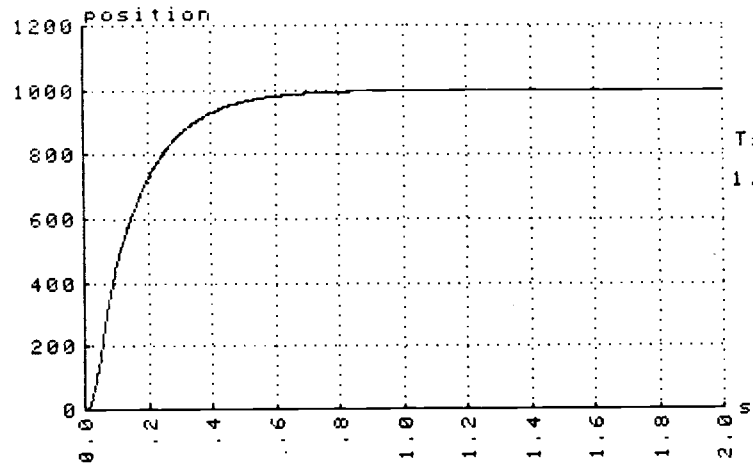
In this chapter, a non-linear approach to the solution is discussed and the simulation results are presented. Some minor problems are also treated with non-linear methods to



(a) Armature current



(b) Motor speed



(c) Position

Figure 4-1. Overload of current and speed

further improve the controller. The final system model, which will be used in the real-time experiment in Chapter 4, will be determined.

4.1 Using Nonlinear Functions to Achieve Optimal Control

For the protection of the motor and achieving the fastest response the ideal motor speed curve should look like a square wave. The corresponding current curve and position curve are shown by Figure 4.2. The optimization is accomplished by keeping one system variable (motor speed) at its extreme. However since this requires current impulses with infinite magnitude, these curves are only theoretical.

The physically possible optimal control is to keep the armature current at the rated value when the motor speed is changing (Figure 4-3). This also optimizes the system since at any instance there is always a system variable kept at its extreme. The realization of such a control needs a current loop within the position loop in the system structure and the supporting hardware, such as a current transducer.

An alternative is to limit the motor armature voltage. By limiting the armature voltage to the voltage rating the motor speed is normally limited to its rating, and so is the armature current. This is illustrated by Figure 4-4. In this approach the armature current is limited to but not kept at

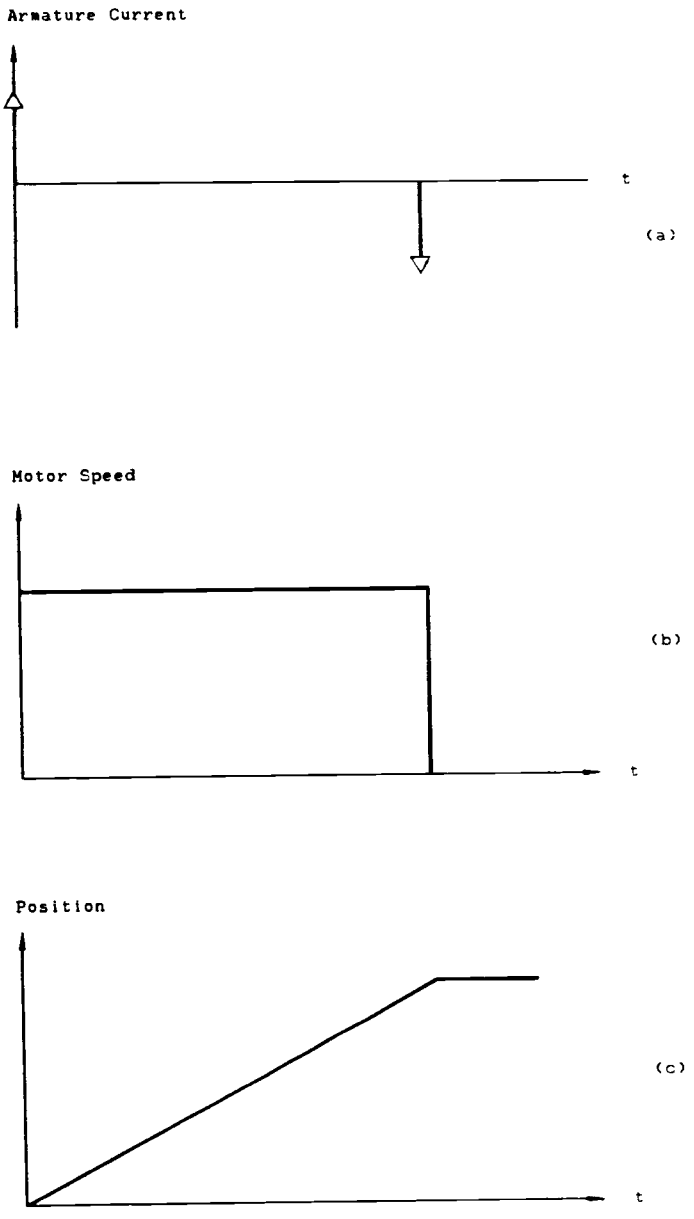


Figure 4-2. Optimal Control with Ideal Speed Curve

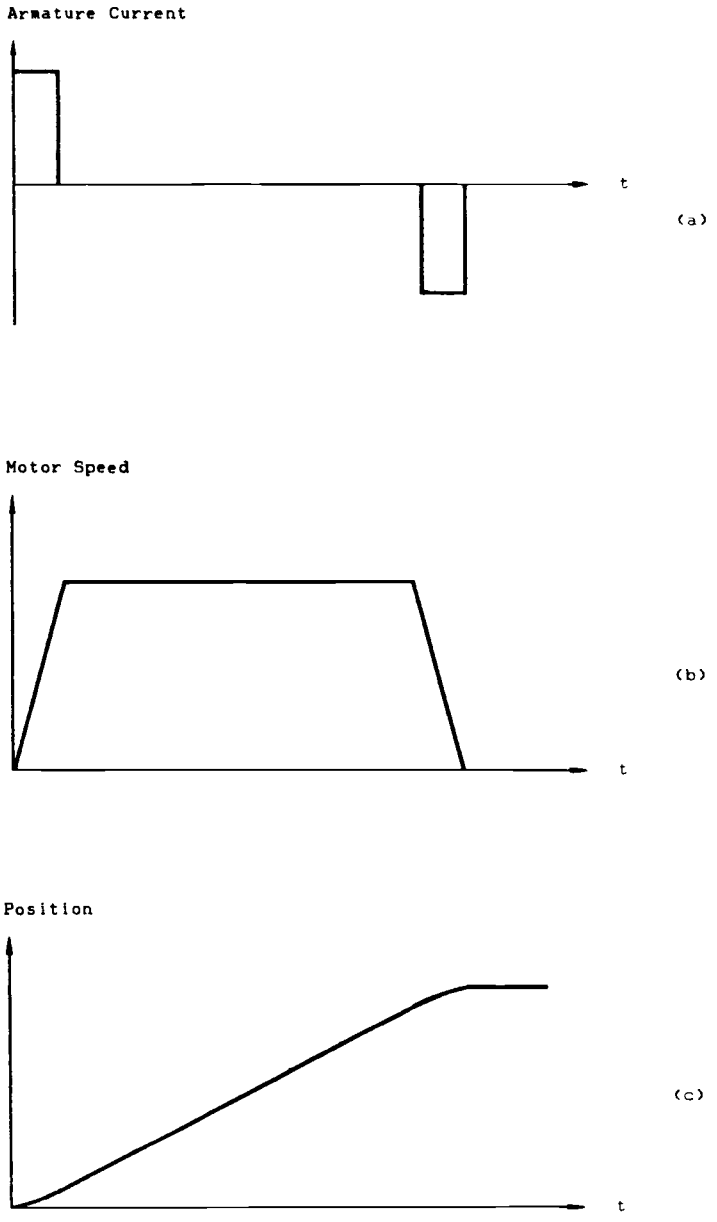


Figure 4-3. Realizable Optimal Control

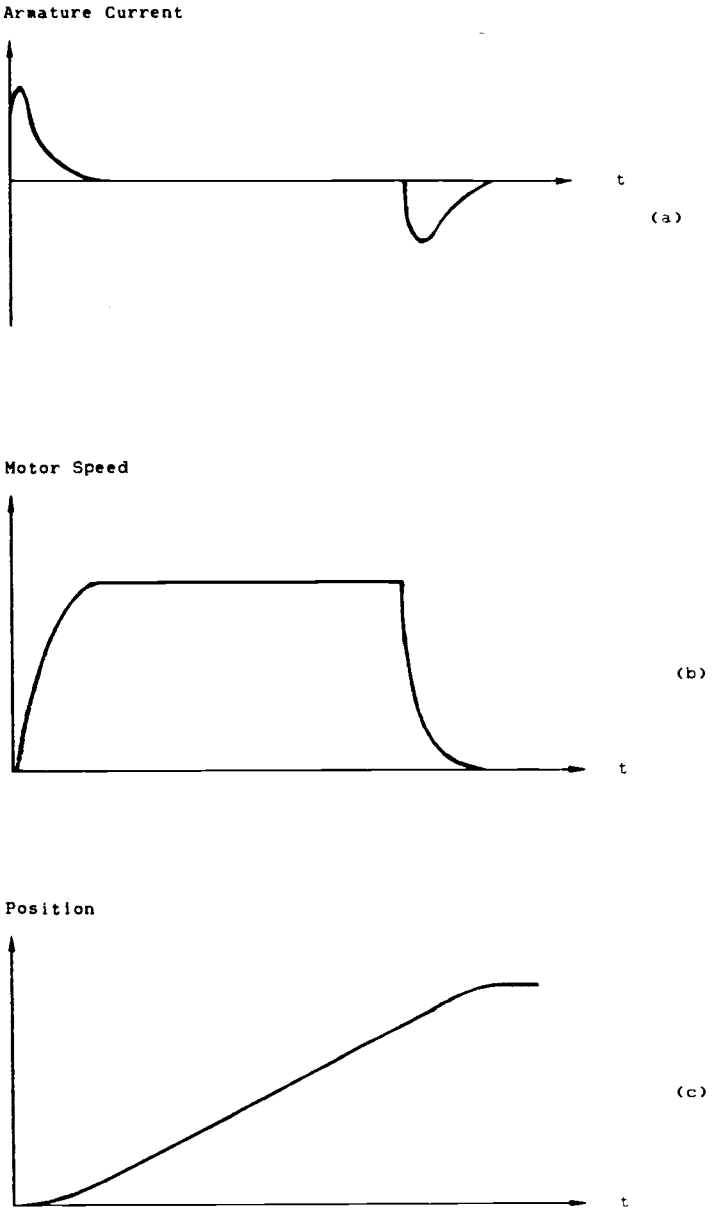


Figure 4-4. Quasi-optimal Control with Saturation

the maximum. Unlike the optimization of Figure 4-3, the optimized portion occurs where the motor speed curve is flat. This normally happens during large **excursion.** Therefore, a greater position input will result in a better optimization and vice versa. Hence the control is "quasi-optimal". The advantage of the quasi-optimal control is its simplicity. Since the motor armature voltage is the controller's output, the control can be easily implemented by using a saturation function (Figure 4-5). The function is symmetrical and has a unity gain. The positive and negative thresholds are set to Umax and -Umax, respectively. Umax can be set to equal to or less than the motor's voltage rating. It determins the maximum motor speed.

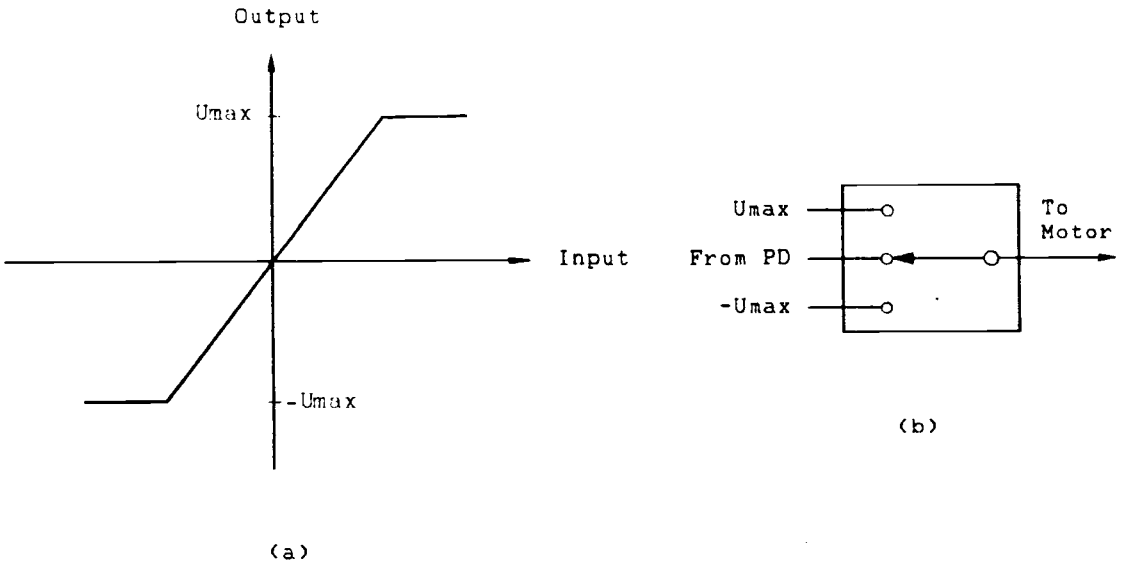
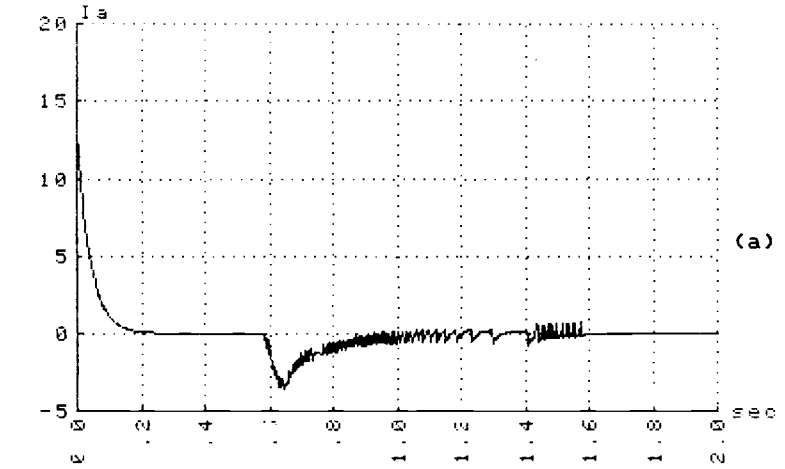
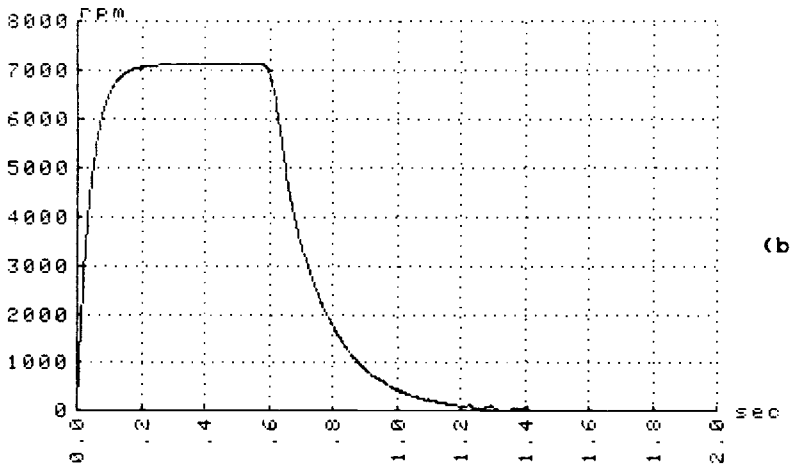


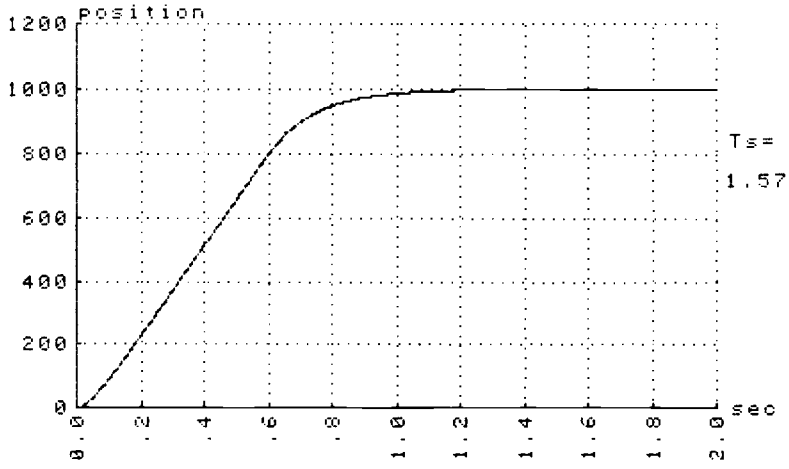
Figure 4-5. Non-linear Function For Quasi-Optimal Control



