MICROCOMPUTER-CONTROLLED SERVO SYSTEM
FOR AN ULTRASONIC
COMPUTER-ASSISTED TOMOGRAPHIC SCANNER

BY

BENJAMIN LERNER

B.S., University of Illinois, 1975

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science
in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1979

Urbana, Illinois

UNIVERSITY OF ILLINOIS
Urbana-Champaign Campus
The Graduate College
330 Administration Building

FORMAT APPROVAL

To the Graduate College:

The format of the thesis submitted by Benjamin Lerner
for the degree of Master of Science is acceptable to the
department of Electrical Engineering.

May 11, 1979
Date

(Signed) Paul E Mayes
Departmental Representative

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

THE GRADUATE COLLEGE

MAY, 1979

WE HEREBY RECOMMEND THAT THE THESIS BY

BENJAMIN LERNER

ENTITLED MICROCOMPUTER-CONTROLLED SERVO SYSTEM FOR AN ULTRASONIC COMPUTER-ASSISTED TOMOGRAPHIC SCANNER

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF MASTER OF SCIENCE

William D. O'Brien, Jr   Michael S. [illegible]
Director of Thesis Research

E. W. Ernst
Head of Department

Committee on Final Examination†

____
Chairman

____

____

____

____

† Required for doctor's degree but not for master's.

O-517

MICROCOMPUTER-CONTROLLED SERVO SYSTEM
FOR AN ULTRASONIC
COMPUTER-ASSISTED TOMOGRAPHIC SCANNER

BY

BENJAMIN LERNER

B.S., University of Illinois, 1975

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science
in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1979

Urbana, Illinois

ACKNOWLEDGMENT

I wish to express my thanks and appreciation to Professor William D. O'Brien, Jr. of the Bioacoustics Research Laboratory and Professor Michael S. Schlansker of the Coordinated Science Laboratory at the University of Illinois for being my advisors on this project and for giving me the support and encouragement needed to make this project a success. I further wish to thank the numerous professors and graduate students with whom I have consulted while engaged in this project. Their help and advice, in disciplines ranging from control theory to bioengineering, has been most beneficial to carrying out the work described herein.

TABLE OF CONTENTS

CHAPTER

I INTRODUCTION 1

II DESIGN CONSTRAINTS 3

III GENERAL CONTROLLER DESCRIPTION 11

IV OVERVIEW OF VELOCITY & POSITION CONTROL. 14

V MODEL OF ARMATURE-CONTROLLED DC MOTOR. 20

VI ANALYSIS AND DESIGN OF A VELOCITY CONTROL SYSTEM 25

VII STATE FEEDBACK WITH FULL-ORDER OBSERVER. 33

VIII DISCRETE IMPLEMENTATION OF VELOCITY CONTROL. . . 37

IX POSITIONAL CONTROL 47

APPENDIX

A SHAFT POSITION SENSING 54

B PULSED MOTOR CONTROL 58

C NUMERICAL CONSTANTS. 65

D SYSTEM PERFORMANCE 70

E PROGRAM LISTING. 73

REFERENCES 90

CHAPTER I

INTRODUCTION

Computer-Assisted Tomography, commonly known as "CAT Scanning", is the process of reconstructing a cross-sectional image, or tomogram, of a biological sample by using a digital computer to analyze a beam of radiation that is transmitted through the sample. The only commercial CAT scanners presently available transmit ionizing radiation (X-rays) and analyze their intensity after they pass through the sample.

The first CAT scanner to use ultrasonic radiation (sound waves at a higher frequency than the audible spectrum) was built at Mayo Clinic at Rochester, Minnesota. It is used primarily to scan women's breasts for cancerous tumors.

From the way ultrasound is believed to interact with biological tissue it is thought that ultrasonic CAT scanning (UCAT) can be used to extract information about the constituent properties of the tissue.

The Bioacoustics Research Laboratory at the University of Illinois at Urbana-Champaign has a grant from the National Institutes of Health's National Institute of General Medical Sciences (GM24994) to investigate new techniques of UCAT and to determine tissue constituent properties with ultrasound. One of the first parts of the project is to build an ultrasonic CAT scanner similar to the one at Mayo. It is from this aspect of the project that this paper developed, dealing with the development of a computer-based controller for the ultrasonic CAT scanner.

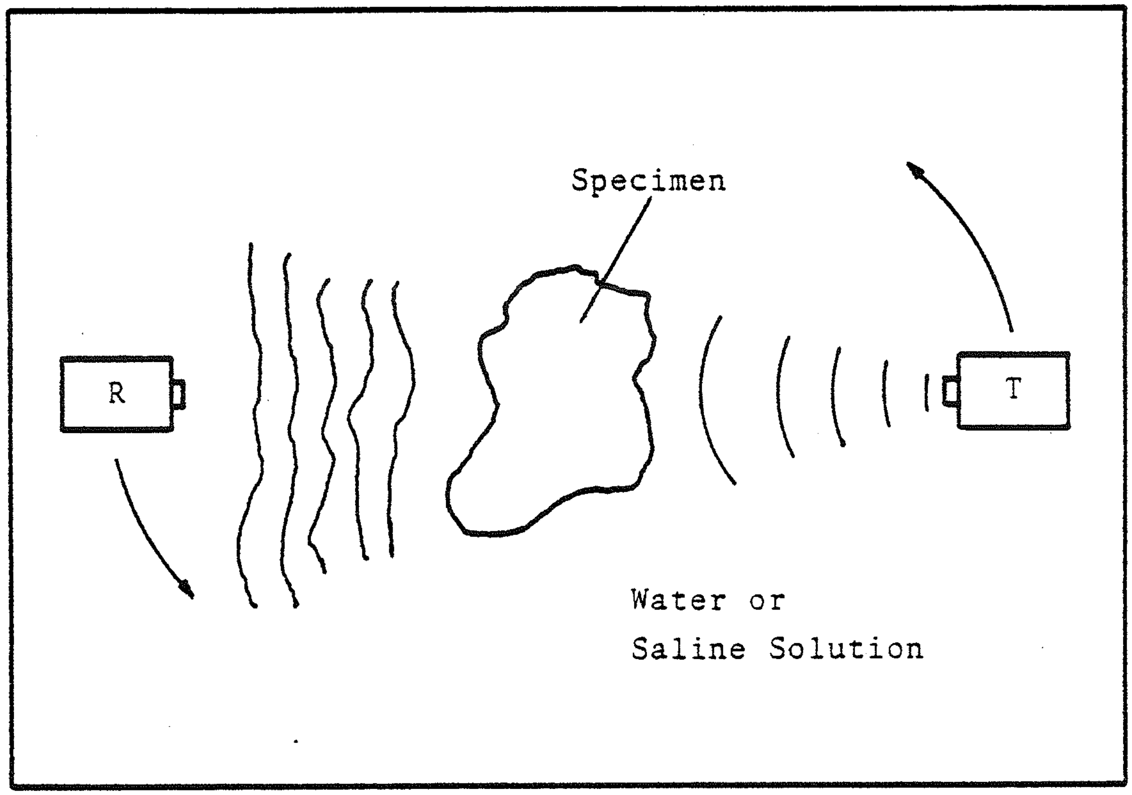
CHAPTER II

DESIGN CONSTRAINTS

Based upon extensive developmental work by Mayo Clinic engineers and scientists[1-4] and upon our evaluation of their scanner design for our research requirements, a modified scanner has been designed. The aim was to be compatable with the system at Mayo Clinic and to satisfy our requirements.

Figure 1 shows the basic scanner concept. The scanner consists of a tank filled with a liquid in which the object to be scanned is suspended. On opposite sides of the object the ultrasonic transducers are mounted, in such a way that they can be revolved around the specimen. One of the transducers will always act as the transmitter, the other as the receiver. During the scanning process the transducers will be revolved around the specimen while the transmitter produces pulses of ultrasound. The receiver will receive the pulses after they pass through the specimen and the received signal will go to a digital computer for processing.

The transducers do not revolve around the specimen with a simple, continuous motion (Figure 2). The basic transmitter-receiver frame is positioned to an angle α with respect to the specimen. With α held constant, the receiver is moved through an angle 2β at a constant velocity, receiving the ultrasonic pulses that have been scattered to various angles by the specimen. This motion of the receiver is known as the "fan beam",



T - ULTRASONIC TRANSMITTER

R - ULTRASONIC RECEIVER

Figure 1. Ultrasonic Scanner Concept.

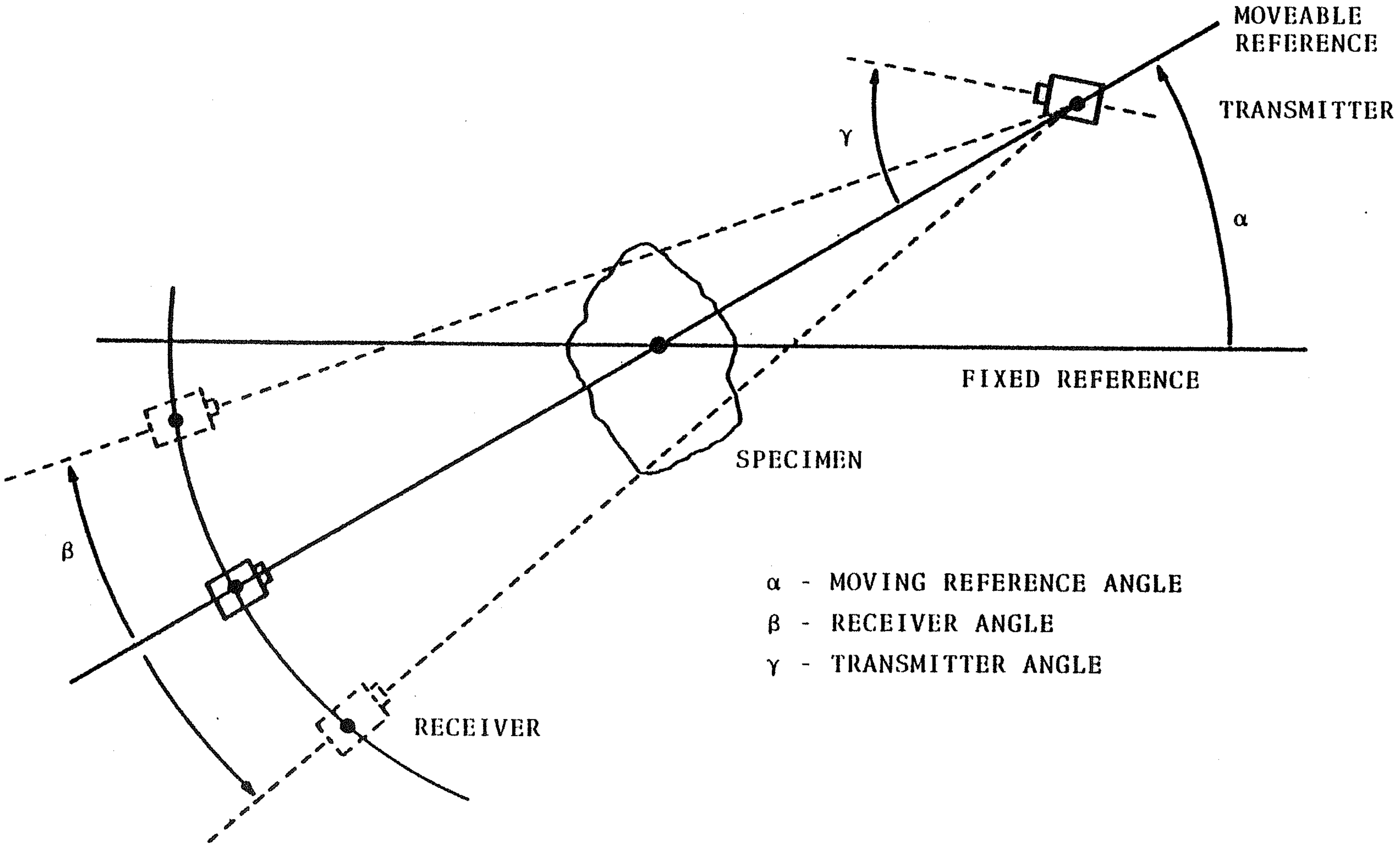


Figure 2. Transducer Motion.

and β is the fan-beam angle. When the fan-beam motion is complete, the receiver is halted and the entire transmitter-receiver frame is stepped through a small angle $\Delta\alpha$, typically 1°. The fan-beam motion is then reversed, and the receiver is moved through an angle of 2β in the opposite direction, while receiving scattered pulses from the specimen. This process is repeated until α has been moved across 180°. Because of symmetry it is not necessary to move across 360°.

The position of the receiver cannot be held constant with respect to the transmitter because the computer-algorithms that reconstruct the tomogram require pulse-transmission information through several angles of the specimen for a given reference angle α. There are two ways this information will be aquired, each requiring a different pivotal motion of the transmitter around its own axis (angle γ). During one type of data acquisition, $\gamma = \beta$ at all times so that the transmitter always points directly at the receiver. For the other type, the transmitter is pivoted to a given position and held there (γ is constant) while the receiver is swept through the fan-beam angle, 2β. It should be noted that the angles β and γ are taken relative to the moving reference-frame. The position of the moving reference frame is α, and is taken relative to the fixed reference frame, as shown in Figure 2.

It has been shown that the transducers have three degrees of freedom. Degree 1 is motion through the angle α, of the basic transmitter-receiver orientation. Degree 2 is the freedom of the receiver to move through the angle β. Degree 3 is the freedom of the transmitter to be pivoted to any angle γ, on its own axis. There is a fourth degree of freedom in the scanner, which is the ability to move the specimen up and down relative to the height of transducers, which is fixed. This allows the transducers to scan successive cross-sections of the specimen.

Each degree of freedom will need to be powered by an electric motor. A major design decision was to decide whether to use DC motors or stepping motors. A stepping motor would be most suited to the 1st degree of freedom, since it involves motion of discrete increments, $\Delta\alpha$. The 2nd degree of freedom would be most easily served by a DC motor, since it requires smooth motion over a considerable distance. Due to the nature of the gear linkages in the scanner, the 3rd degree of freedom is also best served by a DC motor. For purposes of generality, and to simplify the system's control structure, it was decided to use the same type of motor throughout the scanner. The scanner requires a controller that will accept commands from the main computer to start a scan or move to a given position. The controller will control the application of power to the motors to carry out these commands. If all the motors are of the same type the job of the controller is simplified.

In the original scanner developed at Mayo Clinic, stepping motors are used. There are several problems with using stepping motors that were determined at Mayo as a result of their experience. A stepping motor is a discrete motion device and many successive steps must be used to move the receiver through the fan-beam angle. At every step the motor starts and stops with a jerk. All of these jerks introduce vibration into the transducers. One of the measurements that is taken during a scan is the time-of-flight (TOF) of a pulse from transmitter to receiver, which is very sensitive to vibration. To decrease the vibration, Mayo developed a way of producing smooth, continuous motion with a stepping motor by varying the current through its coils in a sine-cosine pattern. With their experience it was decided against this approach because it involves taking a discrete-motion device, the stepping motor, and attaching it to a complex system to allow it to move smoothly. Since the system requires smooth motion, a DC motor was used from the start.

In choosing to use DC motors it became necessary to design a positioning system to allow the transducers to be stepped through α in increments $\Delta\alpha$. This would have been easier to do directly with a stepper motor, but this way the system gained the generality of a uniform control algorithm for all motors. The system now posseses the flexibility to step through the angle β while moving continuously through α, should a future experiment require it.

As was mentioned briefly before, the scanner must be able to receive commands from the main computer, a Perkin-Elmer 7/32. The commands will instruct the scanner where to move the transducers in the course of a scan. A control unit had to be designed for the scanner, in order to receive the commands from the computer and control the motors to carry them out. It also must be able to transmit status information back to the 7/32. It was decided that the most flexible controller would be a microprocessor. It would require the least amount of special-purpose hardware, and design improvements could be achieved simply by modifying the software. This is illustrated in Figure 3.

To this point a description of the tomography project has been presented, followed by the design constraints for the ultrasonic scanner. The project described herein is to analyze and solve the problem of controlling the DC-motors in the scanner.