(a) Armature current

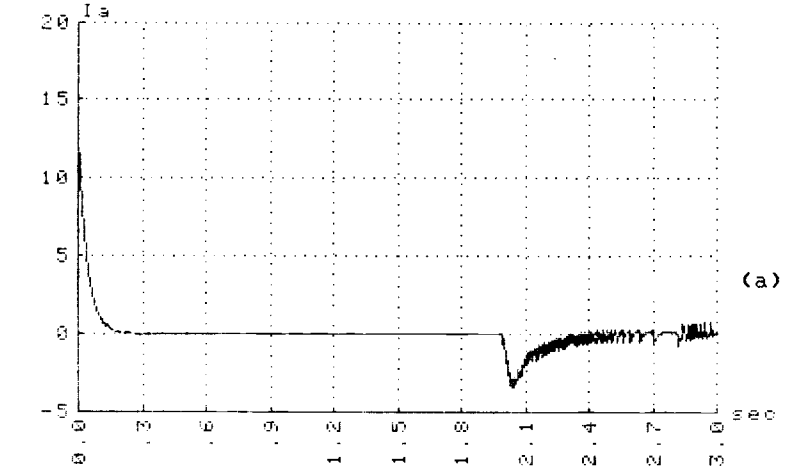


(b) Motor speed

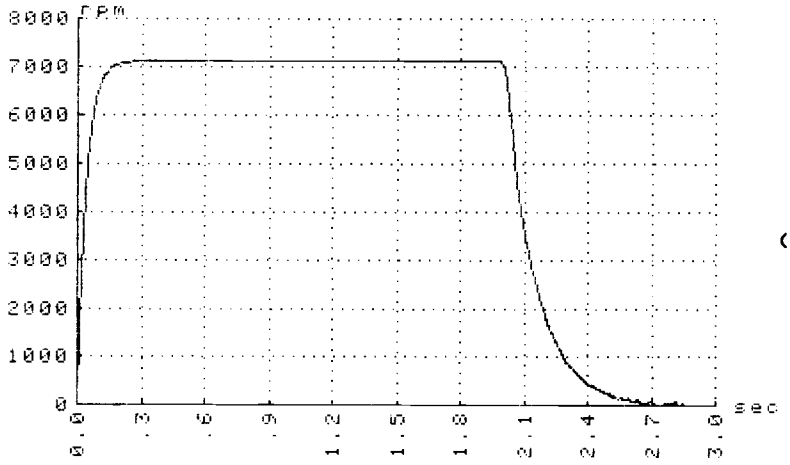


(c) Position

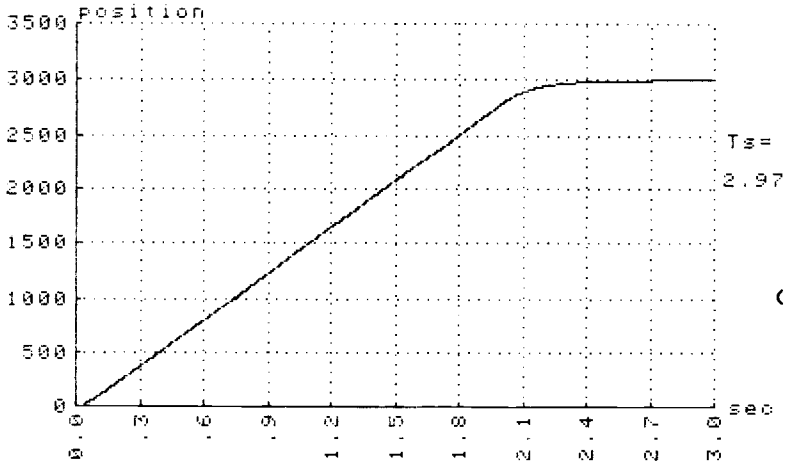
Figure 4-6. Simulated Quasi-Optimal System with Position Input PCOM=1000



(a) Armature current



(b) Motor speed



(c) Position

Figure 4-7. Simulated Quasi-Optimal System with Position Input PCOM=3000

The response of the system implemented with the nonlinear function is simulated and illustrated by Figure 4-6 and 4-7. As can be seen, both the armature current and motor speed are limited to their ratings. The effect of nonlinear control is more obvious with large excursion (Figure 4-7) in which the motor operates at its maximum speed for most of the time during the response processes.

4.2 Other Nonlinear Functions

Looking at the current curves in the previous simulations, we can find a number of small spikes appearing after the position reaches the terminal position, PCOM. This is caused by the quantization error to which the differential control function is very sensitive. Although the linear lowpass filter removes most of the quantization noise the remaining amount is still big enough to cause the spikes. These spikes, though not harmful, cause position vibrations by the end of the control process, shown in Figure 4-9(a), thus lengthening the response time.

To eliminate the vibration, the differential function can be removed from the controller once the position error is zero (Figure 4-8). The controller will do exactly the same job until the very last moment of the control process. Then it switches to a P type controller which is not sensitive to the quantization noise. When there is any position error detected the controller will restore the

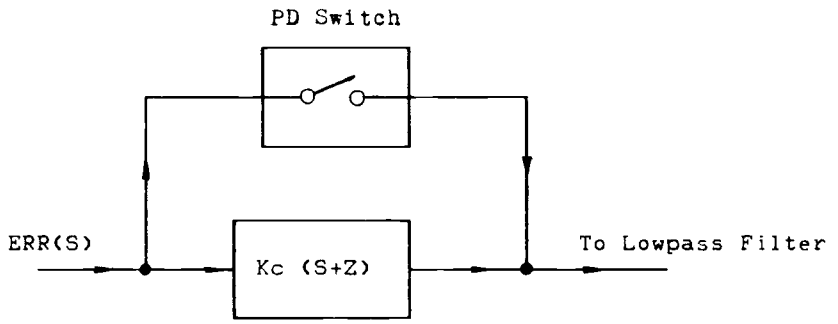
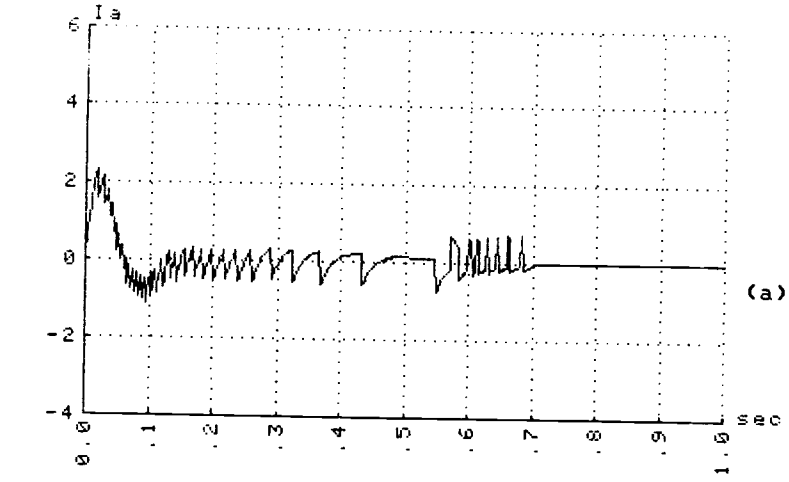


Figure 4-8. Non-linear Function To Eliminate Position Vibration

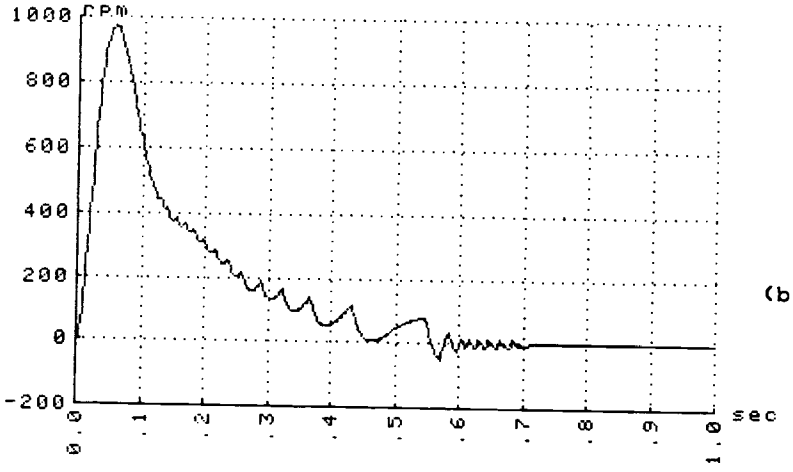
differential function again. Figure 4-9(b) shows the response with the dynamic switch function. The vibration is eliminated while there is no other change (compare the two graphs).

Another problem is the mechanical friction that exists in a d.c. servo motor. The mechanical friction comes from the brush-rectifier **contact**. It causes a dead zone in the motor's output torque, as shown by Figure 4-10(a), and could lead to minor oscillation under certain conditions. By adding a small voltage to the controller's output, the dead zone can be eliminated. This non-linear function is illustrated by Figure 4-10(b).

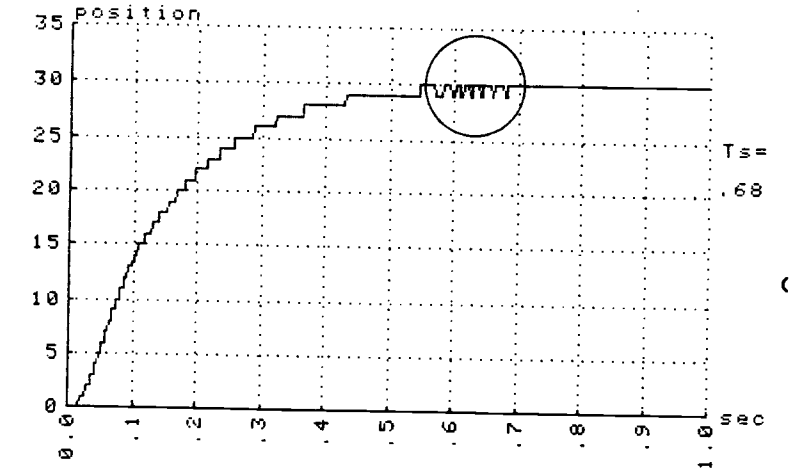
Figure 4-11 shows the system equipped with all the non-linear functions developed in this chapter. The design of



(a) Armature current

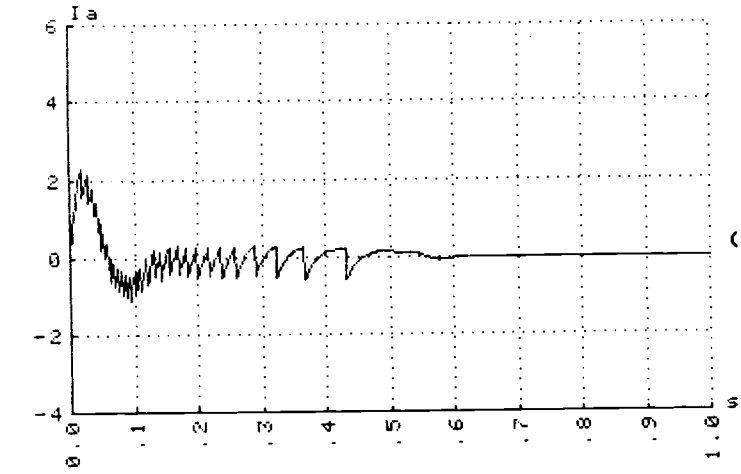


(b) Motor speed

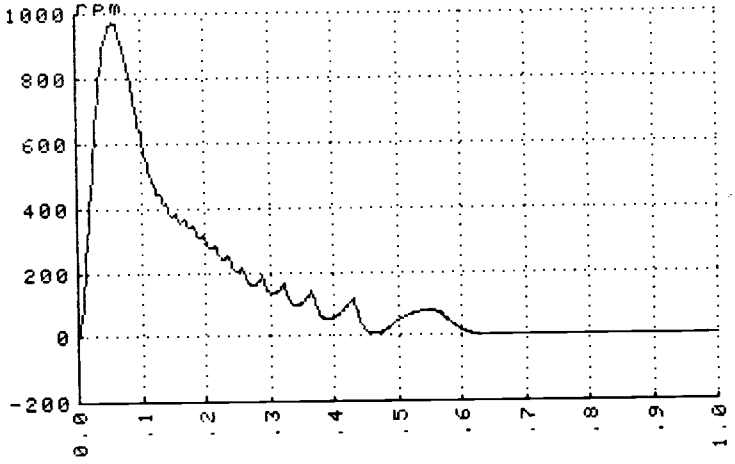


(c) Position

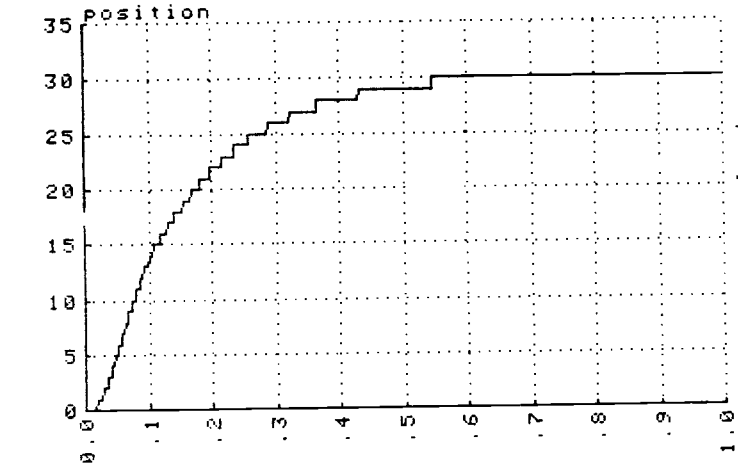
Figure 4-9(a) Vibration of position



(a) Armature current



(b) Motor speed



(c) Position

Figure 4-9(b) Vibration eliminated

the controller is completed at this point and this system model will be used in the next chapter for the real time micro-processor control experiment.

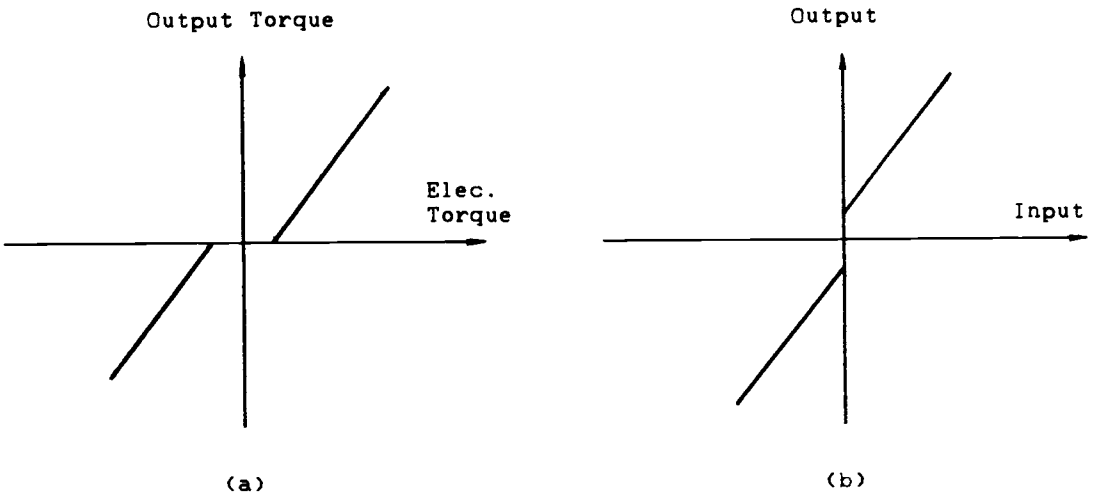


Figure 4-10. Torque Dead Zone and Compensation
(a) Dead Zone of Torque Due to Friction (b) Compensation Function

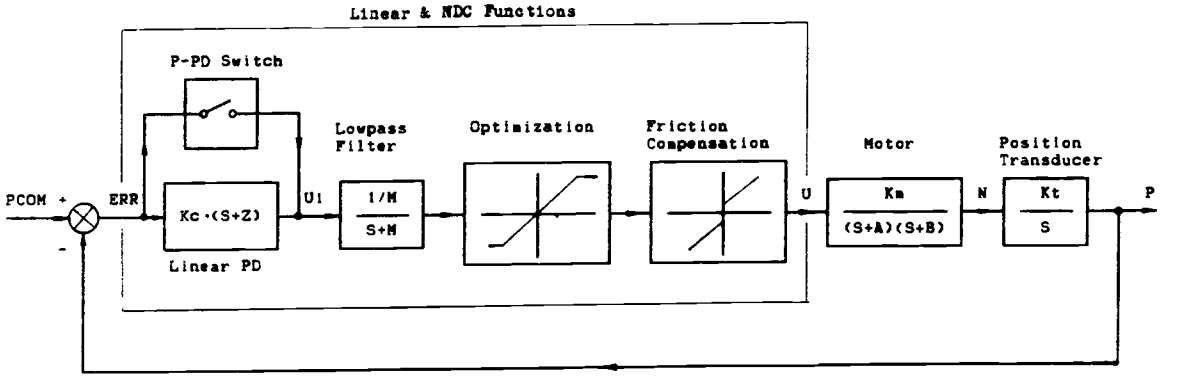


Figure 4-11. System with the NDC Functions

4.3 Summary

The linear system model has trouble when applied to reality: the values of some system variables can exceed their physical limitations. Also, in the linear system the utilization of motor power is poor.