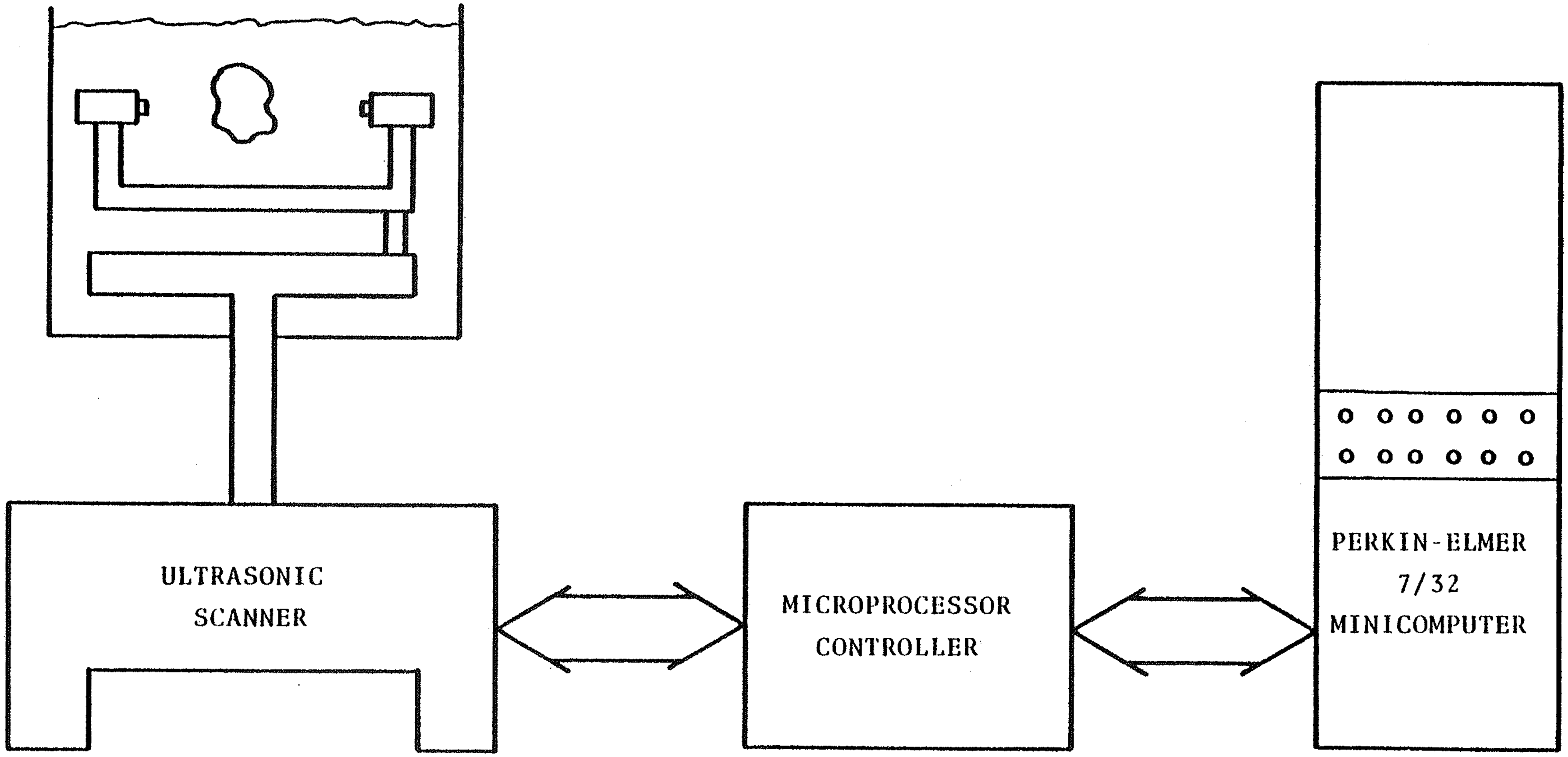


Figure 3. System Layout.

CHAPTER III

GENERAL CONTROLLER DESCRIPTION

The scanner's controller is a microprocessor that is programmed to accept commands from the 7/32 computer and control the scanner's motors in real-time to carry out those commands. It also must transmit information about the position of the transducers back to the 7/32, and is represented in a block diagram in Figure 4. The controller is implemented on a MOS Technology 6502 microprocessor. This processor was chosen because it has a very flexible instruction set [5] and an inexpensive microcomputer system utilizing this processor is available on the market[6]. This system consists of a single circuit board containing the processor, memory, input-output ports, and a sophisticated monitor program. It provides adequate software support to be usable as a self-contained development system, yet is sufficiently inexpensive to be dedicated as the controller.

The bulk of the software in the controller makes up a real-time, discrete "P-I-D" (Proportional-Integral-Differential) control algorithm[7] to control the motors. Feedback to the controller is by way of a shaft-encoder, a device that outputs a pulse for every given increment of a shaft's rotation. The encoder is mounted to provide position and velocity of the degrees of freedom. There is a separate encoder for each motor (see Appendix A).

The motors are controlled by a pulsed voltage. The pulse

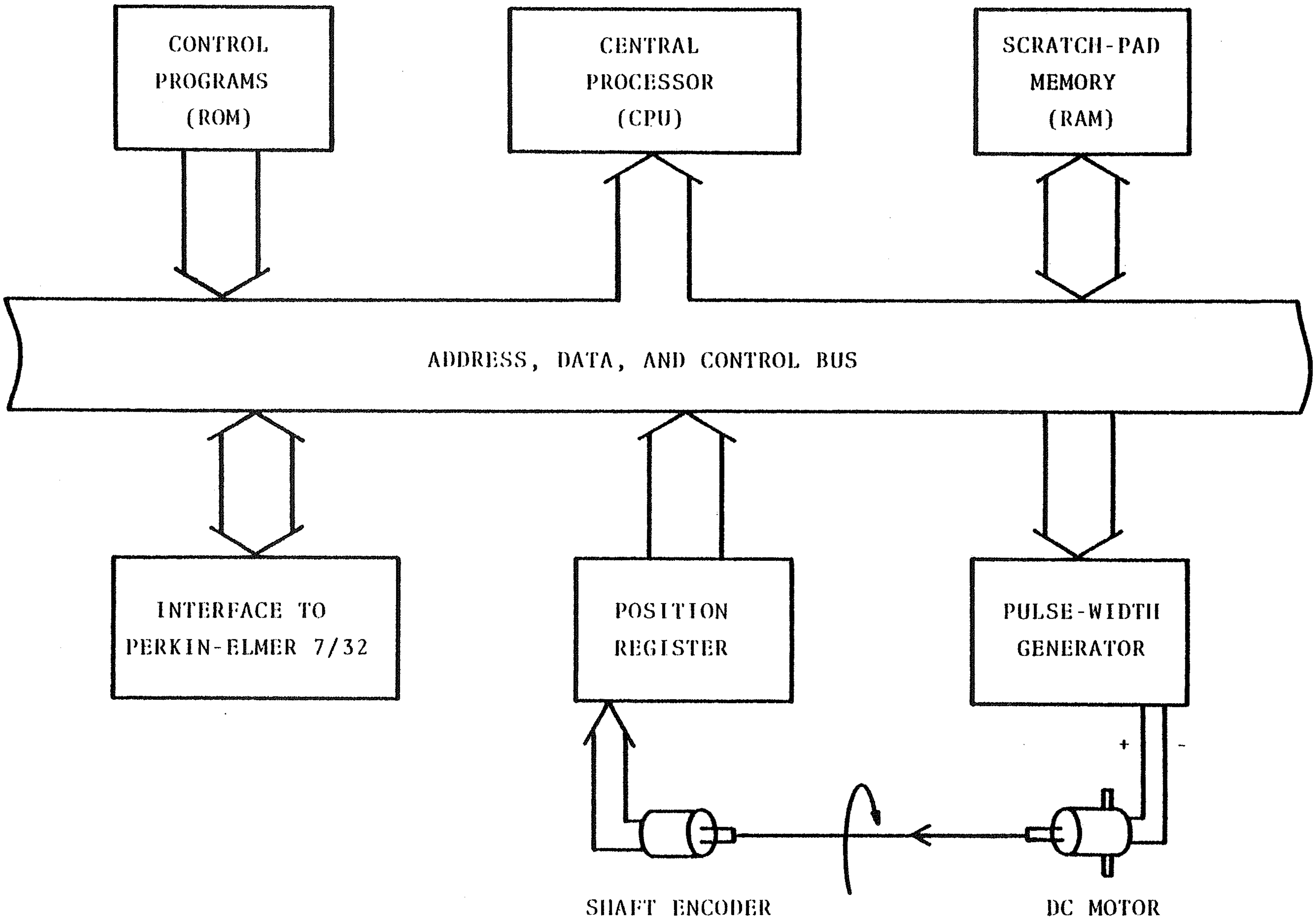


Figure 4. Microprocessor-Controller Block Diagram.

repetition frequency is fixed and the effective voltage is determined by the length of the duty cycle (see Appendix B). The controller outputs a control byte to the Pulse Width Generator (PWG), which causes it to send voltage pulses to the motor whose "ON" time is proportional to the value in the control byte. This method of control is advantageous because the final driver transistors are either off or saturated, so they do not dissipate as much power as they would if they were driven by a digital to analog converter and were operated in their linear region. The disadvantage to this approach is that by rapidly switching the power to the motor, a radio frequency interference (RFI) is caused which may interfere with the sensitive signal processing circuits that process the received ultrasonic signal. Should this become a problem, the PWG could be replaced with a digital to analog converter (DAC) and a power amplifier.

CHAPTER IV

OVERVIEW OF VELOCITY & POSITION CONTROL

The real-time controller will implement a velocity and position controller for a DC motor. An analysis of this problem follows.

Figure 5 is a block diagram of a basic velocity control system for a DC motor. The motor has an input voltage, V_m, and an output shaft velocity, ω_m. The problem at hand is to request a velocity, ω_r, and automatically control V_m so that the motor shaft turns at the rate ω_r. We wish for this to hold true even for fluctuations in the load on the motor. This is accomplished by feeding back ω_m and comparing it with ω_r. The difference between these velocities is the error velocity, denoted by ω_e. This error is the input to the controller. As long as the error is zero, the controller knows it is outputting the correct V_m. If ω_e deviates from zero due to, say, a variation in the motor's load or any other perturbation, the controller must adjust V_m to correct the speed of the motor.

A velocity control loop is the inner part of a position controller (Figure 6). The controller is given a request to move to a specific angle, θ_r. It computes the difference between the requested angle and the motor's present angle, θ_m. This is the angular error, θ_e. If θ_e is zero, the motor is positioned correctly and is not to move. If θ_e is non-zero, the position controller must request the velocity controller to

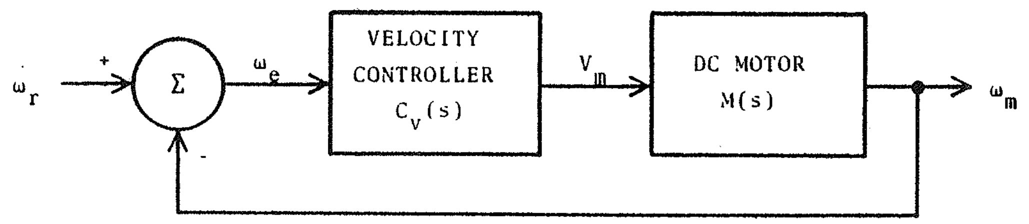


Figure 5. Velocity Control Block Diagram.

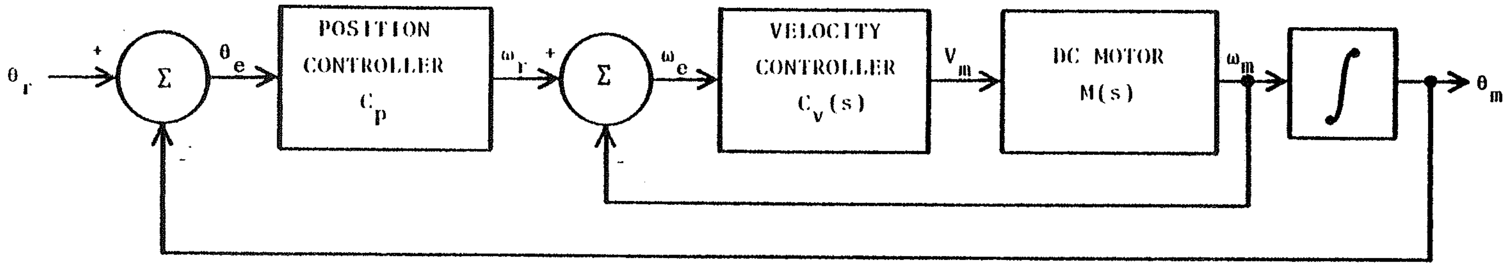


Figure 6. Position Control Block Diagram.

move the motor in the direction that will cause θ_e to decrease to zero. This is most simply done by letting $\omega_r = C_P\theta_e$ where C_P is the positional-control feedback constant. When the motor is in position, ω_r is zero. If the motor is ahead of or behind the requested position, a negative or positive velocity will be requested in proportion to θ_e. (C_P is assumed positive for this example.) As θ_e decreases, ω_r decreases with it.

Although the feedback for position control can be implemented by a constant gain (C_P), velocity control is not so simple. If a simple constant gain were used as feedback for velocity control, the motor voltage would be determined by $V_m = K_P\omega_e$. The subscript 'P' stands for 'proportional', since this is known as proportional control. If ω_e becomes zero V_m also becomes zero and the motor looses speed. An equilibrium point will be reached where the motor's velocity will be less than ω_r by enough so that V_m can keep the motor's speed constant. The error ω_e will never become zero. At equilibrium, ω_e is known as the steady-state error, e_{ss}. It is possible to decrease e_{ss} by increasing K_P, and

$$\lim_{K_P \to \infty} e_{ss} = 0$$

There are practical limitations to this, however, since the noise-sensitivity of the system, its sensitivity to random disturbances, is increased with K_P.

The solution to the problem of having steady-state error is to feed back the integral of the velocity error. Then the feedback expression becomes

$$V_m(t) = K_P\omega_e(t) + K_I \int_0^t \omega_e(\tau)d\tau. \qquad (1)$$

'K_I' is the integral-control constant. It can be seen that with this type of controller ω_e can equal zero at steady-state and V_m will still have a non-zero value, coming from the integral term. In fact, if ω_e deviates from zero by any amount, the value of the integral term will change in such a way as to correct the error. A controller of this type is known as a "PI" controller, or "proportional-integral". The time it takes to make $\omega_m = \omega_r$, or even whether the controller works at all, depends on the values chosen for K_P and K_I. The time delay for ω_m to approach a new ω_r is called the response time, and is critical in many applications including the scanner. Even greater control over response time can be achieved by feeding back the derivative of the error. Then the controller expression becomes:

$$V_m(t) = K_P\omega_e(t) + K_I \int_0^t \omega_e(\tau)d\tau + K_D \frac{d}{dt} \omega_e(t) \qquad (2)$$

'K_D' is the derivative control constant and this is known as full 'PID' control. This is a very versatile control method used in many applications requiring quick response and no steady-state error.

In order to choose the proper values for the feedback gains K_P, K_I, and K_D it is first necessary to understand the dynamics of the system to be controlled. In this case that means knowing the characteristics of the particular DC motor being used, and the characteristics of the motor's load such as moment of inertia, viscous drag, etc. The system dynamics must be described by equations that comprise a mathematical model of the system. The model should be as close as possible to the real system but some error is permissable since a feedback-controller will correct for it. There are two different approaches to computing the feedback gains, once the model of the system is derived. They both center around classical control theory. In the algebraic approach the equations describing the controller are incorporated into the system model to give a complete closed-loop system equation. When the desired closed-loop behavior of the system has been decided on, the closed-loop equation can be manipulated and solved for the values of the feedback gains that will give such a response. This approach was the first one attempted for the DC motor, but was abandoned because the equations became too unwieldly. The second approach involves using a package of computer programs for manipulating linear systems. The package, called "LINSYS", was developed at the University of Illinois [8]. It requires that the system be described in matrix form. The desired behavior of the closed-loop system is specified by requesting values for the system's closed-loop poles.

LINSYS then computes the feedback gains that give the system this behavior. This approach has been used throughout the design of the DC motor controller.

CHAPTER V

MODEL OF ARMATURE-CONTROLLED DC MOTOR

The most popular DC servomotors use armature control and have permanent-magnet field poles. Because of its relatively low cost and linear characteristics, this type of motor was selected for use in the scanner. It is controlled by varying the voltage applied to the terminals of its rotating armature[9].

An electro-mechanical model of such a motor is shown in Figure 7. Summing the voltages around the loop gives

$$V_m = L_a \frac{di}{dt} + Ri + K_v \omega_m \qquad (3)$$

The torque is described by

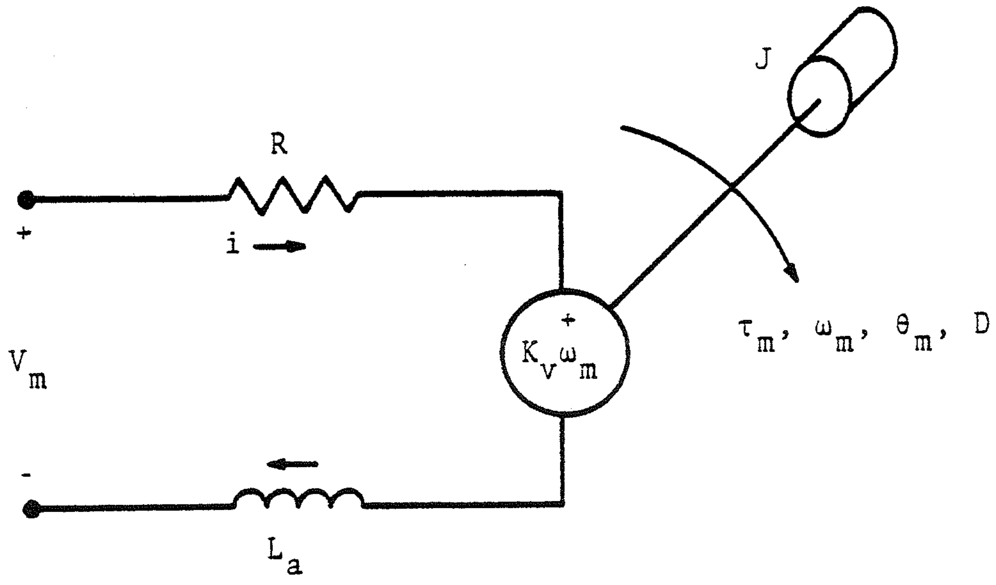
$$\tau_m = K_\tau i + D\omega_m \qquad (4)$$

and the angular acceleration is

$$\frac{d\omega_m}{dt} = \frac{1}{J} \tau_m \qquad (5)$$

These can easily be manipulated into separate electrical and mechanical equations.

$$\text{electrical:} \quad \frac{di}{dt} = -\frac{R}{L_a} i - \frac{K_v}{L_a} \omega_m + \frac{1}{L_a} V_m \qquad (6)$$



τ_m - Torque on motor shaft

ω_m - Angular velocity of shaft

θ_m - Angular position of shaft

D - Viscous damping factor

J - Moment-of-inertia of system

K_v - Armature back-emf constant

R - Armature resistance

L_a - Armature inductance

V_m - Armature voltage

i - Armature current

Figure 7. DC Motor Model.

$$\text{mechanical:} \quad \frac{d\omega_m}{dt} = \frac{K_\tau}{J}\, i + \frac{D}{J}\, \omega_m \tag{7}$$

Of primary interest in the design of a servo control system is the relationship between the applied voltage V_m and the angular velocity ω_m. The simplest way to find this is through the Laplace transform technique. The transform of equation (6) is

$$sI(s) - i\,(o) = -\frac{R}{L_a}\, I(s) - \frac{K_V}{L_a}\, \Omega_m(s) + \frac{1}{L_a}\, V_m(s) \tag{8}$$

and (7) transforms to

$$s\Omega_m(s) - \omega_m(o) = \frac{K_\tau}{J}\, I(s) + \frac{D}{J}\, \Omega_m(s) \tag{9}$$

Now it is possible to manipulate (8) and (9) algebraically to find the transfer function $\frac{\Omega_m(s)}{V_m(s)}$, after letting the initial conditions equal zero.

$$\frac{\Omega_m(s)}{V_m(s)} = \frac{K_\tau}{(L_a s+R)(Js-D)+K_\tau K_V} = \frac{K_\tau}{JL_a(s+\frac{R}{L_a})(s-\frac{D}{J})+K_\tau K_V} \tag{10}$$

If the time-domain behavior of the motor is desired for any given voltage waveform $V_m(t)$, the transformed voltage expression, $V_m(s)$, can be substituted in (10), giving an expression for $\Omega_m(s)$ in the frequency domain. This expression can be converted back to the time domain by taking its inverse Laplace transform, giving $\omega_m(t)$ for the given $V_m(t)$. This is a time-consuming process and won't be illustrated here.

It is often desired to know the steady-state velocity of a DC motor with constant $V_m(t)$. If $V_m(t)$ is constant the steady-state velocity is found through the Final-Value Theorem. It states that if $F(s) = \mathcal{L}\{f(t)\}$, then

$$\lim_{t \to \infty} f(t) = \lim_{s \to 0} sF(s) \tag{11}$$

Let $V_m(t) = v_m = \text{const.}$ Then $V_m(s) = \mathcal{L}\{V_m(t)\} = \frac{v_m}{s}$. Using (11)

$$\begin{aligned} \lim_{t \to \infty} \omega_m(t) &= \lim_{s \to 0} s\Omega_m(s) \\ &= \lim s \frac{K_\tau}{JL_a(s+\frac{R}{L_a})(s-\frac{D}{J})+K_\tau K_V} \frac{V_m}{s} \end{aligned}$$

giving

$$\omega_{m,ss} = \frac{K_\tau}{K_\tau K_V - RD} v_m \tag{12}$$

According to (12), if a DC motor obeys the linear model in Figure 7, its steady-state velocity will be directly proportional to its applied voltage.

The aforementioned transfer function is used as the basis for algebraically determining the feedback values. In order to use the LINSYS package to compute the feedback, the system must be described in matrix form,

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{13}$$

$$y(t) = Cx(t) \tag{14}$$

where x(t) is the system state vector, and u(t) is the input to the system. In this case, $x = \begin{bmatrix} i \\ \omega_m \end{bmatrix}$ and $u = V_m$. The system output is y(t), which represents the state-variables of the system that are available for measurement. The electric and mechanical equations, (6) and (7), are used to find the values for A and B, giving

$$\frac{d}{dt}\begin{bmatrix} i \\ \omega_m \end{bmatrix} = \begin{bmatrix} -R/L_a & -K_V/L_a \\ K_\tau/J & D/J \end{bmatrix}\begin{bmatrix} i \\ \omega_m \end{bmatrix} + \begin{bmatrix} 1/L_a \\ 0 \end{bmatrix} V_m \tag{15}$$

where

$$A = \begin{bmatrix} -R/L_a & -K_V/L_a \\ K_\tau/J & D/J \end{bmatrix} \text{ and } B = \begin{bmatrix} 1/L_a \\ 0 \end{bmatrix}$$

"C" is the output matrix and will be defined later. This method of defining the system is known as the matrix state equation.

The specifications and parameter values for a typical DC servomotor used in the scanner are in Appendix C.

CHAPTER VI

ANALYSIS AND DESIGN OF A VELOCITY CONTROL SYSTEM

As was stated earlier, the simplest type of velocity control uses a proportional controller, with $V = K_P(\omega_r - \omega_m)$. This controller has the drawback of permitting a steady-state error to exist in the motor's velocity. With a full PID controller the steady-state error will be zero. This can be seen by describing the closed-loop system of the motor and PID controller, shown in Figure 5. Let the PID controller be described in the frequency domain by

$$C_v(s) = \frac{V_m(s)}{\Omega_e(s)} = K_P + K_I \frac{1}{s} + K_D s \tag{16}$$

The transfer function of the motor is

$$M(s) = \frac{\Omega_m(s)}{V_m(s)} = \frac{K_\tau}{JL_a(s+\frac{R}{L_a})(s-\frac{D}{J})+K_\tau K_V} \tag{17}$$

To examine the steady-state error in ω_m it is necessary to find the transfer function of the closed-loop system,

$$\frac{\Omega_m(s)}{\Omega_r(s)} = \frac{C_v(s)M(s)}{1+C_v(s)M(s)} \tag{18}$$

which becomes

$$\frac{\Omega_m(s)}{\Omega_r(s)} = \frac{K_\tau K_D\, s^2 + K_\tau K_P s + K_\tau K_I}{JL_a s^3 + (RJ - DL_a + K_\tau K_D) s^2 + (K_\tau K_V + K_\tau K_P - DR) s + K_\tau K_I} \tag{19}$$

To determine the steady-state behavior of ω_m, the reference velocity is made constant and the Final-Value Theorem (11) is applied.

$$\Omega_r(s) = \frac{\omega_r}{s} \tag{20}$$

$$\omega_{m,ss} = \lim_{s \to 0} s \, \frac{K_\tau K_D s^2 + K_\tau K_P s + K_\tau K_I}{JL_a s^3 + (RJ - DL_a + K_\tau K_D) s^2 + (K_\tau K_V + K_\tau K_P - DR) s + K_\tau K_I} \, \frac{\omega_r}{s} \tag{21}$$

To examine the case for non-integral control, K_I is set to zero, and $\omega_{m,ss}$ becomes

$$\omega_{m,ss} = \frac{K_\tau K_P}{K_\tau K_V + K_\tau K_P - DR} \, \omega_r \tag{22}$$

From this, the steady-state error can be written as

$$e_{ss} = \omega_r - \omega_{m,ss} = \left(1 - \frac{K_\tau K_P}{K_\tau K_r + K_\tau K_P - DR}\right) \, \omega_r \tag{23}$$

It can be seen that the error actually increases with the requested velocity. The error decreases as $\frac{K_\tau K_P}{K_\tau K_r + K_\tau K_P - DR} \to 1$, which occurs when $K_P \to \infty$. A non-integral controller can be built that will have a small e_{ss} if a large feedback value is used. The trouble with using a large feedback is that it makes the system sensitive to random disturbances, such as noise.

The alternative to a proportional controller with large gain is an integral controller. From equation (21) with $K_I \neq 0$ it is seen that for integral control, $\omega_{m,ss} = \omega_r$ and $e_{ss} = 0$. Integral control was chosen for the scanner's motors for this reason. Had simple proportional control been used, the high gain necessary for small e_{ss} may have introduced enough noise to take away the inherent smoothness of a DC motor.

Placing the closed-loop system poles by hand involves equating the desired characteristic equation (C.E.) to the C.E. of the closed-loop system, then solving for the feedback gains. For reasons of flexibility and simplicity, it was decided to rework the system into matrix form and place the poles using LINSYS.

LINSYS requires the system to be in the form of equations (13)-(15). The system and desired poles are then input to LINSYS and it returns a feedback matrix $F = [f_1\ f_2\ .\ .\ .\ f_n]$ where n is the number of system state-variables. The closed-loop system will have the requested eigenvalues (poles) if

$$\begin{aligned} u &= Fx \\ &= f_1x_1 + f_2x_2 + \ .\ .\ .\ + f_nx_n \end{aligned} \tag{24}$$

This is called full-state feedback. Nowhere does LINSYS require a reference input, such as ω_r, to place the poles. The feedback is designed to bring the system to equilibrium (all derivatives zero) with the requested response pattern. Thus, during pole-placement, the reference input is considered zero. In a tracking problem such as velocity control, the actual reference is introduced in the feedback. A simple example of this involving non-integral control makes use of the system in equation (15). After specifying the system (A,B) and the required closed-loop poles to LINSYS, it will compute a matrix $F = [f_1\ f_2]$. If the feedback is provided as in equation (24), then the system will track a velocity of zero. If feedback is provided in the form

$$u = f_1 x_1 + f_2(x_2 - r) \tag{25}$$

equivalent to $$V_m = f_1 i + f_2(\omega_m - \omega_r) \tag{26}$$

then the system will adjust itself to minimize $(\omega_m - \omega_r)$ and the motor's velocity will be close to ω_r. Since this example does not provide integral control, there will be a steady-state error in tracking ω_r.

The full-state feedback that LINSYS provides may appear quite different in form than the controller expression in (2). They are, in fact, different representations of the same thing. It is easy to see how the f_2 term in (25) is equivalent to the K_P term in (2) since $\omega_e = \omega_r - \omega_m$ in (2). The f_1 term in (25) comes in place of the K_D term in (2) because $\frac{d}{dt}\,\omega_e = \frac{d}{dt}\,(\omega_r - \omega_m) = -\dot{\omega}_m$ since ω_r is constant. From (7) it is seen that $\dot{\omega}_m$

can be expressed in terms of i and ω_m, so feeding back $-\dot{\omega}_m$ with gain K_D in (2) is equivalent to feeding back i with gain f_1 in (26). The system being controlled in (25)-(26) has no state equivalent to the K_I term in (2) so full state feedback in this case will not provide integral control.

In order to use LINSYS to design an integral controller it is necessary to augment the system in (15) with a third state-variable to represent the integral of $\omega_m - \omega_r$. Then (15) becomes

$$\frac{d}{dt}\begin{bmatrix} i \\ \omega_m \\ \theta_d \end{bmatrix} = \begin{bmatrix} -R/L_a & -K_v/L_a & 0 \\ K_\tau/J & D/J & 0 \\ 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} i \\ \omega_m \\ \theta_d \end{bmatrix} + \begin{bmatrix} 1/L_a \\ 0 \\ 0 \end{bmatrix} V_m + \begin{bmatrix} 0 \\ 0 \\ -\omega_r \end{bmatrix} \quad (27)$$

The new state is

$$\frac{d}{dt}\,\theta_d = \omega_m - \omega_r \quad (28)$$

or

$$\theta_d = \int_0^t (\omega_m - \omega_r)d\tau \quad (29)$$

The final term in (27) is the disturbance input. This term is omitted for purposes of pole placement. LINSYS will compute the values for $F = [f_1\ f_2\ f_3]$ so that with

$$\begin{aligned} V_m &= F_x \\ &= f_1x_1 + f_2x_2 + f_3x_3 \end{aligned} \quad (30)$$

the system will go to equilibrium from any initial state with the requested response time. To implement this integral controller it is necessary to build an integrator for $(\omega_m - \omega_r)$. Then the third term in (30) becomes

$$f_3 x_3 = f_3 \int_0^t (\omega_m - \omega_r) d\tau \qquad (31)$$

and it is easily seen to be equivalent to the K_I term in (2). Full state feedback for the augmented system in (27) is equivalent to full PID control.

A prerequisite to being able to build the integrator for the K_I term is the ability to measure ω_m. This would normally be done by means of a tachometer-generator connected to the motor shaft. Many servomotors have built-in tachometers, including the ones used for the scanner. The tachometer works quite well at producing a voltage proportional to velocity over most of the motor's operating range, but at low velocity the accuracy diminishes greatly. Much of the operation of the scanner will require low motor velocities, especially during positioning. For this reason an alternative had to be found to the direct tachometer approach.

In order to have feedback for the positioning operation, an optical shaft encoder is connected to the final driven shaft. This device produces an output pulse every time its shaft moves through a small angle (Appendix A). This device is also used in place of a tachometer for velocity control. This is done by breaking up the integral term as shown

$$\int_0^t(\omega_m - \omega_r)d\tau = \int_0^t\omega_m d\tau - \int_0^t\omega_r d\tau \qquad (32)$$

The first term on the right in (32) is simply the motor angle, θ_m, which can be measured very accurately by keeping a count of pulses from the encoder. The second term can be thought of as representing a "lead angle", θ_f, which increases at the rate desired of the motor's angle. The integral term can then be written to look like

$$f_3x_3 = f_3(\theta_m - \theta_f) \qquad (33)$$

Whenever θ_m is unequal to θ_f the integral term contributes to the control voltage V_m in such a way as to minimize the difference.

As is seen, the integral velocity controller will try to cause θ_m to increase until it is equal to θ_f. This is equivalent to position control, where θ_f is the angular position being sought. In the case of velocity control, θ_f will be constantly increasing in the direction of ω_r. If the motor's velocity is less than ω_r, the difference angle $\theta_d = \theta_m - \theta_f$ will increase in magnitude, producing a larger contribution to V_m, which in turn will "close the gap". If ω_m gets too large θ_d will decrease in magnitude in turn slowing down the motor. How near θ_m is to θ_f is determined by the motor's voltage requirement for maintaining the requested speed. If ω_r or the load on the motor is changed, θ_d will settle to a new equilibrium value that provides the needed voltage.

If this controller were to be used to control position ω_r would be set to zero and θ_f set to θ_r, the position sought. With $\omega_r = 0$, θ_f will remain constant. As long as θ_m does not equal θ_f a control voltage will be produced to close the gap. There is a drawback to this type of position control. Due to static friction, it is impossible to achieve zero error on positioning, similar to how non-integral velocity control left a steady-state error in velocity, as seen in (22) and (23). The steady-state position error can be decreased by using a large value for f_3 but this is dangerous because 3rd-order systems tend to become unstable when this is done. Whereas this controller provides integral control of velocity, it can be used to produce only non-integral control of position. A position controller that has no error would require still another feedback proportional to $\int(\theta_m - \theta_r)d\tau$, and the system would be 4th-order.