Nonlinear functions can effectively limit the amplitudes of the variables and, more importantly, realize the optimal control. Since the motor power is fully used in the optimized system, such a system has the fastest response.

The quasi-optimal system is simple and economical. It employs a nonlinear function to modify the control output. The degree of the optimization is close to that of the fully optimized system when the input is large.

The minor vibrations near the end of the control process, caused by quantization noise, can be eliminated using a second nonlinear function. A third nonlinear function is used to deal with the motor's mechanical friction.

After adding three nonlinear dynamic compensation functions to the controller the system model is finally determined.

5. REAL TIME MICROPROCESSOR CONTROL EXPERIMENT

This chapter describes a real-time digital position control experiment, using an 8-bit microprocessor. The real-time system is based on the model developed in the previous chapters. The experimental results will be presented and compared with the simulated results.

5.1 Description of System Architecture

The real-time control system has the same structure and control algorithms as the simulated system. It consists of a d.c. servo motor, a position transducer, a driving circuit and a controller. The controller is a microprocessor (Intel 8039). In addition to these, the system is implemented with communication facilities which make the communication between the host and the controller possible. Figure 5-1 shows the structure and the control hierarchy of the system.

5.1.1 The Execution of the Controller Program

The controller program resides in the program memory of the microprocessor as a subroutine of the AGILE operating system . Through the operating system the controller is accessed by the "host computer", an IBM PC, in which a BASIC demonstration program is running.

The operating system was developed by Dr. James H. Herzog at Oregon State University. It provides an easy way for a personal computer to communicate with devices

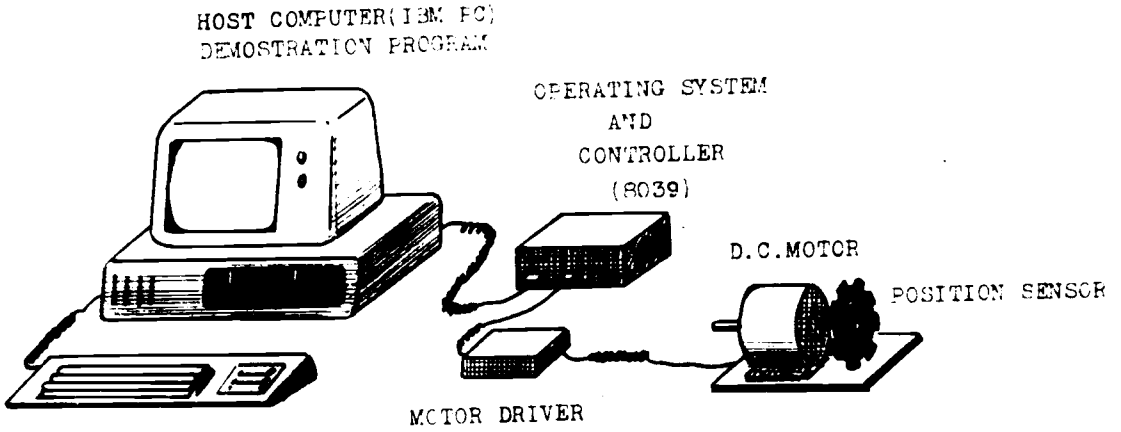
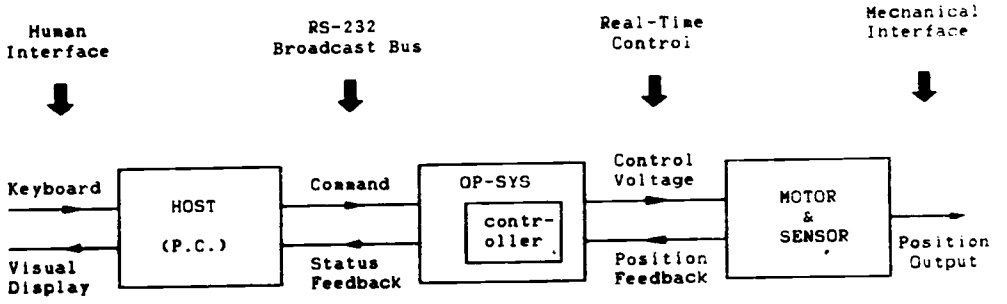


Figure 5-1. (a) Structure of the Real Time System



(b)

Figure 5-1. (b) Architecture of the System Showing the Control Hierarchy

connected to the MCS-48 microprocessor. By sending the ASCII character command packets via the RS-232 line the PC is able to control the various functional subroutines or "tasks" in the operating system. The position controller program is also written in the form of a task subroutine.

Figure 5-2 illustrates the control algorithms of the controller program. The first section of the program sets all the constants in the controller. These constants are related to the open-loop gain, the zero of the controller, the pole of the lowpass filter, the voltage limit for the motor, etc.. Since they are stored in the data memory of the microprocessor, their values can be changed later by the user through the operating system. Then the program enters the 2 millisecond control loop.

The control loop starts with resetting the microprocessor's internal timer. Then it inputs the position command from the host and puts it into the position command buffer. After that it inputs the motor position from the parallel port and put it into the position buffer. The position command is a 16-bit number sent to the controller by the host computer. It contains position setting and other information. The program then checks the MSB of the position command to determine if the host computer is requesting a data transmission.

The data transmission section sends the current motor

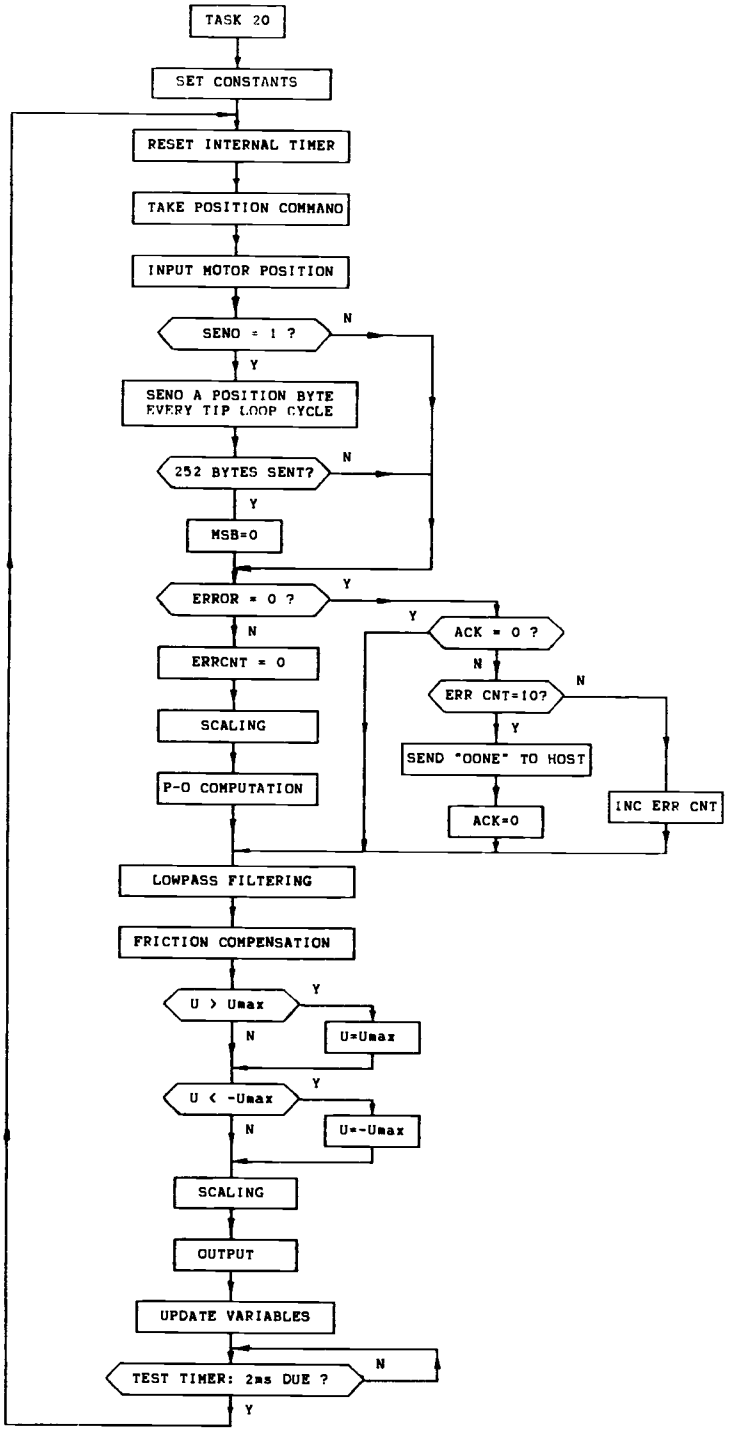


Figure 5-2. Controller Program Flow Chart

position data to the host computer via the operating system at a fixed interval of time. The transmission duration can be controlled by the user by sending a control number from the PC. 252 position data bytes* are sent from the controller to the host during the transmission. These 8-bit data bytes record 126 double byte motor position numbers as sensed by the position sensor during the transmission. If transmission is not requested this section is skipped.

Then the position error is detected by subtracting the present motor positon word from the position setting word. If there is no error the program branches to the lowpass filter section directly (this is one of the nonlinear functions in the controller). Also, if the error is not detected for 10 consecutive times, and if requested by the host, the controller sends a "done" signal to the host to indicate the end of the control process. If there is any error, the program enters the PD computation section in which the error is scaled and processed by the differential control function and the proportional control function. The result is then fed into the lowpass filter section through which most of the quantization noise is filtered out.

The lowpass filter section is followed by the other two nonlinear functions, one adds a small increment to the control signal to overcome the motor's mechanical friction

* This number (252) was used because the IBM PC BASIC's communication buffer has a length of 255 bytes.

and the other limits the output voltage to Umax or -Umax.

In the output section the controller output is scaled to an 8-bit number and is sent to a parallel interface port on the microprocessor which drives a D/A convertor.

After completing the control activities in one loop cycle the program enters an idle loop, testing the microprocessor's internal timer. When the timer indicates that two milliseconds have elapsed since the beginning of the cycle, the program branches to the start of the loop and a new control cycle begins.

The numerical computations needed by the various functions are performed by the computation subroutines in the controller program. All the variables are 16-bit signed integers.

The host computer's access to the controller is accomplished by interrupting the execution the controller program. The user sets a position command by sending a string of ASCII characters to the controller operating system. In the character string the position value and the data memory storage address is specified. The controller operating system interrupts the application program to receive and store the ASCII characters. When the complete ASCII command packet is received and interpreted, the controller operating system again interrupts the controller program to put the new position command number into the

controller's position command buffer. Then the interrupted controller application program is resumed.

5.1.2 The Position Transducer

The position transducer built for the experiment consists of an optical shaft encoder and a 12-bit bi-directional counter. The shaft encoder is mounted on the motor shaft. The shaft encoder has an encoding disc and two pairs of infrared LED/photo diodes, shown in Figure 5-3. The electrical phase angle between two adjacent notches is 2π. The two sensors, A and B, are placed $(2k+0.5)\pi$ apart around the disc, where k is a positive integer. When the disc

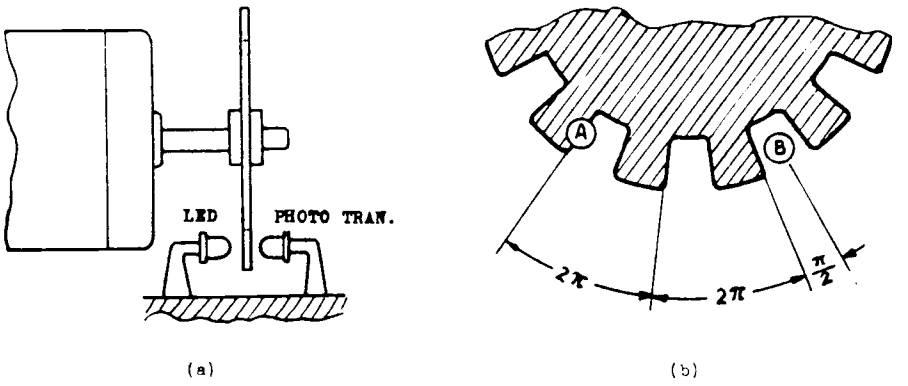


Figure 5-3. The Position Sensor

rotates the sensors transform the motion of the notches to voltage pulse trains. The pulse trains contain information of frequency and direction, from which the position (in the unit of notch numbers or position point) can be determined.

Figure 5-4 illustrates the change of phase in the two pulse trains when the motor rotates in opposite directions. If we define variable D as

D = 1 when pulse A leads;

D = -1 when pulse A lags.

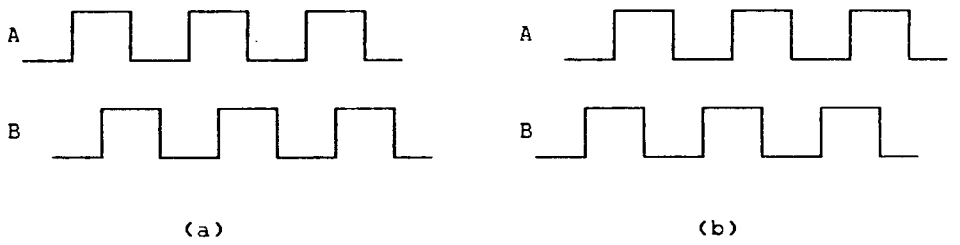


Figure 5-4. Phase Change of Pulses A and B
(a) Clockwise; (b) Counter-clockwise

The motor position can be expressed by equation

$$P_n = \sum_{i=0}^{n} D_i \qquad (5\text{-}1)$$

where N is the total number of pulses received by either of the sensor.

Figure 5-5 illustrates the circuit used in the experiment, which realizes Equation 5-1. Three 4-bit bi-directional binary counters are cascaded to obtain a 12-bit positon number. The counter counts every pulse generated by sensor A. The direction logic determines the rotation direction by checking the phase change of the pulses and generates logic variable $\overline{U}/D$ which controls the counting direction of the counter. A BUSY signal is also generated by

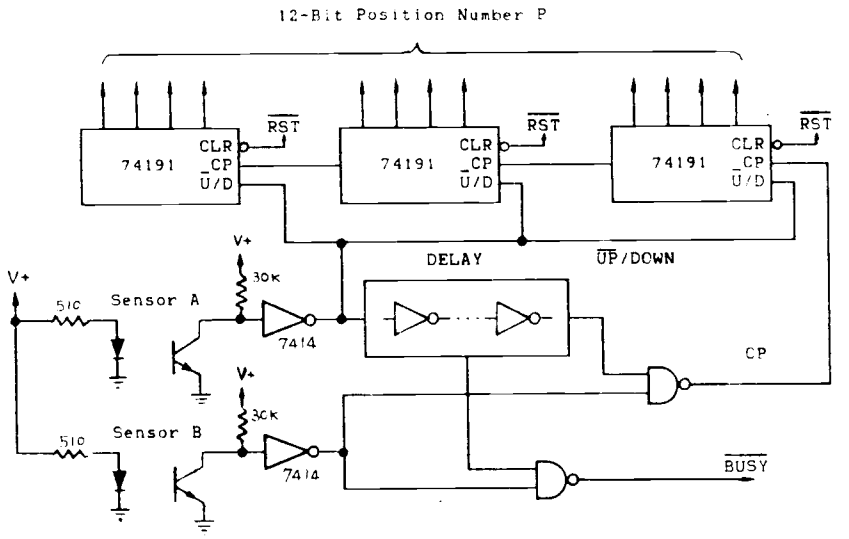


Figure 5-5. Position Transducer Circuit

the logic to prevent the controller from reading the position when the counter is being triggered. Since the counter can count up to 4096 position points, the resolution of the control is 1/4096, or approximately 0.025%.

5.1.3 The Driving Circuit And The Motor

The motor used in the experiment is a 40 watt GE shunt motor. Its major parameters are similar or close to that used in the simulation. The motor is driven by a power amplifier. In order to supply the 20 amperes impulse current needed by the motor, the dynamic source resistance of the amplifier must be very low. A high source resistance could also cause slow acceleration and oscillation. Generally a

source resistance much smaller than the motor armature resistance is required. To obtain a low source resistance,

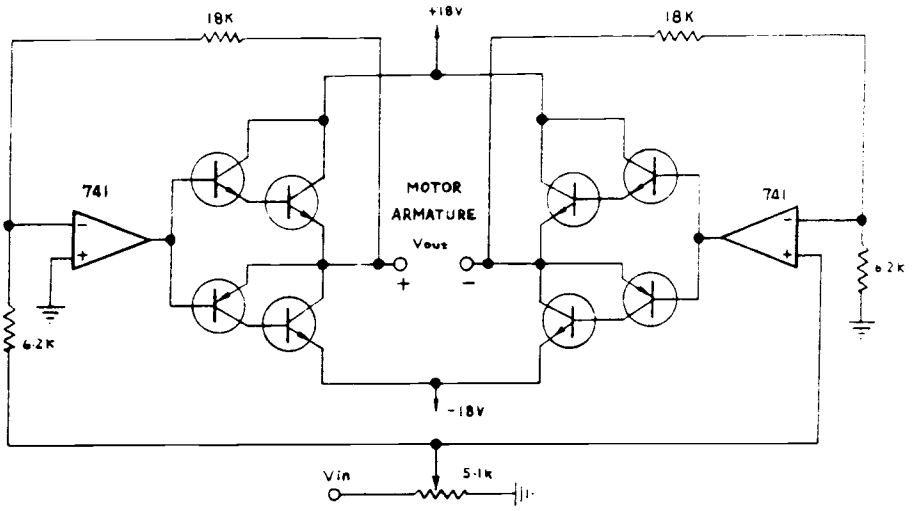


Figure 5-6. Motor Driving Circuit

operational amplifiers were used in the motor driving circuit (Figure 5-6). The measured source resistance of the circuit is less than 0.03 ohm. Compared with the armature resistance of the motor (around 3 ohm), the influence of the amplifier source resistance is negligible.