CHAPTER VII

STATE FEEDBACK WITH FULL-ORDER OBSERVER

Implementing an integral controller with full-state feedback requires knowing the current values of all state-variables, i, ω_m, and θ_m. Unless a tachometer is used to measure ω_m, the only state-variable that is directly measureable here is θ_m. If the system (A,C) is observable it is possible to design an observer that will predict the values of the other state-variables by measuring θ_m. "A" is the system matrix from (27)

$$A = \begin{bmatrix} -R/L_a & -K_v/L_a & 0 \\ K_\tau/J & D/J & 0 \\ 0 & 1 & 0 \end{bmatrix} \qquad (34)$$

and "C" is the output matrix defining the system's output, or directly measureable state-variables. The output, y, is defined by

$$\begin{aligned} y &= Cx \\ &= [0 \quad 0 \quad 1] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad (35) \\ &= \theta_m \end{aligned}$$

since motor position is all that can be measured directly and this does not yet include θ_f (33).

An observer works by simulating the behavior of the system and correcting for any deviation between the real system and the observer's system. The observer's state is available and is used for feedback instead of the real system's state. The observer's state is denoted by $\hat{x}$, and ideally it will converge to x and remain there. The observer equation is

$$\frac{d}{dt}\hat{x} = A\hat{x} + Bu + K(y - \hat{y}) \qquad (36)$$

$$\hat{y} = C\hat{x} \qquad (37)$$

where A, B, and C are the same as for the real system. Since the observer will often start with initial conditions different from the system it's observing, the observer's state will typically be different from that of the system. This can also happen if (A,B) is a less-than-perfect model of the system. The 3rd term in (36) is to correct any error in the observed state. If the output from the observer, $C\hat{x}$, is not equal to the measured system output there is an error in the observer state. The matrix K provides a correction to the observer state based on the amount of this error, and is the "observer feedback matrix".

Determining a value for K is done in a way similar to how F was found to place the poles of (A,B). If (36) is subtracted from the system equation,

$$\frac{d}{dt} x = Ax + Bu \tag{38}$$

the result is

$$\begin{aligned}\frac{d}{dt} (x - \hat{x}) &= A(x - \hat{x}) - K(Cx - C\hat{x}) \\ &= (A - KC)(x - \hat{x})\end{aligned} \tag{39}$$

It is desired that $\hat{x} \rightarrow x$, or $(x - \hat{x}) \rightarrow 0$. If (A,C) is observable, K can be found so that $(x - \hat{x}) \rightarrow 0$ as rapidly as desired. This is equivalent to using K to place the poles of (A,C) the way F was used to place the poles of (A,B) in (30). LINSYS can be used to compute the value of K, given the desired observer poles.

This is called a full-order observer because it reconstructs the complete state of the system. It has the drawback of also reconstructing the measured state-variable, which is redundant. Nevertheless, a full-order observer is the simplest type to design. It is also possible to design a reduced order observer that observes only those state-variables that are not available through y. When state feedback is provided from the observer's state the system will have the same closed loop eigenvalues as if actual state feedback were provided, if the observer poles are faster (farther to the left) than the system's. Otherwise $\hat{x}$ will not be able to track x and the system may be unstable. The poles of (A - KC) should be chosen to be to the left of the poles of (A + BF) in the s-plane.

Ideally it may seem desirable to make the system and observer infinitely fast so $\hat{x} \rightarrow x$ and $\omega_m \rightarrow \omega_r$ immediately. A faster system requires larger feedback values, and there is a practical limit to how great a feedback can be used. Also, a slower system is more insensitive to noise. Noise in this case is any high frequency disturbance, or other inaccuracy in the system. If the system has a faster response it has a higher bandwidth and will attempt to track the noise, producing vibration and unsteadiness at the output. This would defeat the purpose of using a DC motor instead of a stepping motor as described earlier. If the observer has too fast a response it may tend to overcompensate for any inaccuracy in $\hat{x}$ and the trajectory of $\hat{x}$ will swing back and forth as it follows the trajectory of x, unless the implementation of the observer has high accuracy and low noise.

CHAPTER VIII

DISCRETE IMPLEMENTATION OF VELOCITY CONTROL

The preceeding discussion centered on the design of a PID velocity controller, taking note of the requirements of the ultrasonic scanner. The controller is to be implemented on a microprocessor, which makes it a discrete controller. That is, it receives information about the system only at discrete time intervals kT, k = 0, 1, 2, ... It can only update its control output when it receives new input about the system, and the control stays constant during each interval (Figure 8). If the sampling and updating frequency is high the system with a discrete controller will resemble one with a continuous, analog controller. If the feedback-gains for a continuous system are used in a discrete controller, the sampling rate would have to be high or the system may become unstable.

It is possible to design a discrete controller with feedback gains "custom tailored" for a given sampling frequency. To do this it is necessary to know how much the system will change if its input is held constant for time T. Since a DC motor is a linear system it is possible to find the motor's state at time t = (k + 1)T by summing its zero-input response and zero-state response over period T.

$$x[(k+1)T] = \phi(T)x[kT] + \int_{t=0}^{t=T} \phi(t)B\,dt\; u[kT] \qquad (40)$$

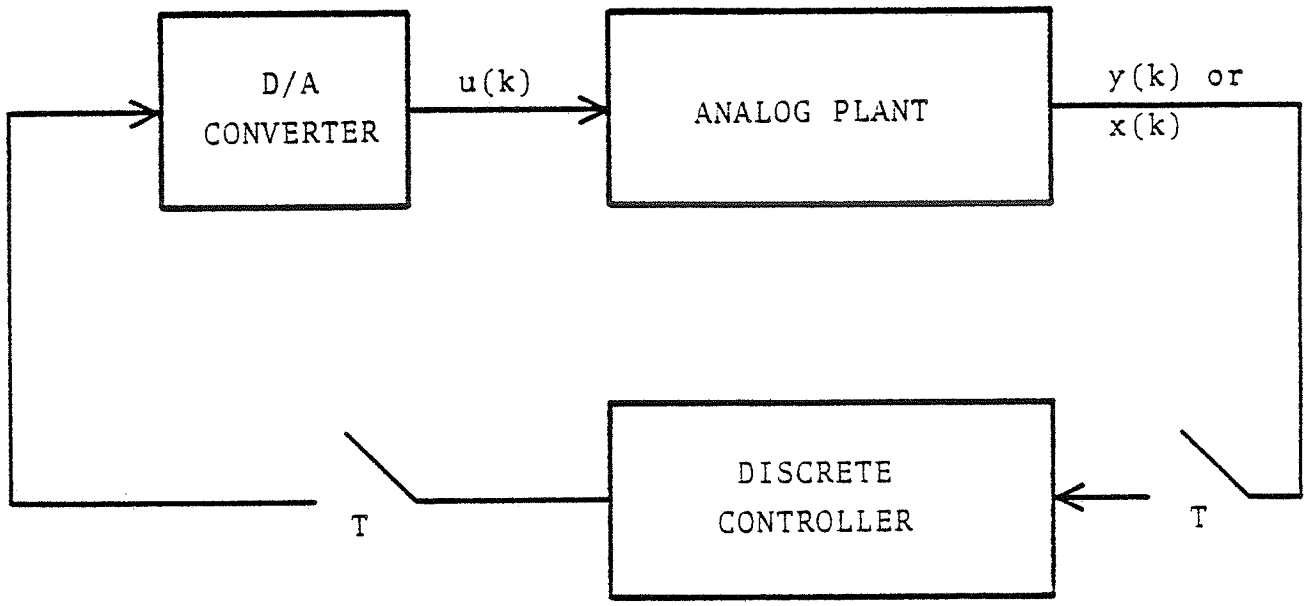


Figure 8. General Sampled-Data System.

The first term on the right in (40) is the zero-input response of the system, starting with its state at kT. The second term is the zero-state response for the control u that was last updated at t = kT. Since u(kT) is a constant during the interval it was removed from within the integral. The matrix $\phi(t)$ is the state transition matrix, where

$$\phi(t) = \mathcal{L}^{-1}\{(sI - A)^{-1}\} \tag{41}$$

A discrete state equation can be written from (40), analogous to the continuous state equation of (13) and (14). In this equation, x(k) is written instead of x(kT).

$$x(k + 1) = A_d\, x(k) + B_d\, u(k) \tag{42}$$

$$y(k) = C_d\, x(k) \tag{43}$$

The discrete system matrix is A_d,

$$A_d = \phi(T) \tag{44}$$

and the discrete input matrix is B_d,

$$B_d = \int_{t=0}^{t=T} \phi(t)\ Bdt \tag{45}$$

In the output equation (43) the C_d matrix equals its continuous counterpart. Equation (42) is a difference equation, giving the state of the system one time period after a specific input was applied to it.

Using (40) it is possible to discretize the DC motor equations from (27). The final result is shown in (46)-(48). It should be noted that a system must be discretized for a particular T. If the period is changed, a new discretized system must be computed.

$$\begin{bmatrix} i(k+1) \\ \omega_m(k+1) \\ \theta_m(k+1) \end{bmatrix} = A_d \begin{bmatrix} i(k) \\ \omega_m(k) \\ \theta_m(k) \end{bmatrix} + B_d V_m(k) \qquad (46)$$

See Appendix C for some values for matrixes A_d and B_d at various sampling periods T.

Determining the feedback to control a discrete system is basically the same as for a continuous system. It is desired to find a matrix or vector $F_d = [f_1 \; f_2 \; f_3]$ such that when

$$u(k) = F_d \, x(k) \qquad (49)$$

the system states will go to zero after several iterations. The number of iterations it takes is determined by the discrete eigenvalues. If state variable x_j is uniquely associated with discrete eigenvalue z_j, then $x_j(k+1) = z_j \, x_j(k)$. If $|z_j| < 1$

$$A_d = \begin{bmatrix} e^{\sigma T}[\cos\omega T + \frac{\sigma - D/J}{\omega}\sin\omega T] & -e^{\sigma T}\frac{K_v/L_a}{\omega}\sin\omega T & 0 \\ e^{\sigma T}[\frac{K_T/J}{\omega}\sin\omega T] & e^{\sigma T}[\cos\omega T + \frac{\sigma + R/L_a}{\omega}\sin\omega T] & 0 \\ \frac{K_T/J}{\sigma^2 + \omega^2}[e^{\sigma T}(\frac{\sigma}{\omega}\sin\omega T - \cos\omega T) + 1] & \frac{e^{\sigma T}}{\sigma^2 + \omega^2}[(\omega + \frac{\sigma}{\omega}(\sigma + \frac{R}{L_a}))\sin\omega T - \frac{R}{L_a}\cos\omega T] + \frac{R/L_a}{\sigma^2 + \omega^2} & 1 \end{bmatrix} \qquad (47)$$

$$B_d = \begin{bmatrix} \frac{1/L_a}{\sigma^2 + \omega^2}\{e^{\sigma T}[(\omega + \frac{\sigma^2 - \sigma D/J}{\omega})\sin\omega T + \frac{D}{J}\cos\omega T] - \frac{D}{J}\} \\ \frac{K_T/JL_a}{\omega(\sigma^2 + \omega^2)}\{e^{\sigma T}(\sigma\sin\omega T - \omega\cos\omega T) + \omega\} \\ \frac{K_T/JL_a}{\sigma^2 + \omega^2}\{\frac{e^{\sigma T}}{\sigma^2 + \omega^2}[\frac{\sigma^2 - \omega^2}{\omega}\sin\omega T - 2\sigma\cos\omega T] + T + \frac{2\sigma}{\sigma^2 + \omega^2}\} \end{bmatrix} \qquad (48)$$

then x_j will go to zero and the closer z_j is to zero, the faster it will converge. If for all discrete eigenvalues z_i,

$$|z_i| < 1 \tag{51}$$

the system is stable. The discrete eigenvalues are related to the continuous poles s_i by

$$z_i = e^{s_i T} \tag{52}$$

Computing the discrete feedback F_d is done in the same way as computing F in the continuous case, and can be done using LINSYS. The desired values for z_i are given to LINSYS along with A_d and B_d. LINSYS will compute F_d such that the poles of the closed-loop system,

$$\begin{aligned} x(k + 1) &= A_d\ x(k) + B_d\ F_d\ x(k) \\ &= (A_d + B_d F_d)\, x(k) \end{aligned} \tag{53}$$

are at z_i. This value of F_d will only provide such a response if it is used at the sampling frequency for which A_d, B_d, and z_i were computed.

The process of determining the feedback for the scanner's controller starts with choosing s-plane poles that give the desired response for the continuous system. These poles are transformed to the z-plane using (52), after a suitable value for T is chosen. The matrices A_d and B_d must be computed for the chosen T is and then they and the poles given to LINSYS. At this point LINSYS can compute F_d.

Since only one state-variable can be directly measured the others must be observed by a discrete observer. A discrete version of the observer of (36) must be designed and implemented on the microprocessor. For a stable system the poles of the observer must be chosen to lie to left of the system poles in the s-plane or closer to the origin in the z-plane. After choosing poles for the observer, LINSYS can compute the discrete observer feedback matrix K_d for the system matrix A_d and the output matrix $C_d = C$. The microprocessor must be programmed to implement the discrete version of (36), which is

$$\hat{x}(k + 1) + A_d \hat{x}(k) + B_d u(k) + K_d(y(k) - C_d\hat{x}(k)) \tag{54}$$

The system being observed is closed-loop and its input is computed from (49). This can be used to simplify (54) by substituting for u(k).

$$\begin{aligned} \hat{x}(k + 1) &= A_d \hat{x}(k) + B_dF_d \hat{x}(k) - K_dC_d \hat{x}(k) + K_d y(k) \\ &= (A_d + B_dF_d - K_dC_d) \hat{x}(k) + K_d y(k) \end{aligned} \tag{55}$$

Equation (55) defines a discrete observer for a closed-loop system. At each iteration it computes a new value of $\hat{x}$ based on the old value and the measured system output, y. The overall discrete observer matrix is

$$A_{od} = A_d + B_d F_d - K_d C_d \tag{56}$$

Equation (55) must be modified slightly to incorporate the velocity setpoint. In the original augmented system of (27) $\hat{x}_3$ was not the true motor angle, but was the integral of $\omega_m - \omega_r$ (29). This must be taken into account in (55) where $\hat{x}_3$ represents the motor angle alone. The control expression is

$$\begin{aligned} u(k) = V_m &= F_d \hat{x}(k) \\ &= f_1 \hat{x}(k) + f_2 \hat{x}_2(k) + f_3 \hat{x}_3(k) \end{aligned} \tag{57}$$

As shown in (32) and (33), the third state-variable is the difference between θ_m and θ_f. For velocity control θ_f increases at the rate $T\omega_r$ each time period, since $\theta_f = \int_0^t \omega_r dt$. One way to modify (55) so that $\hat{x}_3$ becomes $\theta_m - \theta_f$ is to subtract $T\omega_r$ from $\hat{x}_3$ at each iteration. It is also necessary to modify the system output y. In (55) $y = \theta_m$, the measured motor angle. Since $\hat{x}_3$ was modified to become $\theta_m - \theta_f$ it is necessary to let

$$y(k) = \theta_m(k) - \theta_f(k) \tag{58}$$

This adjusts the observer's input, the system's "output", to be consistent with $\hat{x}_3$. A new value of θ_f must be computed during each control iteration.

A summary of the equations to be carried out by the microprocessor during each control cycle follows.

$$V_m = u(k) = F_d \hat{x}(k) \tag{59}$$

$$\hat{x}(k+1) = A_{od} \hat{x}(k) - \begin{bmatrix} 0 \\ 0 \\ T\omega_r \end{bmatrix} + K_d(\theta_m(k) - \theta_f(k)) \tag{60}$$

$$\theta_f(k+1) = \theta_f(k) + T\omega_r \tag{61}$$

The only quantity that is sensed from the outside world is $\theta_m(k)$, which is a count of pulses from the position encoder.

So as not to count unreasonably high, the counter for the encoder resets to zero after counting through 360°. If the encoder turns past zero in the negative direction, the count is set to the maximum. In this way each position has a unique angle associated with it. Whenever the encoder crosses zero the value of θ_f must be corrected so that the difference $\theta_m - \theta_f$ remains constant. The encoder has a zero-reference output line. This line is used to interrupt the processor each time the encoder crosses zero so that θ_f may be corrected.

Figure 9 is a simplified flowchart of the discrete controller-observer. Graphs of this controller's performance are found in Appendix D.

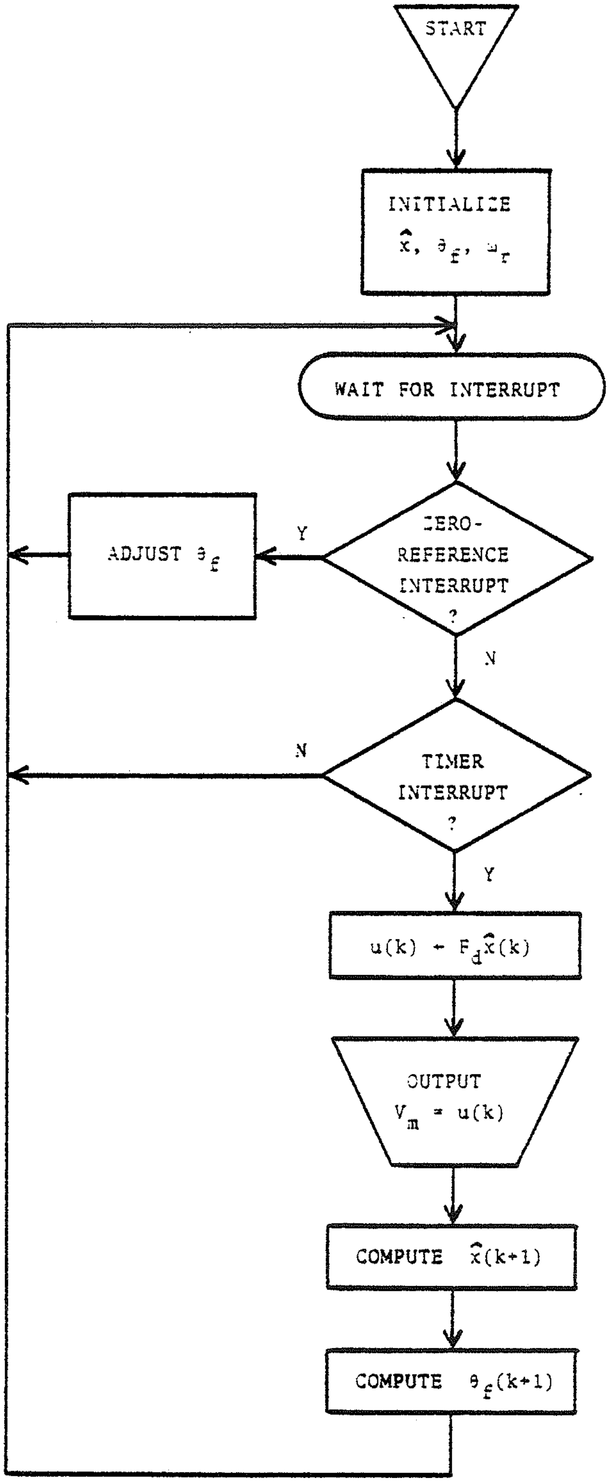


Figure 9. Velocity-Control Algorithm with Observer.

CHAPTER IX

POSITIONAL CONTROL

It has been explained that the velocity control system just described can be used to control motor position, but there will usually be a positioning error due to static friction. In a proportional position controller such as this one, the motor voltage is proportional to the positional error

$$V_m = C_p(\theta_r - \theta_m) \tag{62}$$

When the error becomes small enough that the voltage no longer overcomes static friction the motor stops, leaving a positioning error.

This error can be overcome by using the previously-designed integral velocity controller as the inner loop of the position controller shown in Figure 6. As long as there is a positional error, a nonzero velocity will be requested from the velocity controller to correct the error. However small the requested velocity is, the velocity controller will be able to overcome friction and move toward the requested position, since it uses integral control. Once the velocity controller has been designed the only parameter of position control that remains to be adjusted is C_p, the positional feedback constant, where

$$\omega_r = C_p(\theta_r - \theta_m) \tag{63}$$

The system is not in a form in which LINSYS can be used to find C_p. Rather than manipulate the new system into matrix form it was decided to modify the algorithm of (59)-(61) (Figure 9) to include (63). The modification is to compute a new ω_r at each iteration, rather than keep it fixed as in velocity control. When a zero-reference interrupt occurs, θ_r must be adjusted for the same reason θ_f had to be adjusted during velocity control. Figure 10 is a flowchart of the modified controller. With a controller that implements this algorithm it is possible to note the resultant system response for varying values of C_p, and determine the best value to use experimentally.

The experimental results were not too encouraging. For all but small values of C_p the system had a great deal of "ringing" and overshoot. When C_p was made small enough that this wouldn't happen, the system response was quite slow. Appendix D shows graphs of the new controller's performance for various values of C_p, using the same velocity controller as before. It can be seen that the system is severely underdamped for all but small C_p.

It may be possible to speed up the response and improve the damping by changing the poles of the velocity controller. A more direct approach is to redesign the position controller as a 4th-order system and use LINSYS to determine the feedback, the same way the velocity controller was designed. The result of this will be an integral position controller. The 3rd-order system (27) is augmented with a fourth state-variable, q, to

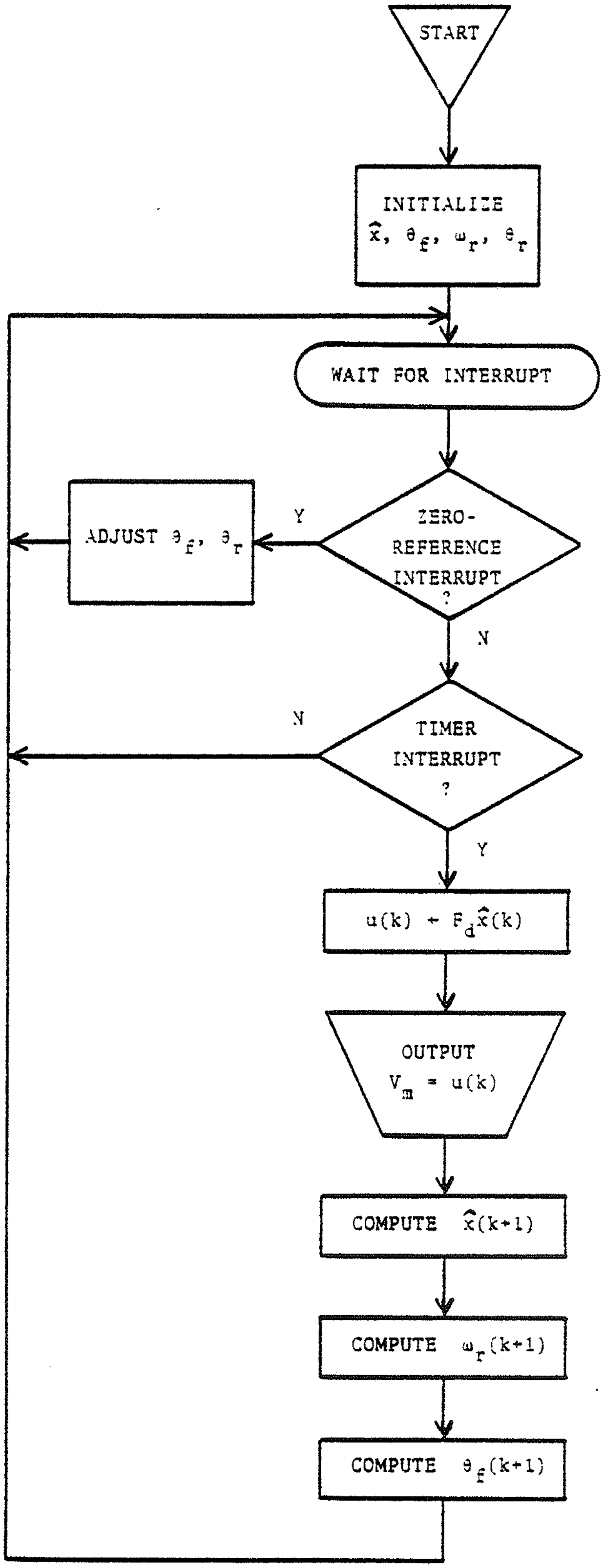


Figure 10. Simplified Position-Control Algorithm.

represent the integral of the position error,

$$q(t) = \int_0^t (\theta_m(\tau) - \theta_r(\tau))d\tau \qquad (64)$$

The augmented system becomes

$$\frac{d}{dt}\begin{bmatrix} i \\ \omega_m \\ \theta_m \\ q \end{bmatrix} = \begin{bmatrix} -R/L_a & -K_V/L_a & 0 & 0 \\ K_\tau/J & D/J & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} i \\ \omega_m \\ \theta_m \\ q \end{bmatrix} + \begin{bmatrix} \frac{1}{L_a} \\ 0 \\ 0 \\ 0 \end{bmatrix} V_m + \begin{bmatrix} 0 \\ 0 \\ 0 \\ -\theta_r \end{bmatrix} \qquad (65)$$

It should be noted that the third state-variable is θ_m and not θ_d as in (27). In (27) the third state-variable was the integral of the velocity error but in (65) the reference velocity is set to zero so the third state-variable is the motor position, θ_m. With full-state feedback, the control voltage becomes

$$V_m = f_1 i + f_2\omega_m + f_3\theta_m + f_4 \int_0^t (\theta_m - \theta_r)d\tau \qquad (66)$$

If the motor is told to seek a position but stops in the wrong place because of friction, the fourth term in (66) will increase until V_m is great enough to overcome the friction.

When it is desired for the system in (65) to have a constant motor velocity, ω_r can be entered as a disturbance as shown in (67).

$$\frac{d}{dt}\begin{bmatrix} i \\ \omega_m \\ \theta_d \\ q \end{bmatrix} = \begin{bmatrix} -R/L_a & -K_V/L_a & 0 & 0 \\ K_\tau/J & D/J & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} i \\ \omega_m \\ \theta_d \\ q \end{bmatrix} + \begin{bmatrix} \frac{1}{L_a} \\ 0 \\ 0 \\ 0 \end{bmatrix} V_m + \begin{bmatrix} 0 \\ 0 \\ -\omega_r \\ -\theta_r \end{bmatrix} \qquad (67)$$

In this system the third state-variable is written as θ_d and has the same meaning as in (27), it being the difference angle between the motor position and the integral of ω_r. The control voltage from (67) is

$$V_m = f_1 i + f_2\omega_m + f_3\theta_d + f_4 \int_0^t(\theta_d - \theta_r)d\tau \qquad (68)$$

If θ_r is any arbitrary constant, the closed-loop system of (67) and (68) will settle to equilibrium with $\theta_d = \theta_r$ and $\omega_m = \omega_r$.

Since this 4th-order system is completely controllable the feedback gains f_i can be chosen for any choice of closed-loop eigenvalues. This allows the positioning system to be designed for a fast response without excessive overshoot. The disadvantage to this approach is that 4th-order systems are less stable than 3rd-order, so fluctuations in system parameters are more likely to cast the system into instability. Furthermore, if the only state-variable available for measurement is θ_m it is necessary to build an observer to facilitate full-state feedback. If the observed state is "noisy", due to inaccuracies in the model or the observer being too sensitive, the system may become unstable.

It is undesirable to build a 4th-order observer for this system, since the 4th state-variable is an artificial one and observing it would be redundant. No other state-variable in the system depends on q, so the 3rd-order observer for the system in (34) and (35) can be used. This significantly reduces the amount of calculating that must be done during each iteration of the control algorithm.

Determining the discrete feedback gains and designing the discrete observer is the same as for velocity control, except that to compute the feedback the discrete form of the system in (65) is used. Since the observer is of a lower order than the augmented system, the simplification of the observer expression shown in (55) must be modified. One way this can be done is by letting

$$\begin{aligned}\hat{x}(k+1) &= A_d\,\hat{x}(k) + B_d u(k) - K_d C_d\,\hat{x}(k) + K_d\,y(k)\\ &= (A_d - K_d C_d)\,\hat{x}(k) + B_d u(k) + K_d\,y(k)\end{aligned} \tag{69}$$

In these expressions, A_d, B_d, K_d, and C_d are the previous matrices from the 3rd-order velocity-control system. The control u(k) is computed with the 4th-order feedback gains,

$$u(k) = f_1\,\hat{x}_1(k) + f_2\,\hat{x}_2(k) + f_3\,\hat{x}_3(k) + f_4 q(k) \tag{70}$$

The control algorithm is complete when the reference inputs for velocity and position are included. It resembles the velocity controller in (59)-(61).

$$V_m = u(k) = [f_1\ f_2\ f_3\ f_4]\begin{bmatrix}\hat{x}(k) \\ ---- \\ q(k)\end{bmatrix} \tag{71}$$

$$\hat{x}(k+1) = (A_d - K_dC_d)\hat{x}(k) - \begin{bmatrix} 0 \\ 0 \\ T\omega_r \end{bmatrix} + B_d u(k) + K_d(\theta_m(k) - \theta_f(k)) \tag{72}$$

$$q(k+1) = q(k) + T[\theta_m(k) - \theta_f(k) - \theta_r] \tag{73}$$

$$\theta_f(k+1) = \theta_f(k) + T\omega_r \tag{74}$$

Velocity control on the above system is achieved by setting ω_r to the desired velocity and θ_r to any constant, typically zero. Positioning is accomplished by setting ω_r and θ_f to zero and initializing $\hat{x}_3$ to θ_m. Then, when θ_r is set to the desired position, the motor will move there. Had θ_f not been set to zero, the motor would have turned to a position such that $\theta_m - \theta_f = \theta_r$.

APPENDIX A

SHAFT POSITION SENSING

The microprocessor controller is responsible for keeping track of the transducers' position angles, α, β, and γ, so that position information can be transmitted to the 7/32 computer while ultrasonic data is being taken. This information can also be used to determine the rate at which a given angle is changing. The task of position sensing is accomplished by connecting an optical shaft encoder to the drive shaft of each degree of freedom.

An encoder consists of a "slotted" disk mounted on a shaft. A beam of light is passed through the disk and is sensed by a photocell. As the shaft turns the light beam gets interrupted, causing the photocell to emit a voltage that varies with the motion of the disk. The photocell output gets passed to a conditioning circuit which processes it to standard logic voltages, and is then transferred out of the encoder. The encoders used in the scanner have three output lines, CCW, CW, and ZREF. If the encoder's shaft is being turned in a counterclockwise direction the CCW line is pulsed at a rate of 10 pulses per degree of shaft rotation. The line is at ground potential otherwise. The same is true of the CW line for clockwise shaft rotation. There is a separate channel of information on the encoder's disk for zero-reference. The ZREF line is pulsed once for each rotation of the shaft as it moves past a fixed point.

The motion of each transducer is induced by its respective DC motor through several reduction gears. This increases the effective torque and improves the positioning accuracy, but also introduces backlash into the system. So as to accurately know each transducer's position despite the backlash, each encoder is directly connected to the shaft on which its degree-of-freedom pivots. It is necessary to know the position of each transducer to within 0.1°, so encoders with 0.1° of accuracy were chosen.