5.2 Experimental Results

The experiments show that the real-time position control system has all the characteristics predicted by the simulations. The response curves recorded from the experiments are very similar to the simulated response curves.

Figure 5-7 thru 5-12 present some typical experimental

curves, together with their counterparts from the simulation. These experimental results are recorded by the BASIC demonstration program which receives the position data from the controller during the control process. The speed curves are obtained by differentiating the position data. Due to the difficulty in measuring and recording the armature current, the current curves are not presented.

Figure 5-7 is the comparison of the step response resulting from the experiment and the simulation. The input is small so the controller is basically linear. In Figure 5-8 the input is increased. Flat top appears in the speed curve which indicates that the optimization function has been activated. In Figure 5-9, the input is further increased. Most portion of the response is optimized; the position curve of each is a desirable "ramp type" curve.

Figure 5-10 illustrates the system's ability to resist parameter change. Tm (or the total motor inertia) is several times larger while the system still has the similar response as before.

Some interesting examples are shown by Figure 5-11 and 5-12. In Figure 5-11, the nonlinear function used to eliminate the current spikes are removed from both the simulated and real-time systems. Minor position vibrations occur in both systems. In Figure 5-12 the controller's zero is removed. Oscillations occur in both systems. Figure 5-13

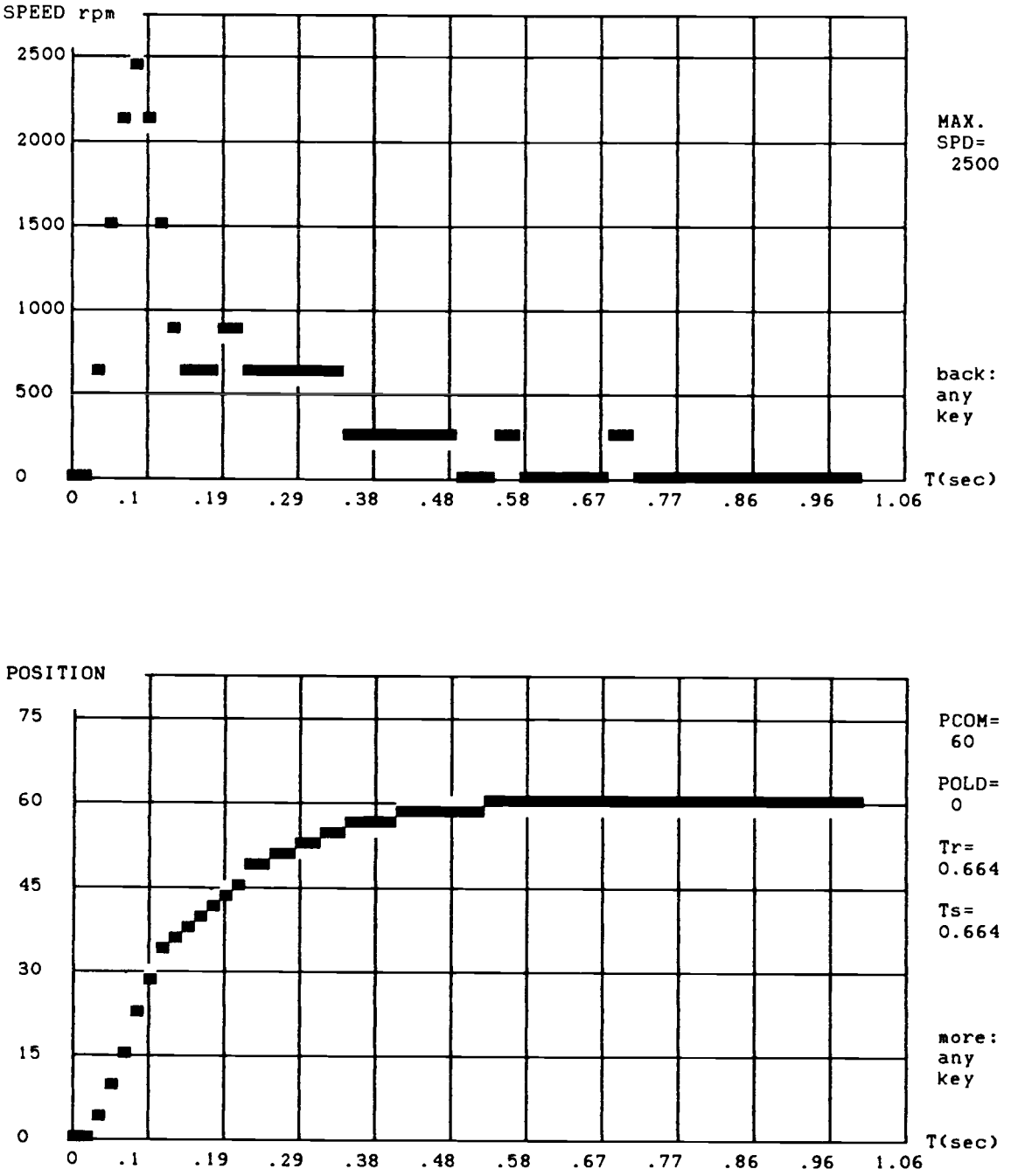


Figure 5-7(a) Real-Time System: PCOM=60

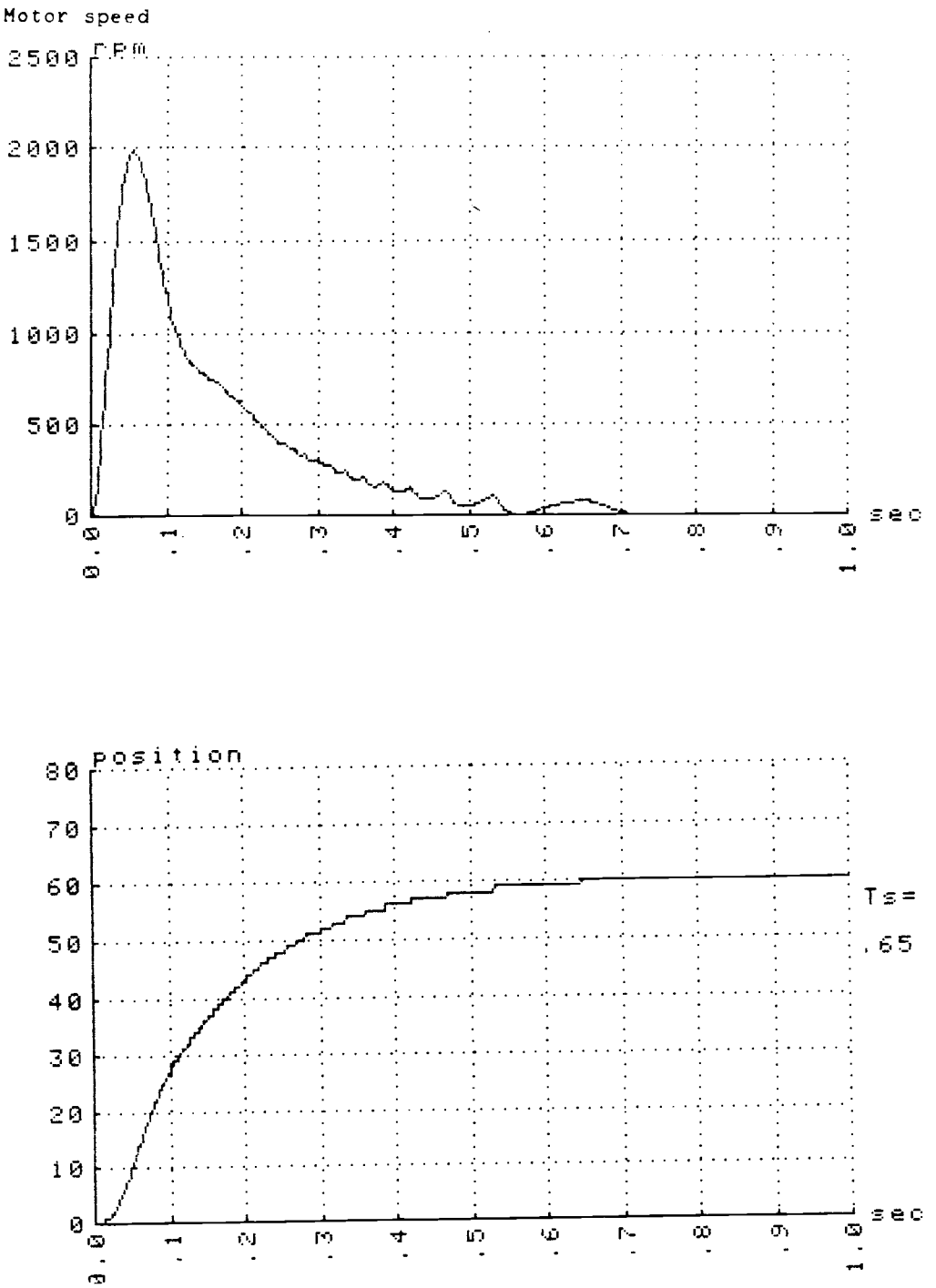


Figure 5-7(b) Simulated system: PCOM=60

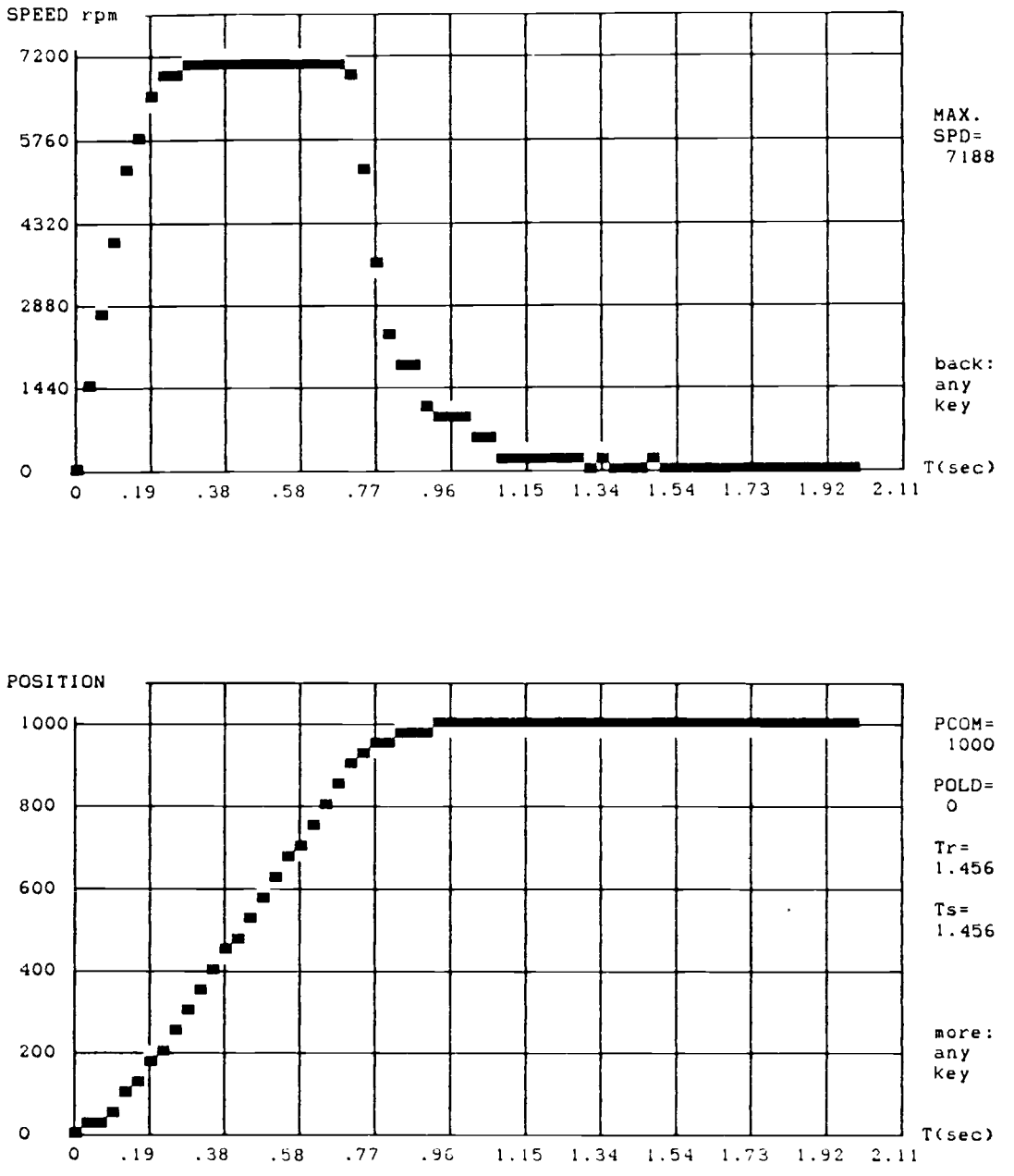


Figure 5-8(a) Real-time system: PCOM=1000

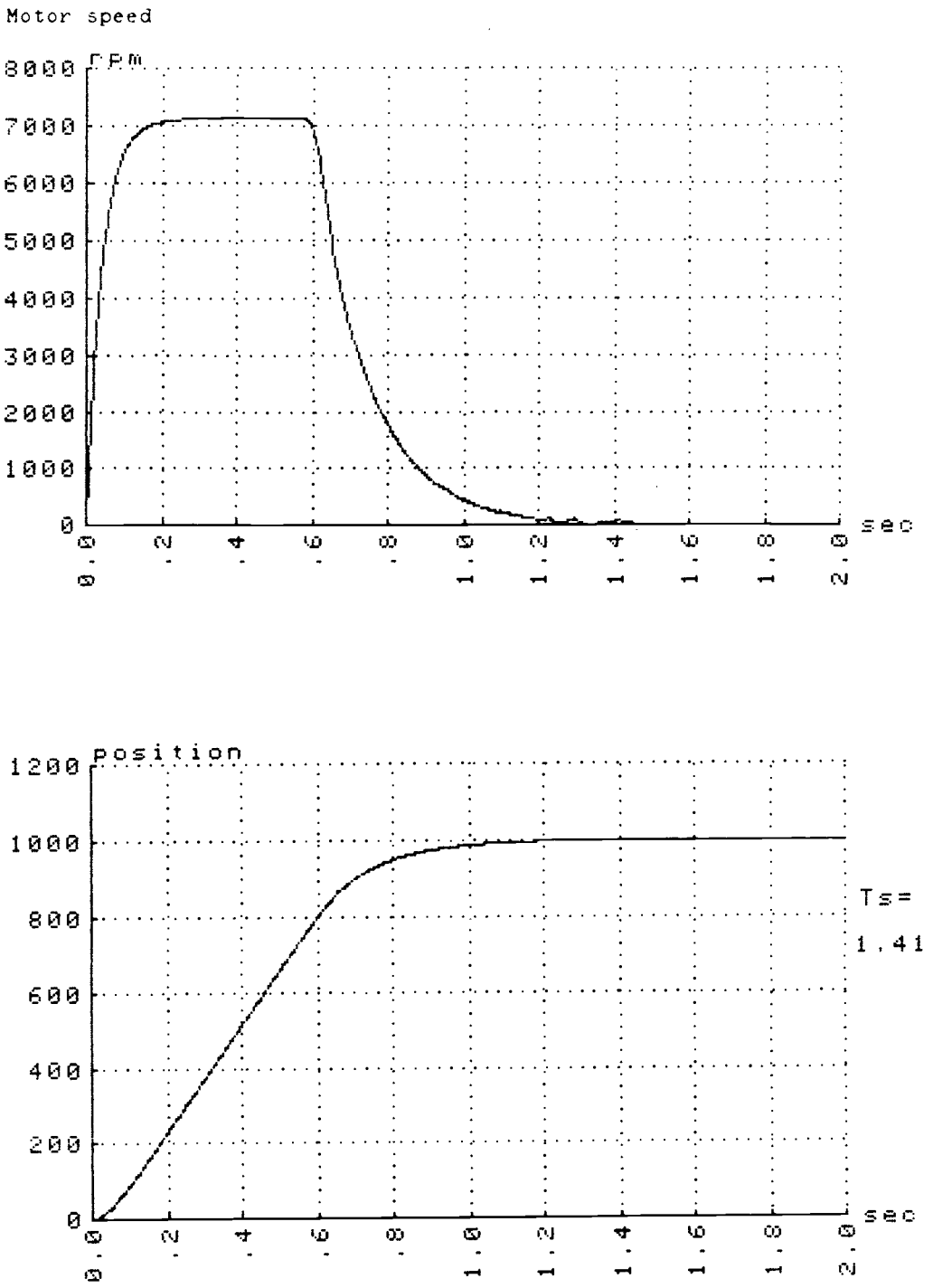
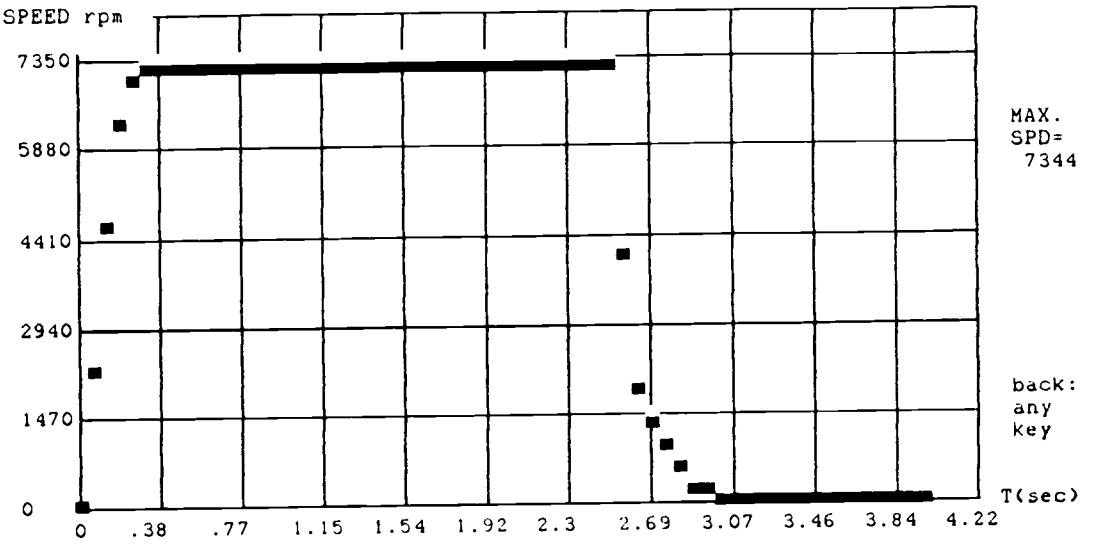