A shaft's absolute position can be determined by counting pulses from the encoder (Figure A-1). A bidirectional binary counter is driven by the encoder outputs, the CCW line causing it to count up and the CW line causing it to count down. A counterclockwise shaft motion was chosen to be positive. The counter is never permitted to reach a value higher than 3599, since at 10 pulses per degree the encoder produces 3600 pulses per turn. The encoder's ZREF line will reset the counter to zero if it attempts to count higher than 3599 while turning counterclockwise. If the encoder is turned clockwise past zero the counter will automatically be loaded with 3599, and is never allowed to go negative. In this way each position of the encoder shaft has a unique angle associated with it. The counter associated with an encoder is the position register for that degree of freedom. It is connected to the microprocessor's bus and can be read by the control program. When the

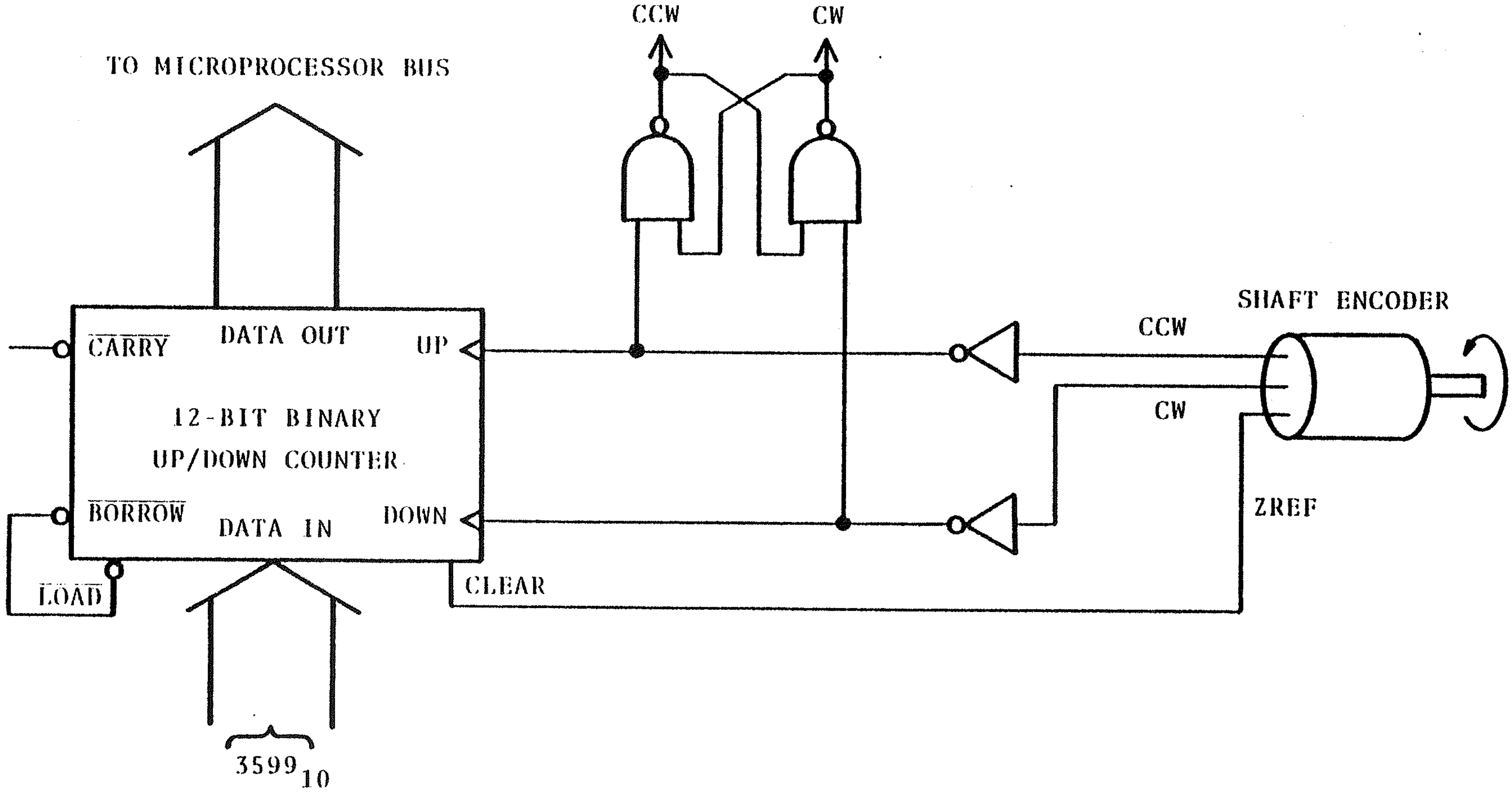


Figure A-1. Position Sensing.

system is powered-on initially the position register will not reflect the position of its associated transducer. This is dealt with by instructing each motor to move in a predetermined direction until the occurance of a pulse from its ZREF line. This will automatically initialize its position register and inform the microprocessor controller that this transducer is at its "home" position. An R-S flipflop connected across the inputs to the counter keeps track of the most recent direction in which the shaft moved. This signal is also read by the microprocessor, and is used by parts of the control program.

APPENDIX B

PULSED MOTOR CONTROL

The DC motors in the scanner are operated over a wide range of speeds, and under viscous loading conditions since the transducers are immersed in water. A significant amount of torque is required to rapidly accelerate the transducers to their required scanning speed and keep the speed constant. As seen in equations (4) and (7) the motor's torque is proportional to the current through its armature, and as the velocity through a viscous medium increases the current-requirements of the motor to counteract the drag increase. From equation (12) it is seen that the motor's steady-state speed is proportional to its supply voltage.

The microprocessor controller will output a control byte representing a motor voltage during each control cycle. Two approaches to powering the motor were considered: sending the byte to a digital to analog converter (DAC) and then to a power amplifier, or using the byte to control a pulsed voltage to the motor. Due to the wide range of voltages and currents that a single motor requires, pulsed voltage control was chosen. This is because a linear power amplifier will have to drop a significant voltage when operated below full power, which requires extensive heat sinking and raised questions of reliability.

The pulsing method that was chosen involves varying the duty cycle of a pulse stream that is of fixed frequency, illustrated in Figure B-1. All pulses are of fixed height, and a higher effective voltage is equivalent to wider pulses. This pulse-width modulation method was chosen because it is straightforward to implement with digital circuitry. A question arose as to whether a DC motor would behave with a pulsed voltage similar to when it is given the equivalent continuous voltage. The DC-motor model of equations (6) and (7) was subjected to a computer simulation using IBM's Continuous Systems Modeling Program (CSMP III)[10]. In the simulation, V_m was allowed to be both continuous and pulsed. As long as the pulse period was less than the motor's response time there was very little difference between the two cases. The steady-state velocity was the same in each case.

The circuit that receives the control byte is shown in simplified form in Figure B-2. A 1 MHz clock is continuously supplied to an 8-bit binary counter. This clock frequency is not critical, but was chosen because it could conveniently be tapped from the microprocessor. The counter's output is sent to an 8-bit magnitude-comparator circuit, to be compared to the control byte. As long as the control byte is greater than the value in the counter the comparator outputs a signal to turn on the motor. When the counter exceeds the value of the control byte the output pulse is turned off, and stays off until the counter resets itself to zero. A higher value of control byte will create longer pulses. The microcomputer outputs a ninth bit, separate from the voltage-byte, to determine motor

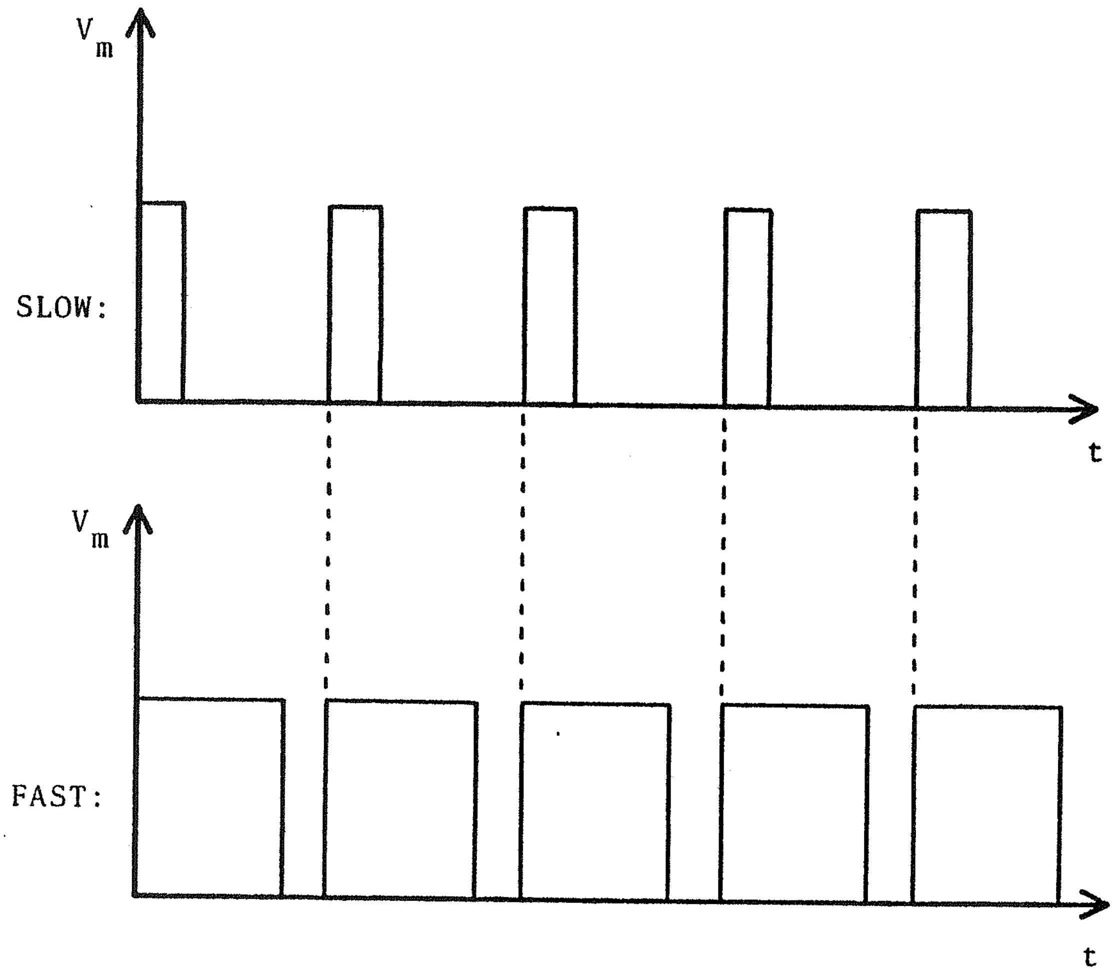


Figure B-1. Pulse-Width Modulation.

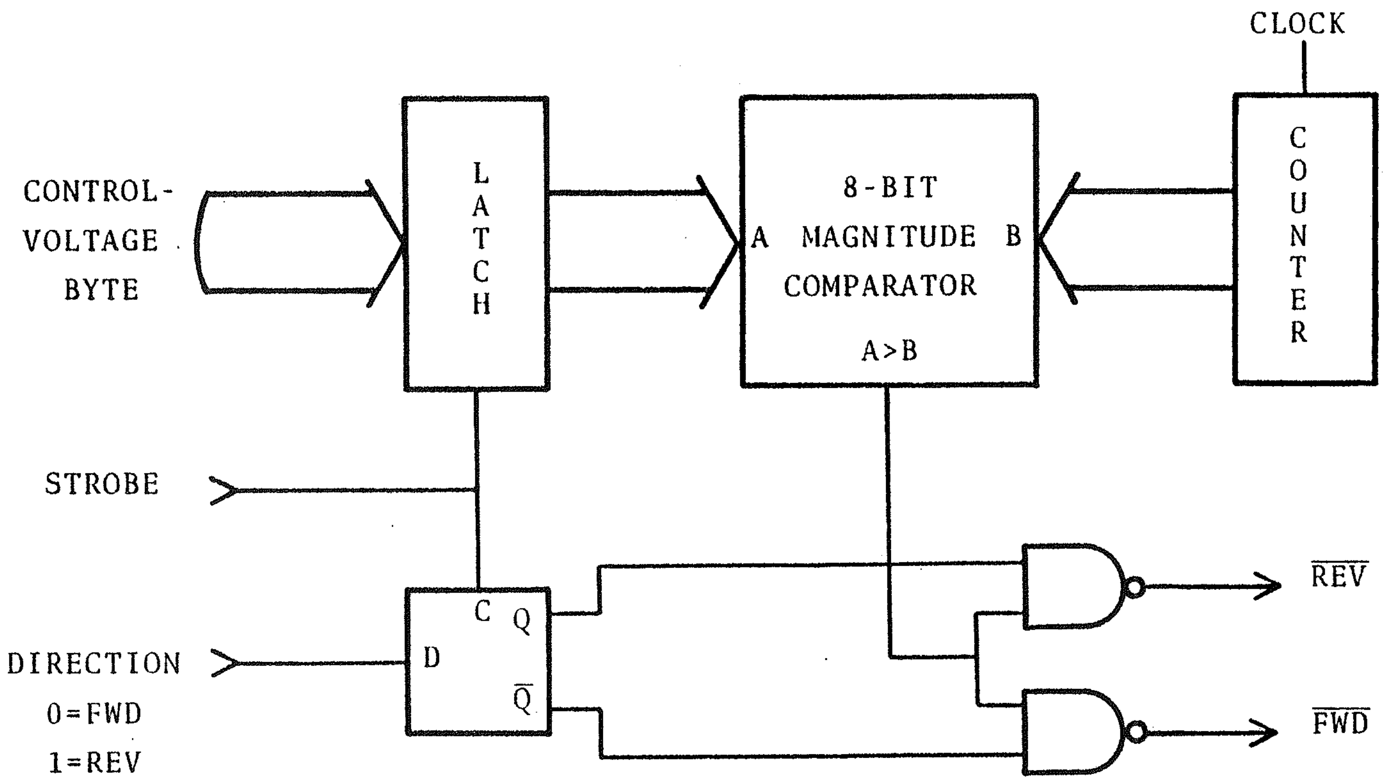


Figure B-2. Pulse-Width Generator.

direction. This bit routes the voltage pulses to the correct side of the bidirectional driver circuit.

The motor driver circuit is shown in Figure B-3. Power is applied to the motor by bringing either the $\overline{\text{FWD}}$ or $\overline{\text{REV}}$ input low. These lines are normally high. With the inputs high, Q_3, Q_2, Q_6 and Q_5 are all saturated, bringing the motor terminals close to ground. When $\overline{\text{FWD}}$ goes low, Q_3 and Q_2 stop conducting causing Q_1 to turn on. Current is driven through Q_1 and D_1 to the motor, and then through Q_5 to ground. The same is true when $\overline{\text{REV}}$ goes low and $\overline{\text{FWD}}$ is high. This circuit requires only a single power supply. Heat sinking is not required because the transistors are either on or off; they are never required to dissipate too much power.

A graph of the performance of this type of motor driver is shown in Figure B-4. The motor that was used is the one documented in Appendix C. Velocity measurements were made using the motor's built-in tachometer. It can be seen that the steady-state motor velocity is very linear with respect to the control byte. The graph does not go through the origin due to static friction.

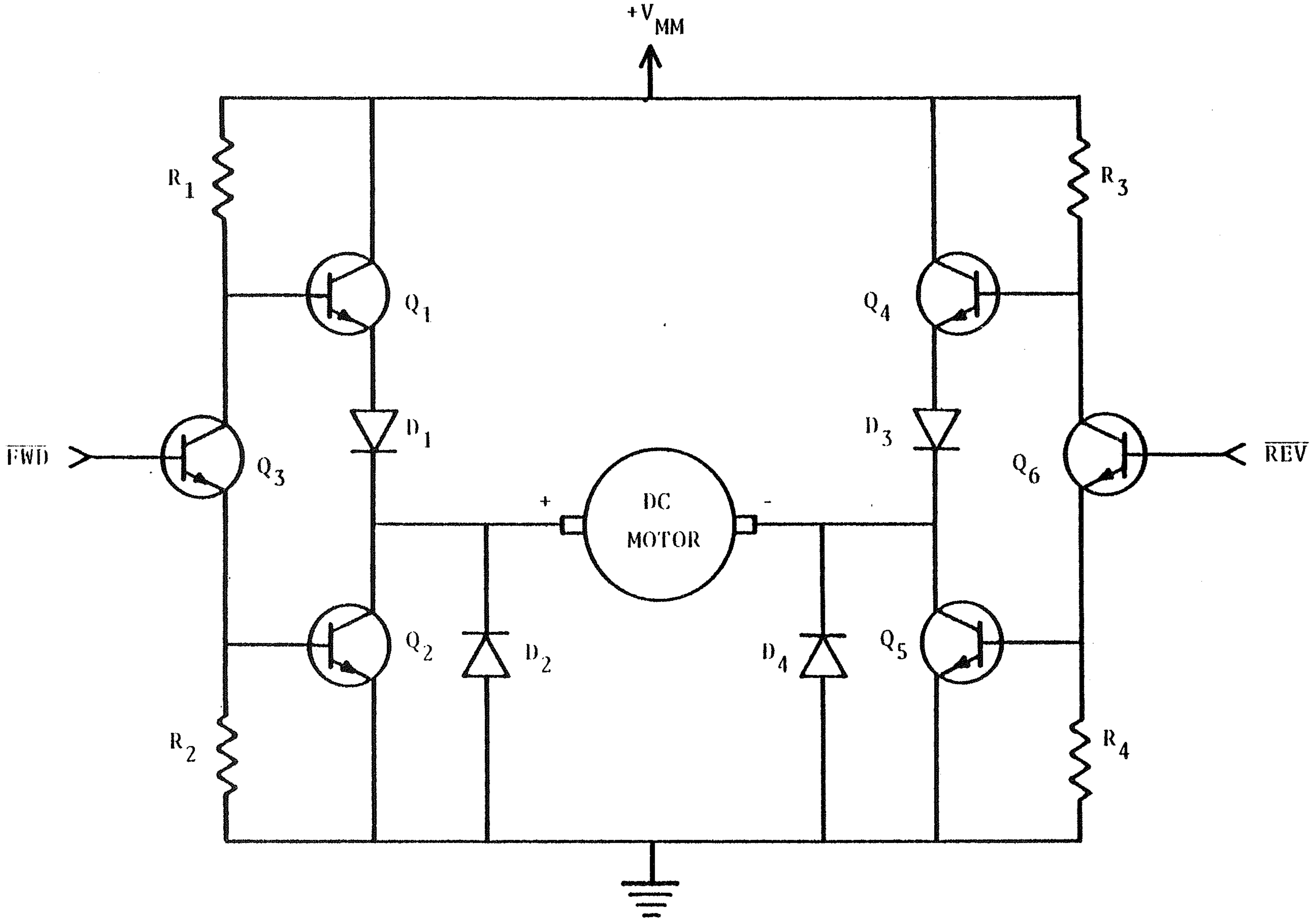


Figure B-3. Bidirectional Motor Driver.

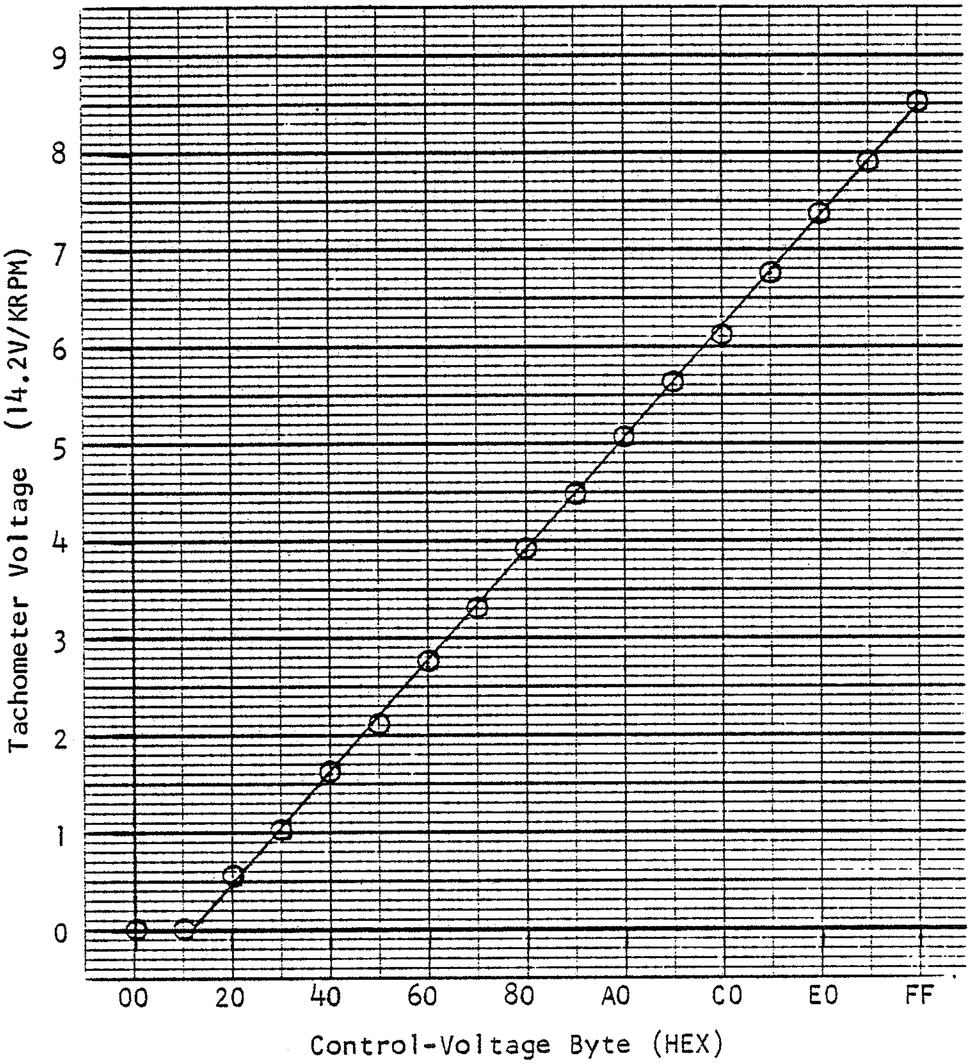


Figure B-4. Linearity of Pulsed Motor-Control.

APPENDIX C

NUMERICAL CONSTANTS

Specifications for the Electro-Craft Model 0660-06-011 Permanent-Magnet DC Servomotor[11]:

MOTOR

Torque Constant	K_τ	21 oz-in/A	±10%
Voltage Constant	K_V	16 V/KRPM or 0.1528 V-sec/rad	±10%
Armature Resistance	R	1.15 Ω	@25°C ±15%
Moment of Inertia	J	0.032 oz-in-sec^2	
Viscous Damping Factor	D	-1.0 oz-in/KRPM	
Static Friction Torque		7 oz-in	
Armature Inductance	L_a	4 mH	

BUILT-IN TACHOMETER

Voltage Gradient	K_T	14.2 V/KRPM	±10%
Terminal Resistance	R_T	750 Ω	@25°C ±15%
Armature Inductance	L_T	140 mH	
Ripple Amplitude		5% of DC output	
Ripple Frequency		11 Cycles/Revolution	

Using these values, the motor-model of equation (27) becomes:

$$\frac{d}{dt}\begin{bmatrix} i \\ \omega_m \\ \theta_m \end{bmatrix} = \begin{bmatrix} -287.5 & -38.2 & 0 \\ 647.9 & -0.2946 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} i \\ \omega_m \\ \theta_m \end{bmatrix} + \begin{bmatrix} 250 \\ 0 \\ 0 \end{bmatrix} V_m$$

The discrete version of this system is given in (46)-(48). The values of A_d and B_d for sampling rates of 10 Hz and 50 Hz are given here.

For T = 0.1 sec

$$A_d = \begin{bmatrix} 3.78 \times 10^{-7} & -4.76 \times 10^{-8} & 0 \\ 8.07 \times 10^{-7} & 7.36 \times 10^{-7} & 0 \\ 2.61 \times 10^{-2} & 1.16 \times 10^{-2} & 1 \end{bmatrix} \quad B_d = \begin{bmatrix} 2.966 \times 10^{-3} \\ 6.522 \\ 5.766 \times 10^{-1} \end{bmatrix}$$

For T = 0.02 sec

$$A_d = \begin{bmatrix} -1.048 \times 10^{-1} & -3.208 \times 10^{-2} & 0 \\ 5.441 \times 10^{-1} & 1.364 \times 10^{-1} & 0 \\ 2.252 \times 10^{-2} & 1.083 \times 10^{-2} & 1 \end{bmatrix} \quad B_d = \begin{bmatrix} 2.125 \times 10^{-1} \\ 5.630 \\ 5.972 \times 10^{-2} \end{bmatrix}$$

The units for these numbers are (-represents unitless):

$$[A_d] = \begin{bmatrix} - & \text{A-sec} & - \\ \text{1/A-sec} & - & - \\ \text{1/A} & \text{sec} & - \end{bmatrix} \qquad [B_d] = \begin{bmatrix} \text{A/V} \\ \text{rad/V-sec} \\ \text{rad/V} \end{bmatrix}$$

The design of a typical system is now shown. Choose continuous-system poles: $s_i = -20, -40 \pm j40$. These poles were chosen because of the critically-damped response they produce. This was determined with a graphing package included in LINSYS (Figure C-1).

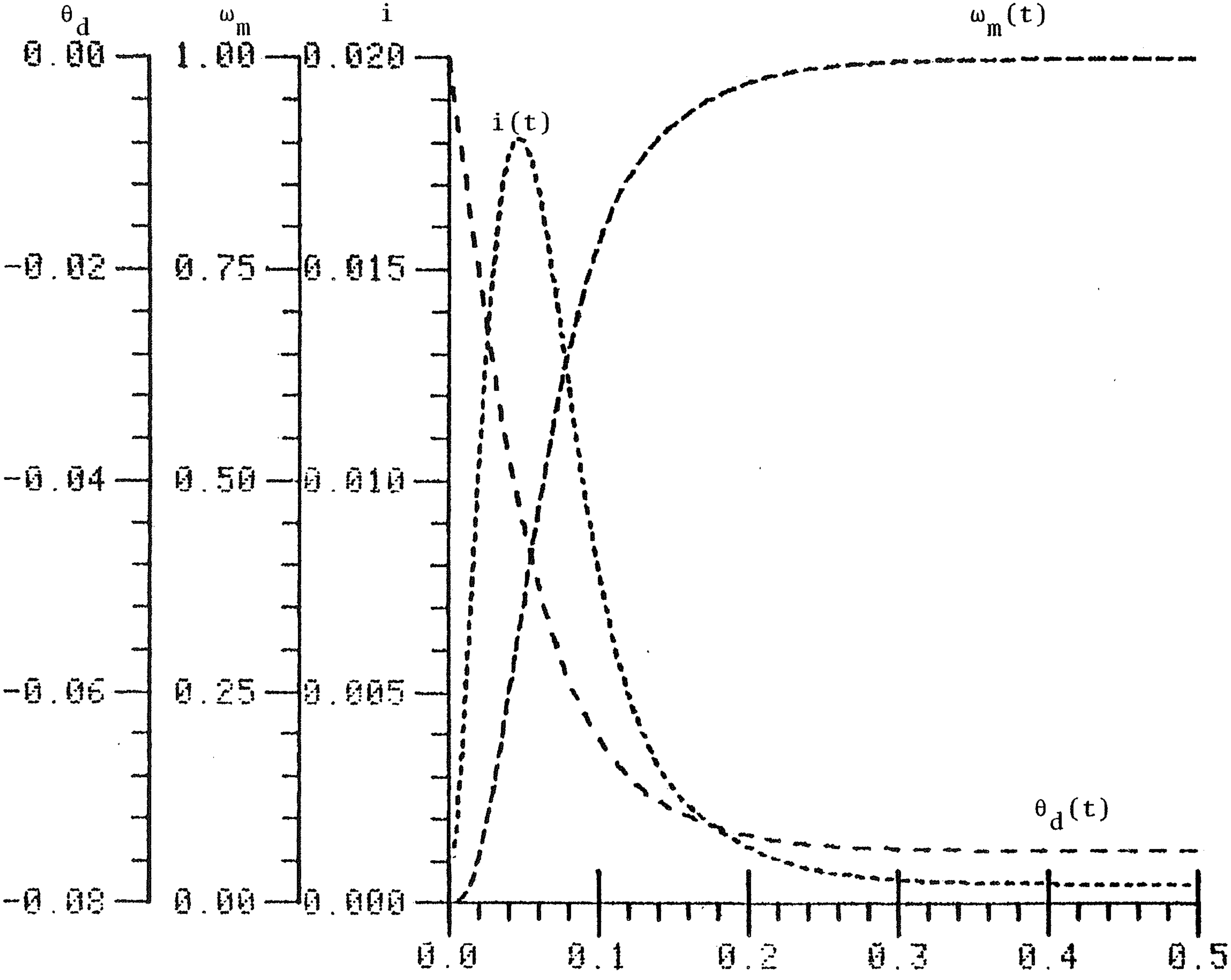


Figure C-1. Continuous-Time Closed-Loop Response of Model to a Unit-Step Input, s_i = -20, -40±j40.

Using T = 0.02 sec these poles transform to the following discrete-time poles: $z_i = 0.6703, 0.3131 \pm j0.3223$ through the transformation $z_i = e^{s_i T}$.

Given these values for z_i, and (A_d, B_d) above, LINSYS computes the feedback:

$$F_d = [7.6728 \times 10^{-1} \quad 3.3982 \times 10^{-2} \quad -1.4981]$$

The units are: $[F_d] = [\text{V/A} \quad \text{V-sec/rad} \quad \text{V/rad}]$

Observer design is accomplished in a similar way. Choose observer poles to be five times "faster" than the system:

$$s_i = -100, -200 \pm j200$$

giving $$z_i = 0.1353, -0.01197 \pm j0.01386$$

Given these values for z_i, and (A_d, C_d) where $C_d = [0 \ 0 \ 1]$, LINSYS computes the observer feedback matrix:

$$K_d = \begin{bmatrix} 2.7185 \times 10^{-1} \\ -1.3579 \\ 9.2024 \times 10^{-1} \end{bmatrix} \qquad \text{Units:} \quad [K_d] = \begin{bmatrix} \text{A/rad} \\ \text{1/sec} \\ - \end{bmatrix}$$

The overall discrete-observer matrix is (see equations (55) and (56)):

$$A_{od} = \begin{bmatrix} 5.8248 \times 10^{-2} & -2.4659 \times 10^{-2} & -5.9019 \times 10^{-1} \\ 4.8639 & 3.2772 \times 10^{-1} & -7.0764 \\ 6.8342 \times 10^{-2} & 1.2859 \times 10^{-2} & -9.7100 \times 10^{-3} \end{bmatrix}$$

$$\text{Units:} \quad [A_{od}] = \begin{bmatrix} - & \text{A-sec/rad} & \text{A/rad} \\ \text{rad/A-sec} & - & \text{1/sec} \\ \text{rad/A} & \text{sec} & - \end{bmatrix}$$

APPENDIX D

SYSTEM PERFORMANCE

The performance of the velocity control system is shown in Figure D-1. Figure D-1(a) shows how the system rapidly converges to the requested velocity. There is a small amount of velocity overshoot that did not occur in the continuous-time simulation (Figure C-1). This is due to inacuracies between the model and the physical system. These velocity measurements were made using the motor's built-in tachometer, which is why there is a ripple on the steady-state velocity.

The system's response to a varying load is seen in Figure D-1(b). At points A and C the motor was slowed down by adding friction to its shaft, but in each case the integral controller was able to correct for this and the motor resumed its original speed. Points B and D are where the extra drag was removed. The speed momentarily increased and then resumed correctly.

Responses of the simplified positioning system are shown in Figure D-2. It is slow to converge to a requested position, although its initial response is quite fast. A significant amount of ringing is produced for even the small value of C_p in Figure D-2(a). For values of C_p larger than used here, the system became unstable.

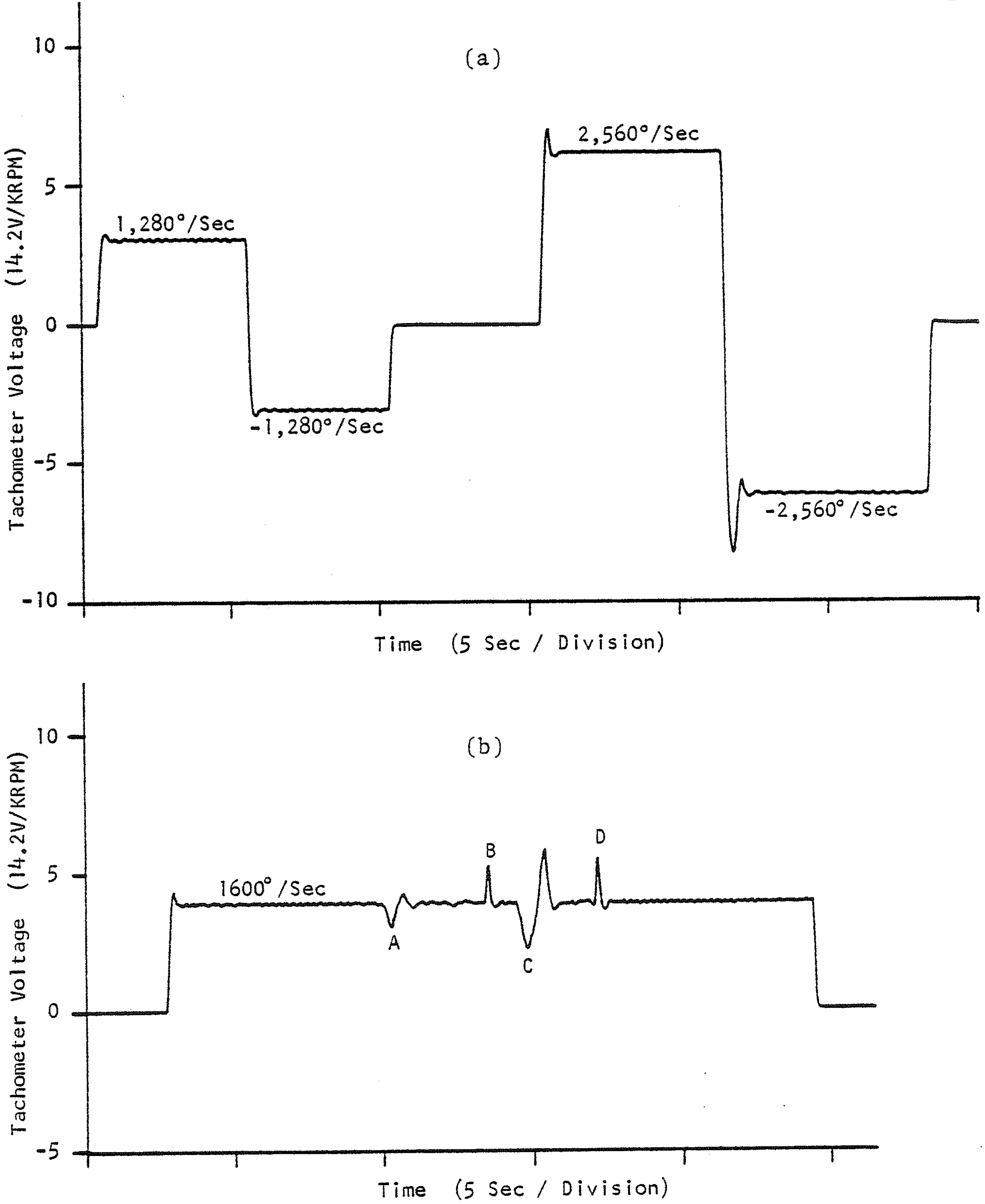


Figure D-1. Response of Velocity Controller.
(a) Step Inputs, (b) Varying Load.