Figure 5-8(b) Simulated system: PCOM=1000



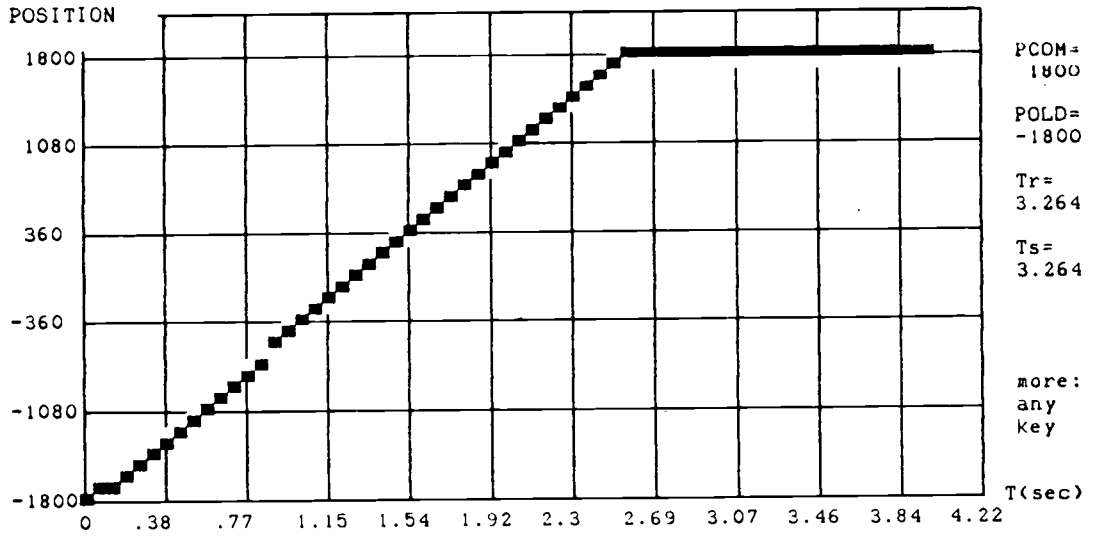


Figure 5-9(a) Real-time system: PCOM=1800, POLD=-1800

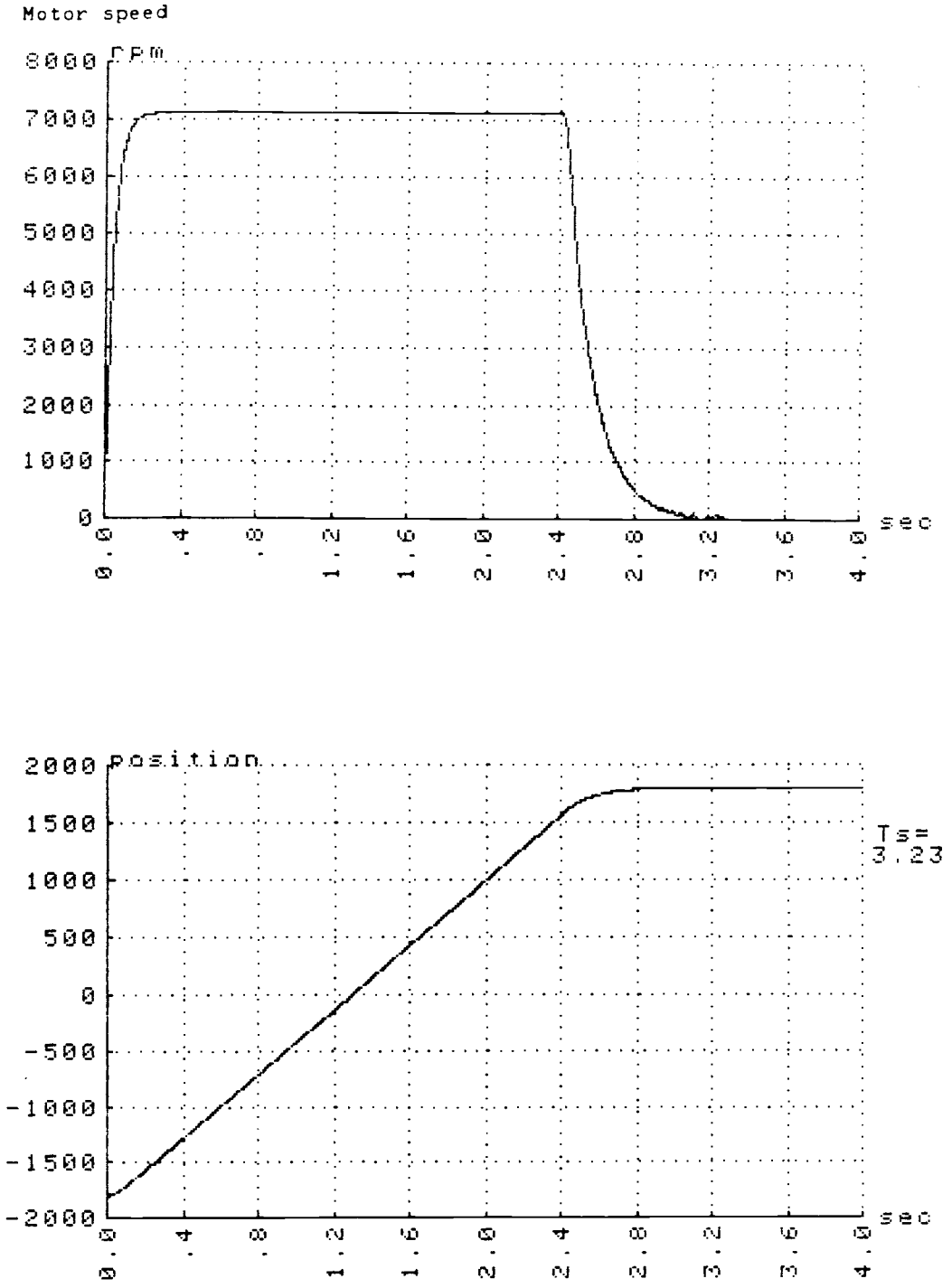


Figure 5-9(b) Simulated system: PCOM=1800, POLD=-1800

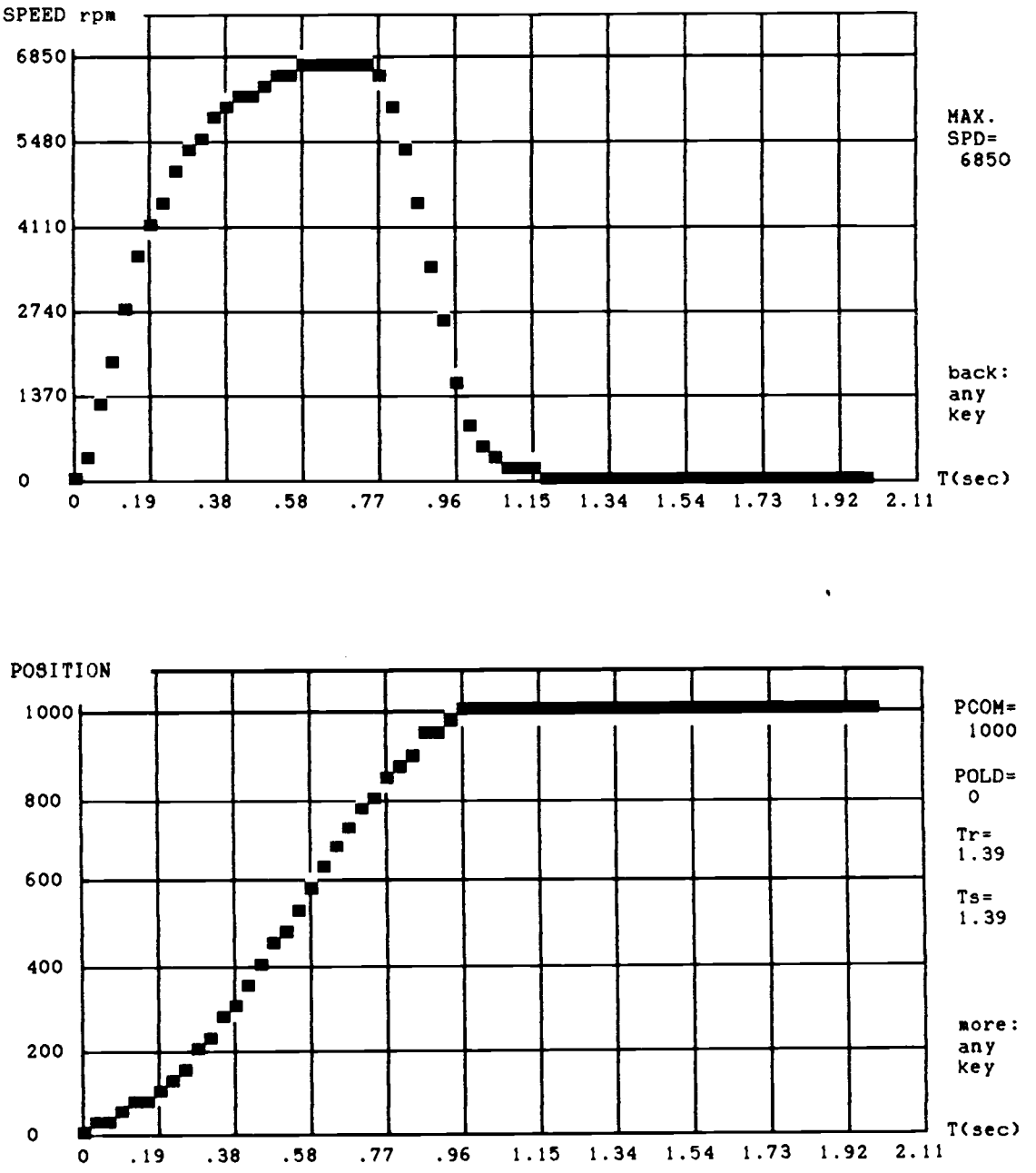


Figure 5-10(a) Real Time System: Increased Inertia

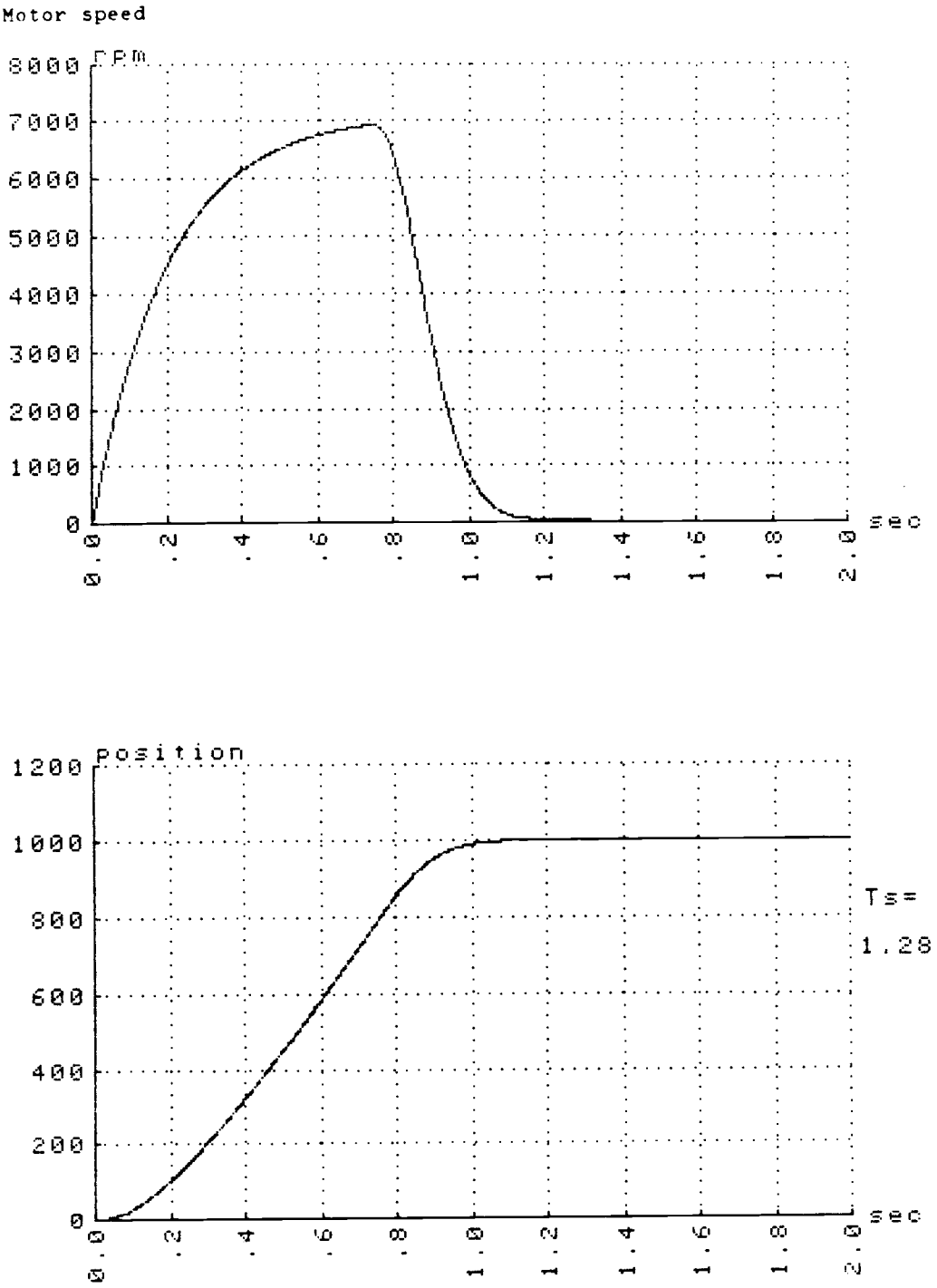


Figure 5-10(b) Simulated system: increased inertia

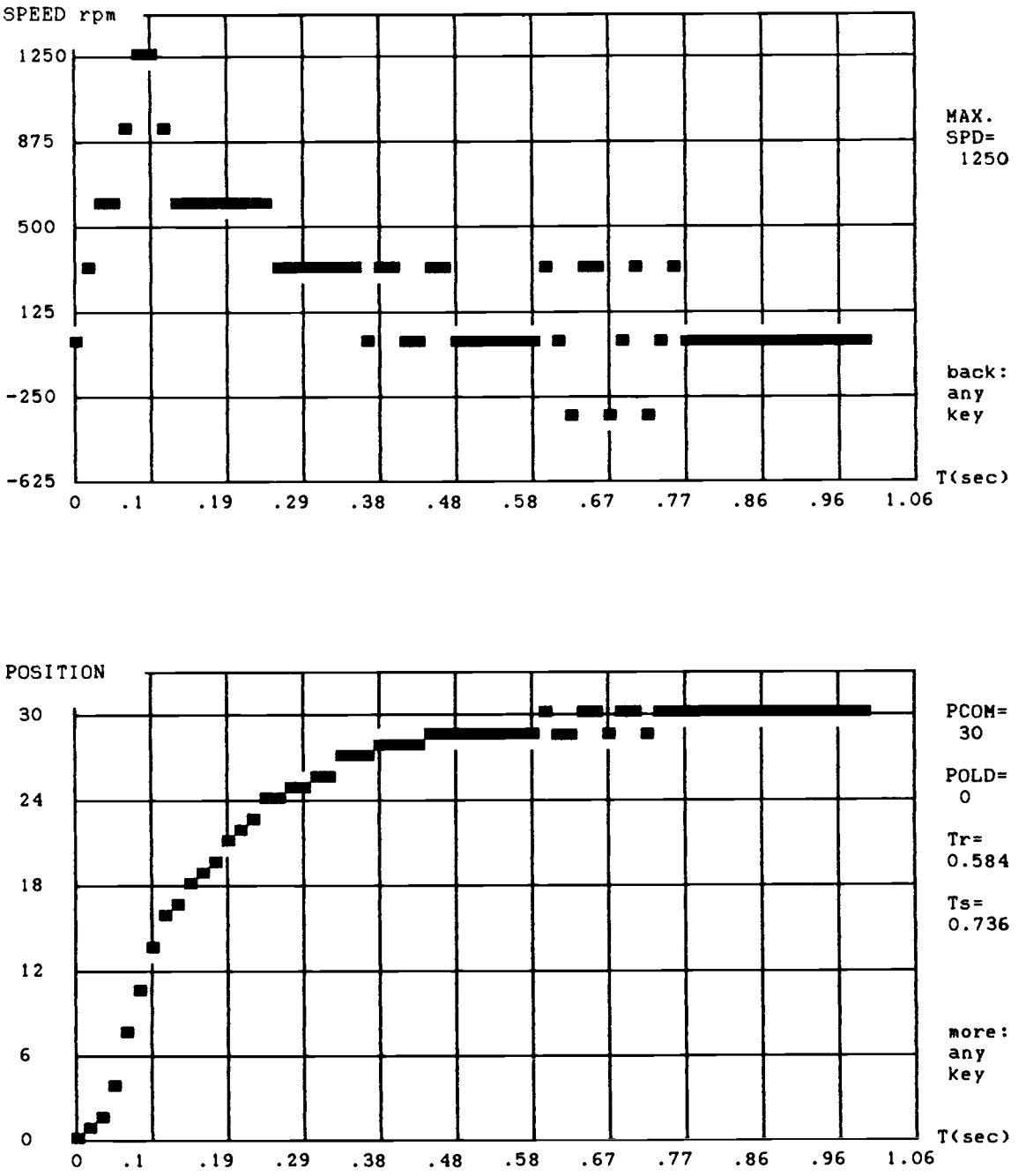
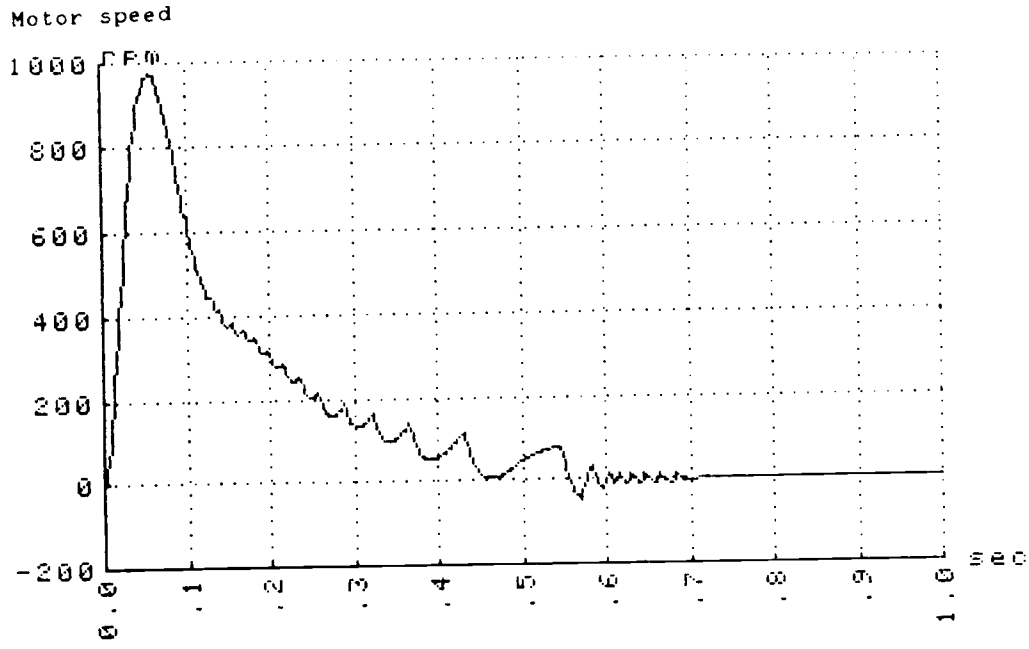


Figure 5-11(a) Real-Time System: P-PD Switch Removed



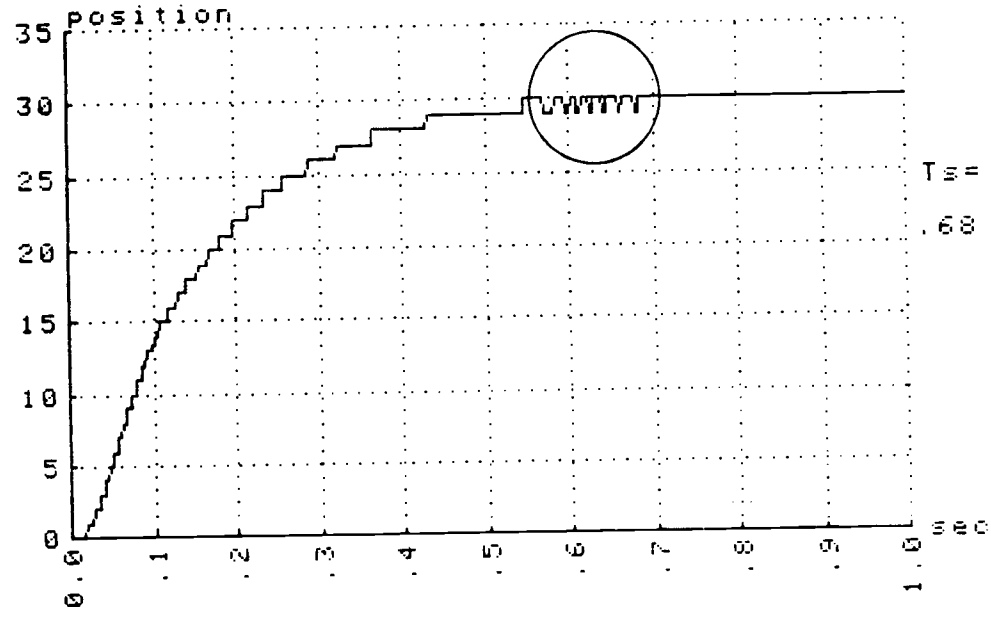


Figure 5-11(b) Simulated system:

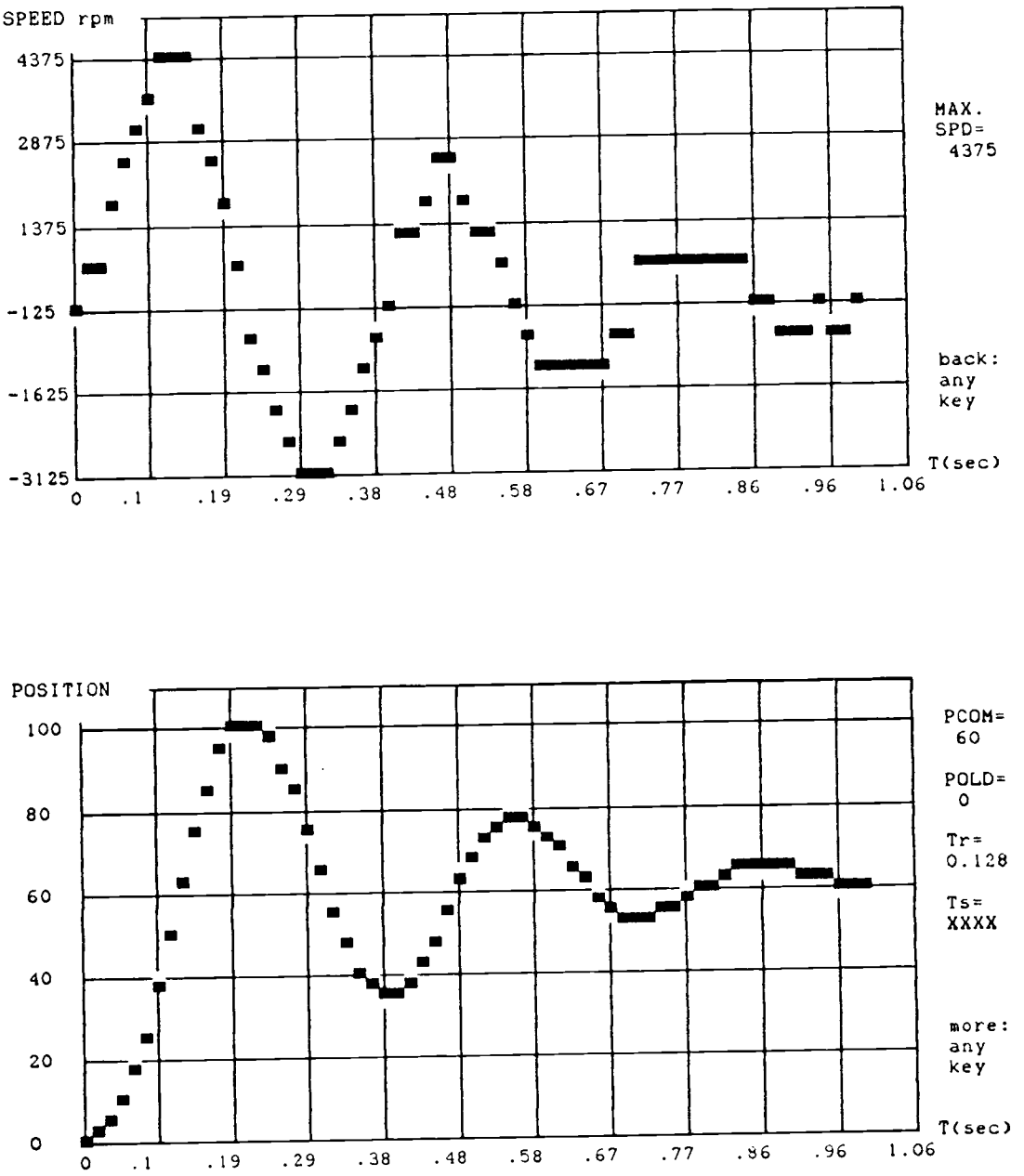


Figure 5-12(a) Real-time system: S+Z removed from controller

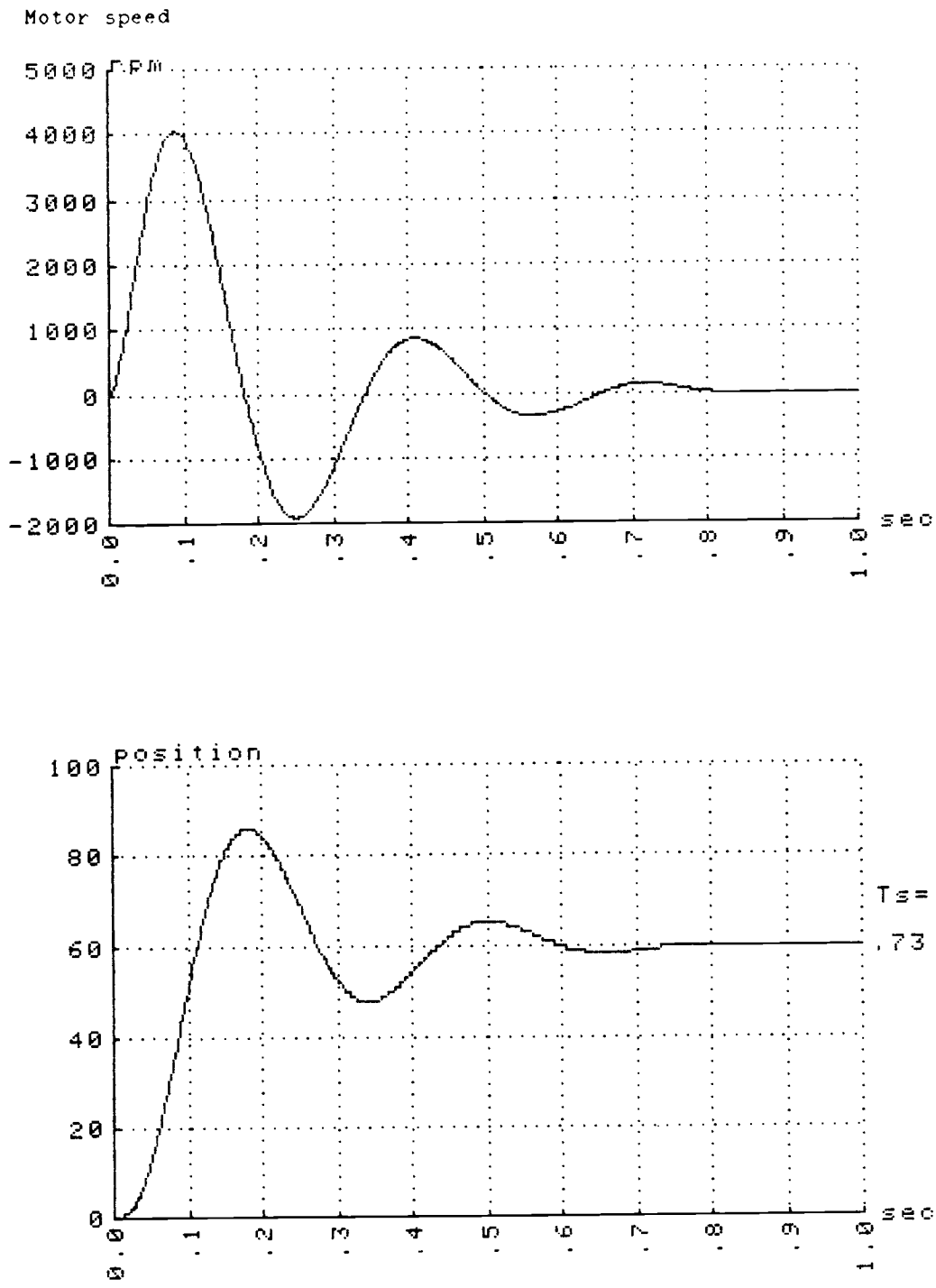


Figure 5-12(b) Simulated system: S+Z removed from controller

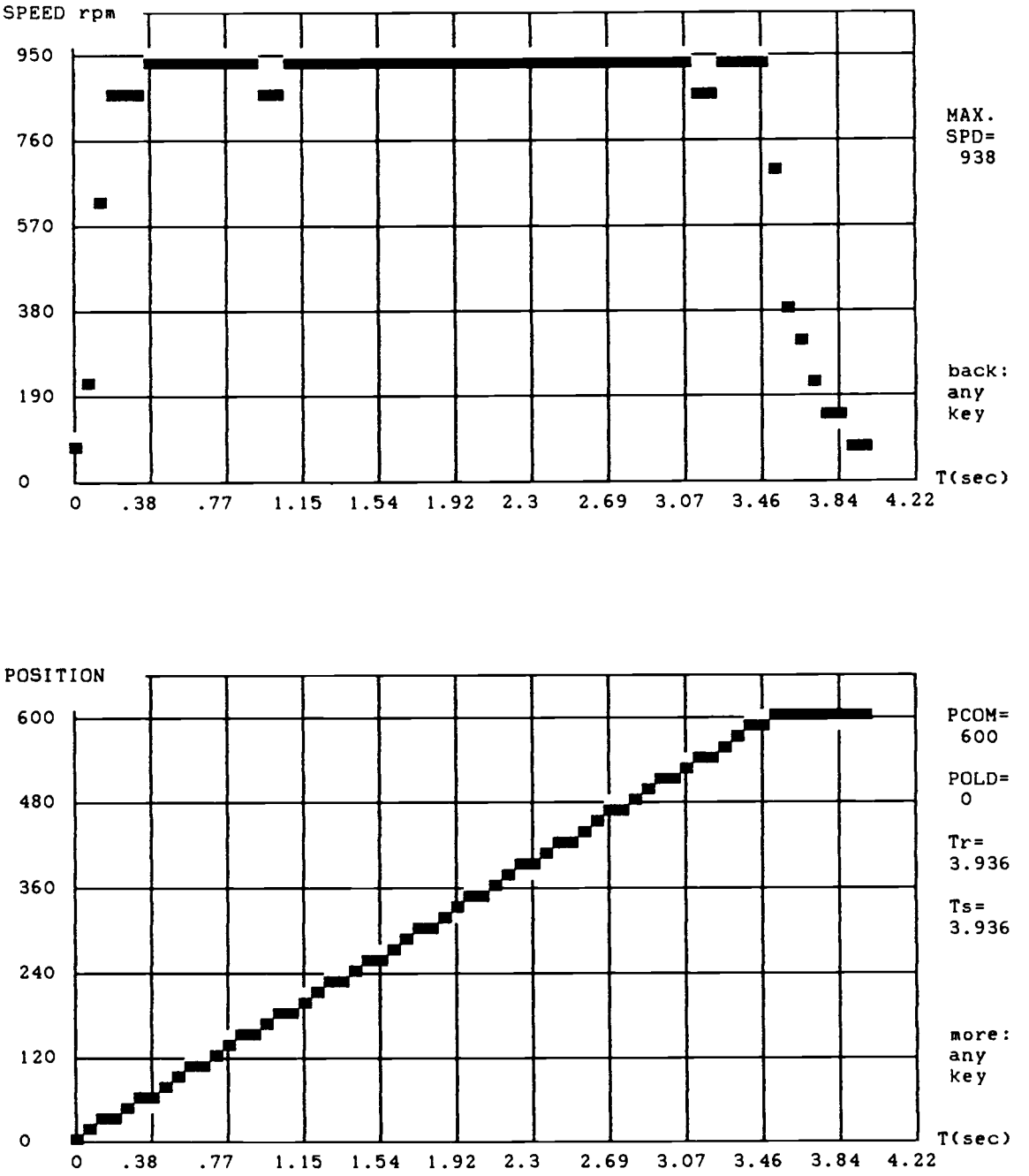


Figure 5-13(a) Real-Time System: Operating At Low Speed

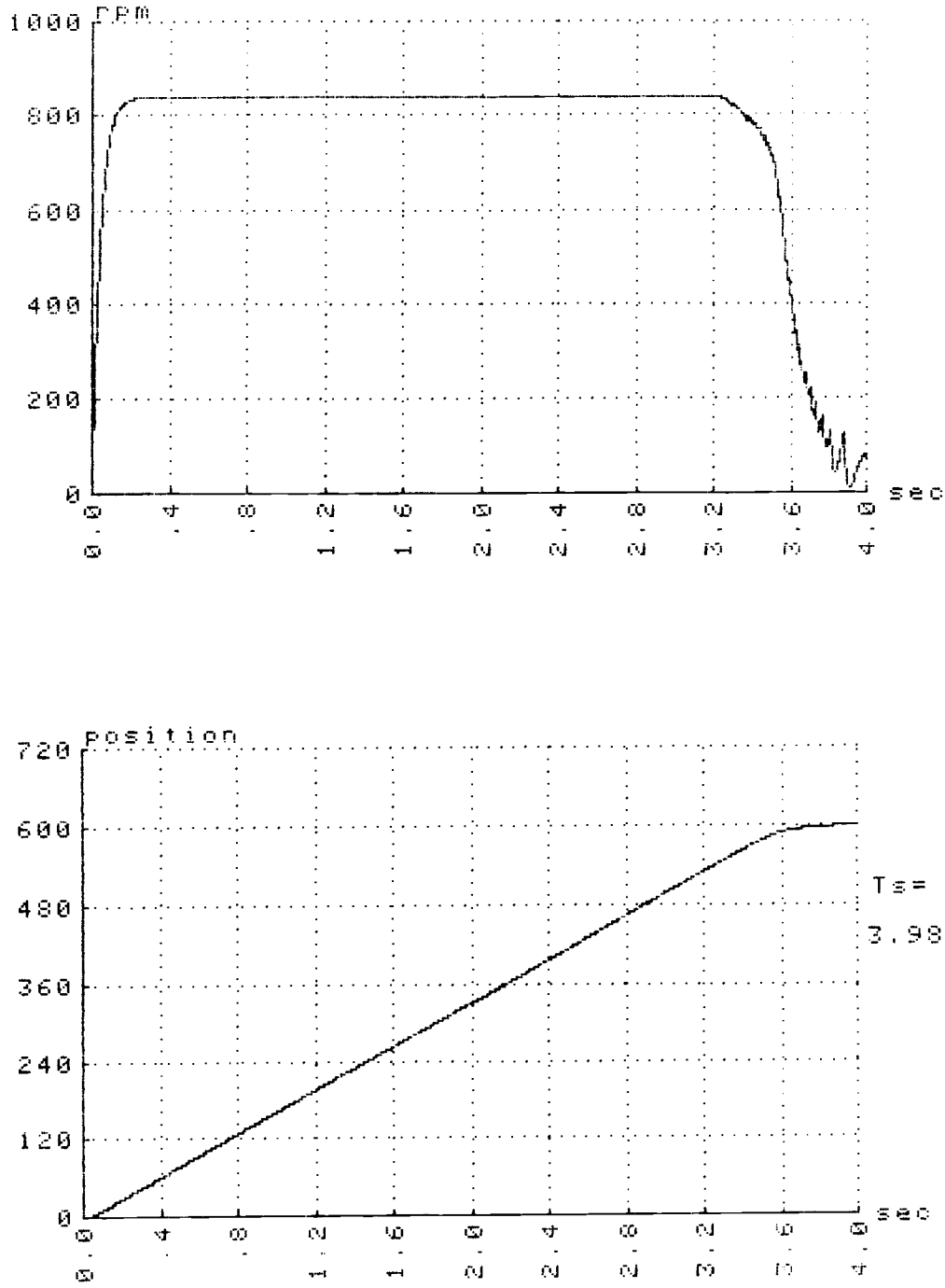


Figure 5-13(b) Simulated System: Operating At Low Speed

illustrates the speed control by using different voltage limit. A small Umax is used here and the motor speed is kept very low.

5.3 Summary

The real-time digital position control experiment is done with a microprocessor controller, using the system model developed in the previous chapters.

The experiment shows that the real-time system has the same response to position input as the simulated system. It presents all the desired characteristics such as fast response, insensitivity to inertia change and following the position command without overshoot.

6. CONCLUSION

From the results of this study, the following conclusions are drawn:

1. A non-overshooting, inertia-insensitive position control system can be accomplished by using properly designed leading-lagging (PD-Lowpass) functions in the controller.

2. Fast response can be achieved by using nonlinear functions which make the motor power (nearly) fully utilized.

3. The accuracy of the position control depends on the length of the positon word; i.e., the more bits there are in the position counter, the higher the accuracy is.

4. By using programmable digital controller a control system is able to operate hierarchically. The host computer is free from actual control activities yet has easy access to all of the system parameters when needed.

5. System modeling and simulation is an efficient way to develop a real time digital control system. Most design work can be done at the simulation stage in which all system parameters are determined before the real time experiment.

BIBLIOGRAPHY

Schwarzenbach, J. and Gill, K. F.: System Modelling and Control, Edward Arnold Ltd., 1984

Jacquot, Raymond G.: Modern Digital Control Systems, Marcel Dekker, Inc., 1981

Towill, D. R.: Transfer Function Techniques for Control Engineers, Iliffe Books Ltd., 1970

Lurie, B. J. : Feedback Maximization, Artech House, Inc., 1986

Morris M. Henry: "Reliable Position Sensing a Must for Advanced Automation Techniques", Control Engineering, November 1984, Volume 31, No. 11, p.62

Miller, Timothy J.: "Digital Positioning Trade Hardware For Software", Control Engineering, August 1982, Volume 29, No. 8, p.59

Morris M. Henry: "Robotic Servo Control Need Accurate Position Feedback Inputs", Control Engineering, January 1984, Volume 31, No. 1, p.90

APPENDICES

APPENDIX I

DEFINITIONS OF MOTOR PARAMETERS

1. Mechanical Time Constant Tm: the major time constant of a d.c. servo motor. It reflects the delay caused by mechanical inertia. The value of Tm is proportional to the total mechanical inertia on the motor shaft. If the motor is connected to a mechanical mass via a gear box, the total inertia is then (motor armature inertia) + W (external inertia), where W is the gear ratio, normally less than 1.

2. Electrial Time Constant Te: A circuit time constant, introduced by the armature stray inductance (L) and the armature resistance (Rt). The value of Te is equal to L/Rt and is usually smaller than Tm. It is also a delay factor in the system.

3. Voltage Constant Ke: electromotive potential a d.c. motor can generate at unity speed. It can be roughly calculated by Ke = (rated armature voltage)/(rated motor speed).

4. Armature Electromotive Potential E: the potential generated by the armature windings when they rotate in the magnetic field. E is proportional to motor speed N and is equal to Ke N.

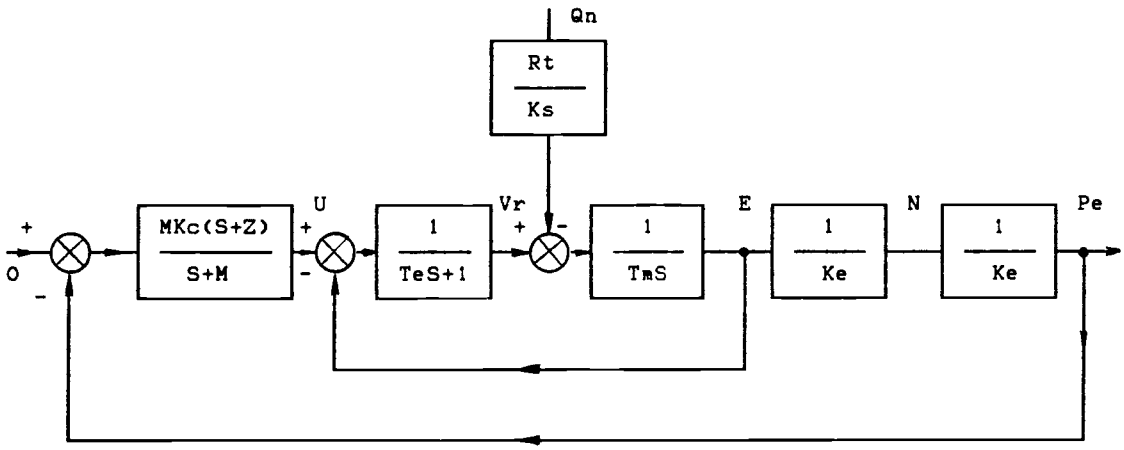
5. Armature Current Ia: Ia represents the load of motor and is proportional to the motor torque.

6. Armature Voltage Drop Vr: Vr is related to Ia by Vr = Ia Rt. It is used in the motor's model as a branch voltage in the armature circuit loop.

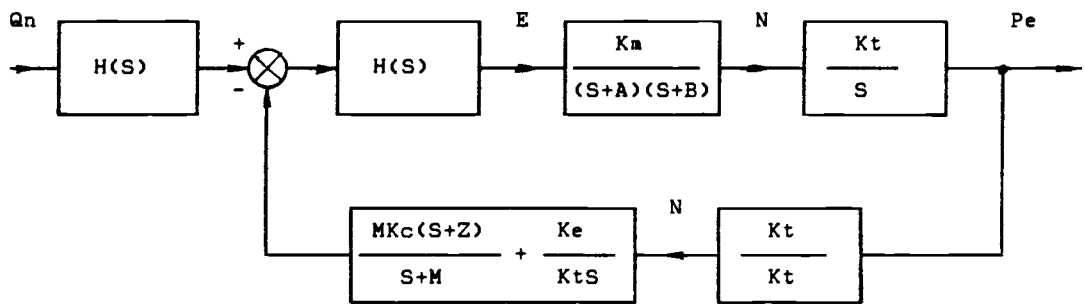
APPENDIX II

STEADY-STATE ERROR WITH CONSTANT TORQUE

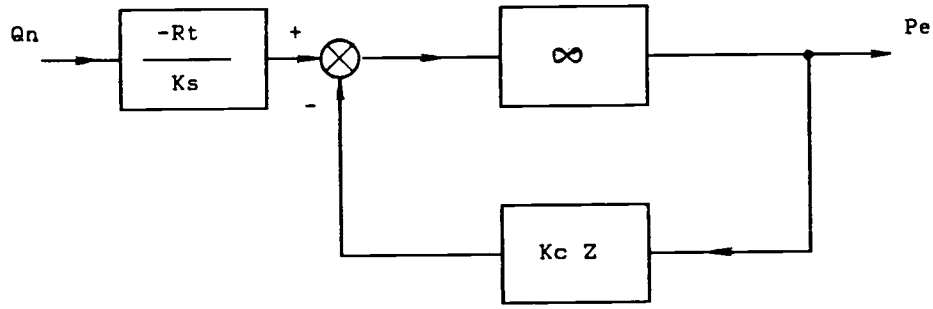
A position control system using the PD controller can have steady-state position error when constant torque is applied. Assume the net external torque is Qn. The system with Qn is shown below:



It can be redrawn as:



Let S $\rightarrow$ 0 to simulate the steady state, the following diagram is obtained:



Thus we obtain the equation

$$Pe = - \frac{Rt}{Ks\ Kc\ Z} Qn$$

where Pe is the position error.

The maximum torque without causing the position error can be calculated by

$$Qn(max) = (Qm+Qg)/W$$

where Qm+Qg is the total friction torque of the motor and the gear box, W is the speed reductuion ratio.

APPENDIX III

HP-87 BASIC SIMULATION PROGRAM

```
10  ! ************************************************
20  ! **   DIGITAL POSITION CONTROL SYSTEM SIMULATION   **
30  ! ************************************************
40  !                    MAY 1986
50  !
60  ! ******* PLOTTER PARAMS ********
70  IMAX=20   @ IMIN=-5 @ ISPC=5
80  NMAX=8000 @ NMIN=0  @ NSPC=1000
90  PMAX=1200 @ PMIN=0  @ PSPC=200
100 ! ******************************

120 INTEGER N(2500),P(2500),P,PCOM,PP,SCAL,C0,C1,C2,U,U1,I
125 DIM M$[50]
130 DELT=.002      ! system's sampling/control cycle
140 TE=.002        ! motor's electrical time constant (sec)
150 TM=.041        ! motor's mechanical time constant (sec)
160 KE=.0042       ! motor's voltage constant (volt/rpm)
170 RT=2.4         ! motor's armature resistance (ohm)
180 VF=.1          ! voltage needed to overcome mechanical
                   ! friction
200 SCAL=32        ! controller's scaling factor
210 UMAX=30        ! saturation threshold for motor voltage
220 Z=8.33         ! controller's zero
230 M=31.5         ! lowpass filter's pole
240 K=.1           ! proportional to system gain
250 KT=.2          ! position transducer gain
260 K1=K/KT        ! controller gain
270 C0=1/(DELT*Z)  ! differential control coefficient
280 C1=SCAL*K1     ! proportional control coefficient
290 C2=1/(DELT*M)  ! lowpass filter parameter

310 DISP "PCOM=";
320 INPUT PCOM
330 DISP "DURATION(sec, max=5) ="
340 INPUT TT
345 DISP "MESSAGE TO PRINT";
346 INPUT M$
350 NSAM=TT/DELT
360 A=DELT/TE
370 B=DELT/TM
380 C=DELT*KT

400 VR0=0
410 E0=0
420 P0=0
430 VR=VR0
440 IA(0)=VR0/RT
```

```
450 E=E0
460 N(0),N,PN=E0/KE
470 P,PP,P(0),PCONT=P0
480 U=0

500 GCLEAR
510 LOCATE 18,150,12,92
520 FXD 1,0
530 SCALE 0,TT,IMIN,IMAX
540 LAXES TT/10,ISPC,0,IMIN
550 LINE TYPE 3
560 GRID TT/10,ISPC,0,IMIN
570 LINE TYPE 1
580 MOVE 0,0

600 !************* CONTROLLER **************
610 FOR I=1 TO NSAM
620 IF P=PCOM THEN U1=0 @ GOTO 640          ! NDC(PD switch)
630 U1=C1*(PCOM-P+C0*(PP-P))
635 IF U1>U THEN U=U-INT((U-U1)/C2) @ GOTO 650
640 U=U+INT((U1-U)/C2)           ! smoothing & NDC(dead zone)
650 IF U>=UMAX*CSAL THEN U=UMAX*SCAL       ! NDC(saturation)
660 IF U<=-(UMAX*SCAL) THEN U=-(UMAX*SCAL)
670 UX=U/SCAL
690 GOSUB MOTOR
690 PP=P
700 PCONT=C*N+PCONT      ! PCONT is the continuous position
710 P=PCONT              ! sampling & quatization
720 P(I)=P
730 IF P<>PCOM THEN TS=DELT*(I+1)
740 NEXT I
750 !*****************************************

770 TS=.01*INT(100*TS+.5)
780 MOVE DELT*.005*NSAM,1.04*IMAX
790 LABEL "Ia"
800 MOVE DELT*1.025*NSAM,IMIN
810 LABEL "sec"
820 PRINTER IS 701
830 DUMP GRAPHICS 0,0,0,1
840 GCLEAR
850 SCALE 0,TT,NMIN,NMAX
860 LAXES TT/10,NSPC,0,NMIN
870 LINE TYPE 3
880 GRID TT/10,NSPC,0,NMIN
890 LINE TYPE 1
900 MOVE 0,0
910 FOR I=0 TO NSAM STEP 1
920 PLOT DELT*I,N(I)
930 NEXT I
940 MOVE DELT*.005*NSAM,1.025*NMAX
950 LABEL "rpm"
960 MOVE DELT*1.03*NSAM,NMIN
```

```
970 LABEL "sec"
980 DUMP GRAPHICS 0,0,0,1
990 GCLEAR
1000 SCALE 0,TT,PMIN,PMAX
1010 LAXES TT/10,PSPC,0,PMIN
1020 LINE TYPE 3
1030 GRID TT/10,PSPC,0,PMIN
1040 LINE TYPE 1
1050 MOVE 0,0
1060 FOR I=0 TO NSAM STEP 1
1070 PLOT DELT*I,P(I)
1080 NEXT I
1090 MOVE DELT*.005*NSAM,1.03*PMAX
1100 LABEL "position"
1110 MOVE DELT*1.025*NSAM,PMIN
1120 LABEL "sec"
1130 MOVE DELT*1.02*NSAM,.7*PMAX
1140 LABEL "Ts="
1150 MOVE DELT*.99*NSAM,.6*PMAX
1160 LABEL TS
1170 DUMP GRAPHICS 0,0,0,1
1180 PRINT @ PRINT @ PRINT M$
1190 END

1250 !********** MOTOR ***********
1260 MOTOR: VR=A*(UX-E-VR)+VR      ! VR is voltage across Rt
1270 PLOT DELT*I,VR/RT
1280 ! * FRICTION MODEL, F=elec. torque; f=friction *
1290 IF E<>0 THEN GOTO ROTATING          ! jump if speed <>0
1300 IF VR>VF THEN E=B*(VR-VF) @ GOTO 1400     !stop if F>f
1310 IF VR<-VF THEN E=B*(VR+VF) @ GOTO 1400    !stop if F>f
1320 E=0 @ GOTO 1400                           !stop if F<f
1330 ROTATING:IF E>0 THEN GOTO 1370            ! rotating:
1340 E=B*(VR+VF)+E                  ! backward & f against F
1350 IF E>0 THEN E=0 @ GOTO 1400    ! may cause stop
1360 GOTO 1400
1370 E=B*(VR-VF)+E                  ! forward & f against F
1380 IF E<0 THEN E=0                ! may cause stop
1390 !*
1400 PN=N
1410 N=E/KE                        ! N is motor speed in rpm
1420 N(I)=N
1430 RETURN
```

APPENDIX IV

8039 ASSEMBLY LANGUAGE CONTROLLER PROGRAM

```
TASK20:  NOP

**********************************************************
**                                                      **
**   POCO                                  Jan. 1987    **
**   Real Time d.c. Motor Position Control              **
**   Controller: PDVL                                   **
**   (Proportional/diffrential/lowpass/voltage limit    **
**                                                      **
**********************************************************

RDNCNT  EQU 5EH          ; DONE count,256-DNC=count #
RLOB    EQU 5FH          ; a temp. storage
RPHOST  EQU 60H          ; 2-byte host interface for PCOM
RPCOM   EQU 62H          ; 2-byte PCOM buffer
RP      EQU 64H          ; 2-byte buffer for position input
RPP     EQU 66H          ; 2-byte storage for last position
RUMAX   EQU 68H          ; 2-byte constant
RC0     EQU 6AH          ; 1-byte constant
RC1     EQU 6BH          ; 1-byte constant
RC2     EQU 6CH          ; 1-byte constant
RWND    EQU 6DH          ; 256-wnd=# P bytes sent to host
RTIP    EQU 6EH          ; 256-tip=interval sending P bytes
RTPCNT  EQU 6FH          ; interval counter
TPST    EQU 0ECH         ; timer preset:interval, 1954 usec
DNC     EQU 0F6H         ; 256-DNC=times delayed before ack.

        DIS TCNTI        ; sorry,  no timer interrupt.
        EN  I            ; allow the op sys to receive
                         ; and  process  host command.

**
**      initializations and constants set-up
**

        MOV R1, #RC0
        MOV @R1, #60     ;Co=1/(Delt*Z)=60
        MOV R1, #RC1
        MOV @R1, #4      ;C1=log2(16000/1000)=4
        MOV R1, #RC2
        MOV @R1, #4      ;C2=log2(1/(Delt*M))=4

        MOV R1, #RUMAX
        MOV @R1, #0C0H
        INC R1
        MOV @R1, #03H    ;UMAX=30v*16/K1=960
```

```
        MOV R1, #RWND
        MOV @R1, #04H    ; 256-WND=252, # bytes sent to host
        MOV R1, #RTIP
        MOV @R1,#0FCH    ; 256-TIP=4, send every 4 loop cycle
        MOV R1, #RTPCNT
        MOV @R1, #0FFH   ; reset TPCNT
        CLR F0

        CLR A
        MOV R6, A
        MOV R7, A        ; R7/R6 reserved for U1

**
**      start of the 2 msec control loop
**

POC1:   STOP   TCNT      ; preset the timer
        MOV A, #TPST
        MOV T, A
        STRT T

**
**  take host position command and put in buffer
**
        MOV R1, #RPHOST ;input PCOM from host interface
        MOV R0, #RPCOM
        MOV A,  @R1
        MOV @R0, A       ; PHOST low byte in PCOM low byte
        INC R1
        INC R0
        MOV A, @R1       ; PHOST high byte in A
        NOP
        JB3 PHN          ; jump if negative position command
        ANL A, #0FH      ; positive, clear bit 8-11
        JMP PH1
PHN:    ORL A, #0F0H     ; set bit 8-11 to make a 16 bit -#
PH1:    MOV @R0, A       ; PHOST in PCOM

**
**    input motor position from PPI port A and B
**
INPO:   MOV R1, #0E0H    ;input P from port E0/E1
        MOV R0, #RP
        MOVX A, @R1
        MOV @R0, A
        INC R1
        INC R0
        MOVX A, @R1
        JB7 VALID        ; counters not busy
        JMP INPO         ; counter busy, input again
VALID:  JB3 BKWD         ; the 12-bit position is neg
        ANL A, #0FH      ; pos, remove 4 msb
        JMP POC2
```

```
BKWD:   ORL A, #0F0H     ; neg, set 4 msb
POC2:   MOV @R0, A       ; motor position in P

**
**       This section controls the sending of position words
**   to the host.Each position word has 2 8-bit bytes. These
**   position bytes are sent every 256-TIP loop cycles.  The
**   host initiates the sending by setting bit 15 of the
**   position command (PHOST). A total number of 256-WND
**   position bytes are sent.
**
        MOV R1, #RPHOST+1
        MOV A, @R1       ; PHOST hi byte in A
        NOP
        JB7 SPST         ; go send if S=1 (bit15)
        JMP SELD         ; jump if S=0

SPST:   MOV R0, #RTPCNT  ; R0 points to TPCNT
        INC @R0          ; increment TPCNT
        MOV A, @R0       ; TPCNT in A
        JNZ SELD         ; <>0,don't send

SNDP:   MOV R1, #RTIP    ; R1 points to TIP
        MOV A, @R1       ; TIP in A
        MOV @R0, A       ; TIP in TPCNT
        MOV R1, #RWND    ; R1 points to WND
        INC @R1          ; inc WND
        MOV A, @R1       ; WND in A
        JNZ SDP1         ; less than 252 bytes sent, continue
        MOV @R1, #04H    ; sending complete: restore WND
        MOV @R0, #0FFH   ; reset TPCNT
        MOV R1,#RPHOST+1
        MOV A, @R1       ; PHOST hi byte in A
        ANL A, #7FH      ; kill S
        MOV @R1, A       ; put back
**
**   this section sends P high and P low alternatively
**

SDP1:   JF0 SLOB         ; F0=1:send low byte
        MOV R1, #RP+1    ; F0=0:send high byte
        MOV A, @R1       ; P high byte in A
        CALL DUMP        ; send it to host
        MOV R1, #RP
        MOV A, @R1       ; P low byte in A
        MOV R1, #RLOB
        MOV @R1, A       ; P low byte in LOB
        CPL F0
        JMP SELD
SLOB:   MOV R1, #RLOB
        MOV A, @R1       ; P low byte in A
        CALL DUMP        ; send it to host
        CLR F0
```

```
**
**   here is  our controller.  the following section removes
**   differential  computation if no position error detected.
**   a "done" acknowledgement is sent to host if so required
**

SELD:   MOV R1, #RPCOM  ; if P=PCOM then no PD computation
        MOV R0, #RP
        MOV A, @R1
        XRL A, @R0
        JNZ CMPT        ; P<>PCOM
        INC R1
        INC R0
        MOV A, @R1
        XRL A, @R0
        JNZ CMPT        ; P<>PCOM

        MOV R1,#RPHOST+1 ;no error detected
        MOV A, @R1      ; PHOST high byte in A
        NOP
        JB6 WT          ; if ACK then check DNCNT
        JMP NOD         ; if ACK=0 then forget ack.
WT:     MOV R0,#RDNCNT  ; test if no error for 10 times
        MOV A,@R0       ; DNCNT in A
        JZ KILA         ; if overflow go send ack.& kill ACK
        INC @R0         ; if not overflow inc DNCNT
        JMP NOD

KILA:   MOV @R0,#DNC    ; preset DNCNT
        MOV A,@R1       ; PHOST high in A
        ANL A, #0BFH    ; kill ACK
        MOV @R1, A      ; put back
        MOV A,#0E7H     ; pass #
        CALL DUMP       ; send pass # to host for ack.
NOD:    CLR A
        MOV R3, A
        MOV R2, A
        JMP LOPAS       ; skip PD section

**
**     this portion does linear PD computation
**

CMPT:   MOV R1,#RDNCNT  ; preset DONE counter when P<>PCOM
        MOV @R1,#DNC

        MOV R1, #RPP    ;U2=2^C1*[(PCOM-P)+Co*(PP-P)]
        MOV A, @R1      ;  R3/R2<--PP
        MOV R2, A
        INC R1
        MOV A, @R1
        MOV R3, A       ;  PP in R3/R2
        MOV R1, #RP     ;  R5/R4<--P
```

```
        MOV A, @R1
        MOV R4, A
        INC R1
        MOV A, @R1
        MOV R5, A          ;  P in R5/R4
        CALL DMIN          ;  PP-P in R3/R2
        MOV R1, #RCO
        MOV A, @R1
        MOV R4, A          ;  CO in R4
        CALL HMLT          ;  CO*(P-PP) in R3/R2

        MOV R1, #RPCOM     ;  do Co(P-PP)+PCOM
        MOV A, @R1
        MOV R4, A
        INC R1
        MOV A, @R1
        MOV R5, A          ;  PCOM in R5/R4
        CALL DADD          ;  PCOM+C1(P-PP) in R3/R2

        MOV R1, #RP        ;  do  above-P
        MOV A, @R1
        MOV R4, A
        INC R1
        MOV A, @R1
        MOV R5, A          ;  P in R5/R4
        CALL DMIN          ;  PCOM-P+C1(P-PP) in R3/R2

        MOV R1, #RC1       ;  do  above*2^C1
        MOV A, @R1         ;  C1 in A
        CALL DLSH          ;U2 in R3/R2

LOPAS:  MOV A, R6          ;U1=U1+(2^-C2)*(U2-U1)
        MOV R4, A
        MOV A, R7
        MOV R5, A          ;  U1 in R5/R4
        CALL DMIN          ;  U2-U1 in R3/R2
        MOV R1, #RC2       ;  do (2^-C2)*above
        MOV A, @R1         ;  C2 inA
        CALL DRSH          ;  (2^-C2)*(U2-U1) in R3/R2
        MOV A, R6
        MOV R4, A
        MOV A, R7
        MOV R5, A          ;  U1 in R5/R4
        CALL DADD          ;U1new in R3/R2
        MOV A, R2
        MOV R6, A
        MOV A, R3
        MOV R7, A          ;U1 new  saved in R7/R6 for n+1 time
**
**    Saturating the output voltage if the absolute
**    value  of the linearly calculated  voltage is
**    beyond the bound specified by UMAX
**
```

```
        MOV R1, #RUMAX
        MOV A, @R1
        MOV R4, A
        INC R1
        MOV A, @R1
        MOV R5, A          ;  UMAX in R5/R4
        CALL DMIN          ;  U1-UMAX in R3/R2 for compare
        JB7 COMPLO         ;  if U1<UMAX then jump
        MOV A, R4
        MOV R2, A
        MOV A, R5
        MOV R3, A          ;U=UMAX in R3/R2
        JMP UOUT
COMPLO: MOV A, R6
        MOV R2, A
        MOV A, R7
        MOV R3, A          ;  U1 in R3/R4
        CALL DADD          ;  U1+UMAX in R3/R4
        JB7 LOLIM          ;  if U1<-UMAX then jump
        MOV A, R6
        MOV R2, A
        MOV A, R7
        MOV R3, A          ;U=U1 in R3/R2
        JMP UOUT
LOLIM:  CLR A
        MOV R2, A
        MOV R3, A
        CALL DMIN          ;U=-UMAX in R3/R2
**
**   this portion forms the appropriate code
**   and outputs the voltage to PPI port C
**
UOUT:   MOV A, #3          ;do U/8 + 128 to fit 8-bit D/A
        CALL DRSH          ; U/8 in R3/R2
        MOV A, R2
        ADD A, #80H        ; U/8 + 128 lower in A
        MOV R1, #0E2H      ; Port C address
        MOVX @R1, A        ; U/8+128 in Port C
**
**   update the past position with the current position
**
UPDT:   MOV R0, #RP
        MOV R1, #RPP
        MOV A, @R0
        MOV @R1, A
        INC R0
        INC R1
        MOV A, @R0
        MOV @R1, A         ;P in PP

**
**   idle til time due
**
```

```
IDL:    JTF REL
        JMP IDL
REL:    JMP POC1        ; end of contol loop, start again


**
**   POCO computing subroutines
**

DADD:   MOV A, R2       ;16-bit addition
        ADD A, R4       ;R3/R2+R5/R4==>R3/R2
        MOV R2, A
        MOV A, R3
        ADDC A, R5
        MOV R3, A
        RET

DMIN:   MOV A, R2       ;16-bit subtraction
        CPL A           ;R3/R2-R5/R4=R3/R2
        ADD A, R4
        CPL A
        MOV R2, A
        MOV A, R3
        CPL A
        ADDC A, R5
        CPL A
        MOV R3, A
        RET

HMLT:   MOV A, #8       ;16-bit/8-bit multiplication
        MOV R0, A       ;(R3/R2)*R4==>R3/R2
        CLR A           ;negative multiplicant ok
        MOV R5, A       ;A,R0,R2,R3,R4,R5 used
        CLR C           ;result abs must<=7FFFH

HM1:    XCH A, R5
        RRC A
        XCH A, R5
        RRC A           ;R5/A right shifted, C=A0

        XCH A, R4
        RRC A
        XCH A, R4       ;R4 right shifted; C=LSB

        JNC HM2
        ADD A, R2
        XCH A, R5
        ADDC A, R3
        XCH A, R5

HM2:    DJNZ R0, HM1
        XCH A, R5       ;final shift
        RRC A
```

```
        XCH A, R5
        RRC A
        XCH A, R4
        RRC A
        MOV R2, A       ;result in R4/A
        XCH A, R4
        MOV R3, A       ;result in R3/R2
        RET

DLSH:   MOV R4, A       ;16-bit left shift
        MOV A, R3       ;negative number ok
        ANL A, #80H     ;mask 0-6; R3/R2=input data
        MOV R5, A       ;A=shift # (>=1)
        MOV A, R3       ;result in R3/R2
                        ;set counter
DL1:    XCH A, R2
        CLR C
        RLC A
        XCH A, R2
        RLC A           ;R3/R2 shifted
        DJNZ R4, DL1

        ANL A, #7FH     ;restore sign
        ORL A, R5
        MOV R3, A       ;result in R3/R2
        RET

DRSH:   MOV R4, A       ;16-bit right shift
        MOV A, R3       ;negative number ok
        JB7 DR2         ;R3/R2=data
        ORL A, R2       ;A=shift # (>=1)
        JNZ DR1         ;result in R3/R2
        RET

DR1 :   CALL DNEG       ;-R3/R2 in R3/R2
        MOV A,R3
        CALL DR2        ;shifted and rounded -
        CALL DNEG       ;now R3/R2 shifted & rounded +
        RET

DR2:    CLR C           ;vacancies filled with 1
        CPL C
        RRC
        XCH A, R2
        RRC A
        XCH A, R2
        DJNZ R4, DR2
        MOV R3, A       ; R3/R2 shifted & rounded-
        RET


DNEG:   MOV A, R3       ; R3/R2 <-- (-R3/R2)
        CPL A
```

```
        MOV R3, A
        MOV A, R2
        CPL A
        INC A
        MOV R2, A       ; -R3/R2 in R3/R2
        JZ CARRY
        RET
CARRY:  INC R3
        RET             ; -R3/R2 in R3/R2

DUMP:   MOV R1, A       ; send A to host
        MOV R0, #UCR
DMP1:   MOVX A, @R0
        RRC A
        JC DMP2
        JMP DMP1
DMP2:   MOV R0, #UDR
        MOV A, R1
        MOVX @R0, A
        RET

**

TASK21   EN I
**
**  host word to ram, arg1=start addr, arg2 & 3 = data
**
        MOV A, R2
        ADD A, #02H
        MOV R0, A
        MOV A, @R0      ; start address in A
        MOV R1, A       ; R1->ram start address
        INC R0          ; R0->arg1
        MOV A, @R0
        MOV @R1, A      ; arg1 in ram
        INC R0          ; R0->arg2
        INC R1          ; R1->next ram byte
        MOV A, @R0
        MOV @R1, A      ;arg2 in next ram
        DIS I
        RET

        END
```