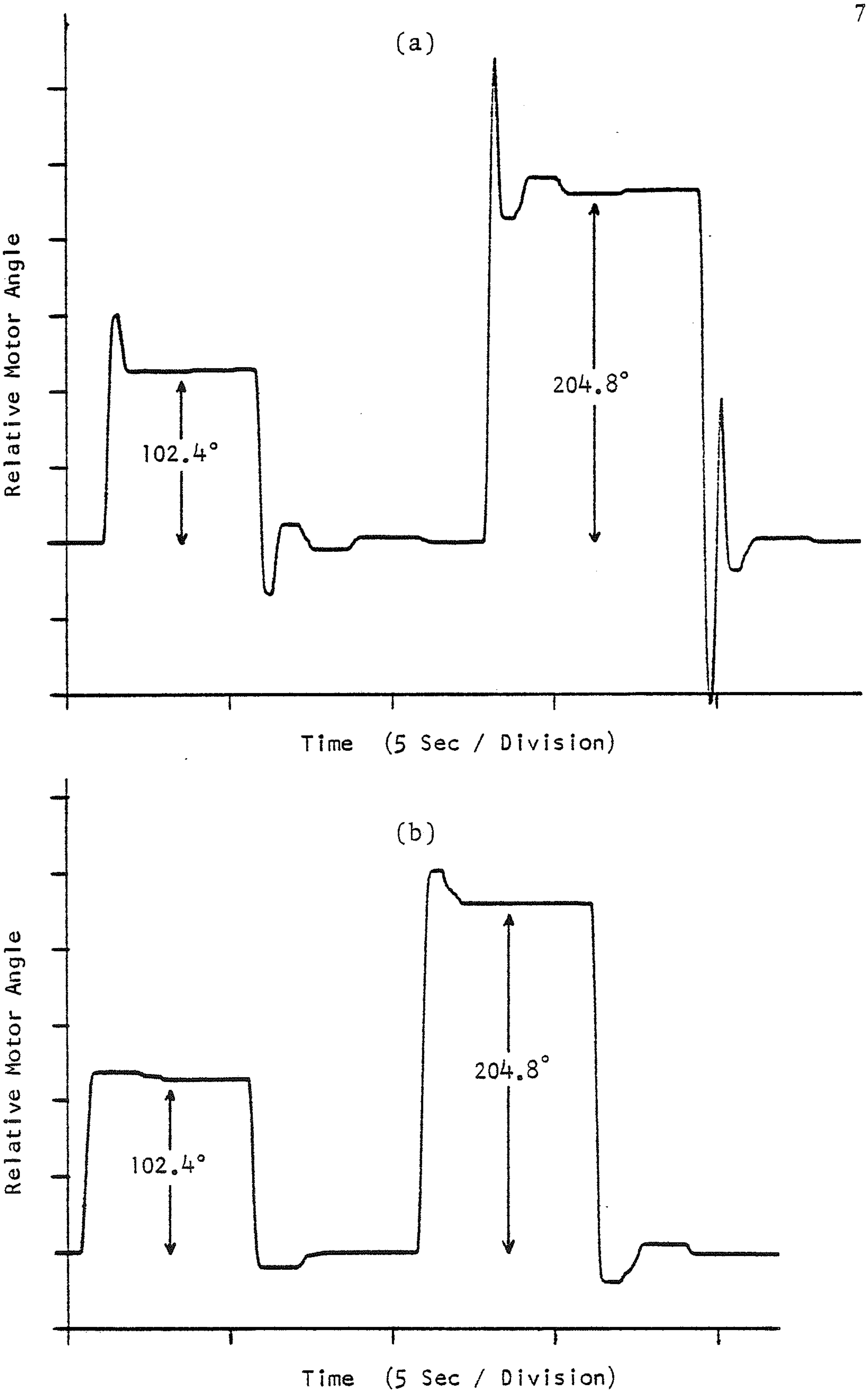


Figure D-2. Response of Simplified Position Controller to Step-Inputs. (a) C_p=3/16, (b) C_p=2/16.

APPENDIX E

PROGRAM LISTING

```
                                                            PAGE    1
CARD # LOC     CODE       CARD
  1                       ;
  2                       ;  THIS FILE CONTAINS THE DEFINITIONS OF ALL SYMBOLIC NAMES AND
  3                       ;  THE ALLOCATION FOR PAGE-ZERO MEMORY.
  4                       ;
  5                       NSTATE =   3       ; NUMBER OF SYSTEM STATES.
  6                       MVOLTS =   $A0     ; MAX POSSIBLE OUTPUT VOLTS (4.4 FORMAT, UNSIGNED).
  7                       THMAXH =   $38     ; HIGH BYTE OF 3599 (DEC) IN S13.2 FORMAT.
  8                       THMAXL =   $3C     ; LOW BYTE OF THE ABOVE.
  9                       ;   SYSTEM MASKS.
 10                       MSKCA1 =   $02     ; MASK FOR CA1 BIT IN IFR.
 11                       MSKT1  =   $40     ; MASK FOR T1 BIT IN IFR.
 12                       ;   REGISTERS IN VIA #1.
 13                       ORB1   =   $A000   ; OUTPUT REG B, MOTOR DIRECTION CONTROL.
 14                       ORA1   =   $A001   ; OUTPUT REG A, MOTOR VOLTAGE CONTROL.
 15                       DDRB1  =   $A002   ; DATA DIRECTION REG B.
 16                       DDRA1  =   $A003   ; DATA DIRECTION REG A.
 17                       ;   REGISTERS IN VIA #2.
 18                       IRB2   =   $A800   ; INPUT REG B, MOTOR ANGLE, HIGH BYTE.
 19                       IRA2   =   $A801   ; INPUT REG A, MOTOR ANGLE, LOW BYTE.
 20                       DDRB2  =   $A802   ; DATA DIRECTION REG B.
 21                       DDRA2  =   $A803   ; DATA DIRECTION REG A.
 22                       ACR2   =   $A80B   ; AUX CONTROL REG.
 23                       PCR2   =   $A80C   ; PERIPHERAL CONTROL REG.
 24                       ;   REGISTERS IN VIA #3.
 25                       T1CH3  =   $AC05   ; TIMER 1 COUNTER, HIGH.
 26                       T1LL3  =   $AC06   ; TIMER 1 LATCH, LOW.
 27                       T1LH3  =   $AC07   ; TIMER 1 LATCH, HIGH.
 28                       ACR3   =   $AC0B   ; AUX CONTROL REG.
 29                       PCR3   =   $AC0C   ; PERIPHERAL CONTROL REG.
 30                       IFR3   =   $AC0D   ; INTERRUPT FLAG REGISTER.
 31                       IER3   =   $AC0E   ; INTERRUPT ENABLE REGISTER.
 32                       ;   SYSTEM MONITOR ADDRESSES.
 33                       UIRQVC =   $A678   ; POINTER TO INTERRUPT HANDLER.
 34                       ACCESS =   $8B86   ; ROUTINE TO UNWRITE-PROTECT SYSTEM RAM.
 35                       GETKEY =   $88AF   ; ROUTINE TO READ FROM HEX KEYBOARD.
 36                       INBYTE =   $8109   ; ROUTINE TO READ 2 HEX DIGITS FROM KBD.
 37                       ;   INTERRUPT HANDLER ADDRESS.
 38                       INTADL =   $00
 39                       INTADH =   $03
 40                       ;
 41                       ;  PAGE-ZERO MEMORY ALLOCATION.
 42                       ;
 43   0000                *      =   $10
 44   0010     00 00      MQ     .WORD 0
 45   0012     00 00      AC     .WORD 0
 46   0014     00 00      MC     .WORD 0
 47   0016     00 00      SUM    .WORD 0,0    ; RESULT FROM VCMULT.
 47   0018     00 00
 48                       K      =   *          ; FEEDBACK GAIN MATRIX, S(-3).18 FORMAT.
 49   001A     00 00      K1     .WORD 0
 50   001C     00 00      K2     .WORD 0
 51   001E     00 00      K3     .WORD 0
 52   0020     00 00      VBIAS  .WORD 0         ; BIAS VOLTAGE TO CANCEL STATIC FRICTION, S11.4 FORMAT.
 53   0022     00         VOUT   .BYT  0         ; MAGNITUDE OF MOTOR CONTROL.
 54   0023     00         SOUT   .BYT  0         ; SIGN OF VOUT ( 00=CCW, 80=CW ).
 55   0024     00 00      CPOS   .WORD 0         ; POSITIONAL FEEDBACK CONSTANT.
 56                       ;  OVERALL A-MATRIX, S.15 FORMAT.
```

```
CARD # LOC      CODE        CARD
  57                        A01      =  *          ; ROW 1.
  58   0026     00 00       A011     .WORD 0
  59   0028     00 00       A012     .WORD 0
  60   002A     00 00       A013     .WORD 0
  61   002C     00 00       H1       .WORD 0       ; OBSERVER VECTOR, 1ST ELEMENT.
  62                        A02      =  *          ; ROW 2.
  63   002E     00 00       A021     .WORD 0
  64   0030     00 00       A022     .WORD 0
  65   0032     00 00       A023     .WORD 0
  66   0034     00 00       H2       .WORD 0       ; OBSERVER VECTOR, 2ND ELEMENT.
  67                        A03      =  *          ; ROW 3.
  68   0036     00 00       A031     .WORD 0
  69   0038     00 00       A032     .WORD 0
  70   003A     00 00       A033     .WORD 0
  71   003C     00 00       H3       .WORD 0       ; OBSERVER VECTOR, 3RD ELEMENT.
  72   003E     20 4E       TPER     .WORD 20000   ; TIMER CONST FOR T = 0.02 SEC.
  73                        ;   OBSERVER STATE VECTOR (XHAT), S13.2 FORMAT.
  74                        XH       =  *
  75   0040     00 00       XH1      .WORD 0
  76   0042     00 00       XH2      .WORD 0
  77   0044     00 00       XH3      .WORD 0
  78   0046     00 00       DELTH    .WORD 0       ; INPUT TO OBSERVER, Y = DELTH = THM - THF .
  79   0048     00 00       THM      .WORD 0       ; MOTOR THETA, READ FROM IRA2,IRB2.
  80   004A     00 00       THF      .WORD 0       ; 'FOLLOWING ANGLE', USED FOR INTEGRAL CONTROL.
  81                        ;   NEW OBSERVER STATE VECTOR, XHAT(K+1), S13.2 FORMAT.
  82                        XHKP1    =  *
  83   004C     00 00       XH1KP1   .WORD 0
  84   004E     00 00       XH2KP1   .WORD 0
  85   0050     00 00       XH3KP1   .WORD 0
  86                        ;
  87   0052     00 00       THREF    .WORD 0       ; DESIRED ANGLE FOR POSITION CONTROL.
  88   0054     00 00       THERR    .WORD 0       ; ANGULAR ERROR FORPOSITION CONTROL.
  89   0056     00 00       TWR      .WORD 0       ; VELOCITY SETPOINT, IN 'TICKS'/T , S13.2 FORMAT.
  90   0058     00          TWRS     .BYT  0,0,0   ; SAME AS TWR, BUT SHIFTED SO IT LINES UP W/ SUM.
  90   0059     00
  90   005A     00
  91   005B     00          TEMP     .BYT  0       ; USED BY MULT.
  92   005C     00          TEMP2    .BYT  0       ; USED BY VCMULT.
  93   005D     00          TEMP3    .BYT  0       ; USED BY DRIVER.
  94   005E     00          PFLAG    .BYT  0       ; POSITION/VELOCITY CONTROL FLAG.
  95   005F     00          DIM      .BYT  0       ; DIMENSION OF VECTORS.
  96   0060     00          SMQ      .BYT  0       ; MQ STORAGE FOR SCALE ROUTINE.
  97   0061     CC          SC       .BYT  $CC     ; SCALE FACTOR (25.5 DEC), 5.3 FORMAT, UNSIGNED.
  98                        ;
  99   0062                 *        =  $0200
 100                        ;
 101                        ;   DRIVER - MAIN PROGRAM FOR THE INTERRUPT-DRIVEN PID CONTROLLER.
 102                        ;   THIS ROUTINE INITIALIZES THE I/O CONFIGURATION, AND MONITORS
 103                        ;   THE HEX KEYBOARD FOR COMMANDS WHILE WAITING FOR INTERRUPTS.
 104   0200     20 86 8B    DRIVER   JSR ACCESS    ; UN-WRITE PROTECT SYSTEM RAM.
 105   0203     78                   SEI           ; DISABLE INTERRUPTS INITIALLY.
 106                        ;   STORE ADDRESS OF INTERRUPT HANDLER.
 107   0204     A9 00                LDA #INTADL
 108   0206     8D 7E A6             STA UIRQVC
 109   0209     A9 03                LDA #INTADH
 110   020B     8D 7F A6             STA UIRQVC+1
 111                        ;   CLEAR THE CONTROL OUTPUT REGISTERS AND INIT THE DATA DIRECTION REGS.
```

```
CARD # LOC    CODE        CARD
 112   020E   A9 FF              LDA #$FF      ; SET THE VOUT REG
 113   0210   8D 03 A0           STA DDRA1     ; TO ALL OUTPUTS.
 114   0213   A9 80              LDA #$80      ; SET B7 OF THE
 115   0215   8D 02 A0           STA DDRB1     ; SOUT REG TO OUTPUT.
 116   0218   A9 00              LDA #0
 117   021A   8D 01 A0           STA ORA1      ; ZERO VOLTS TO THE MOTOR.
 118   021D   8D 00 A0           STA ORB1      ; CCW DIRECTION.
 119                       ;  SET THE DATA DIRECTION REGS FOR THE MOTOR ANGLE TO ALL INPUTS.
 120   0220   A9 00              LDA #0
 121   0222   8D 03 A8           STA DDRA2
 122   0225   8D 02 A8           STA DDRB2
 123                       ;  INIT THE 'FOLLOWING ANGLE' TO THE NEGATIVE OF THE MOTOR ANGLE.
 124   0228   38                 SEC
 125   0229   A9 00              LDA #$00
 126   022B   ED 01 A8           SBC IRA2
 127   022E   85 4A              STA THF
 128   0230   A9 00              LDA #$00
 129   0232   ED 00 A8           SBC IRB2
 130   0235   85 4B              STA THF+1
 131                       ;  SET UP THE ZERO-REF INTERRUPT (CA1=3) TO TRIGGER ON LEADING EDGE.
 132   0237   A9 01              LDA #$01
 133   0239   8D 0C AC           STA PCR3
 134                       ;  SET TIMER 1 IN VIA #3 TO FREE-RUNNING MODE.
 135   023C   A9 40              LDA #$40
 136   023E   8D 0B AC           STA ACR3
 137                       ;  INIT THE TIMER 1 LATCH AND START THE TIMER.
 138   0241   A5 3E              LDA TPER
 139   0243   8D 06 AC           STA T1LL3
 140   0246   A5 3F              LDA TPER+1
 141   0248   8D 07 AC           STA T1LH3
 142   024B   8D 05 AC           STA T1CH3         ; START THE TIMER.
 143                       ; CLEAN INTERRUPT FLAGS FOR T1 AND CA1, THEN ENABLE INTERRUPTS.
 144   024E   A9 C2              LDA #$C2
 145   0250   8D 0D AC           STA IFR3
 146   0253   8D 0E AC           STA IER3
 147   0256   58                 CLI
 148                       ;
 149                       ;  THIS IS THE MAIN WAIT LOOP.  WHILE WAITING FOR AN INTERRUPT,
 150                       ;  READ AND CARRY OUT COMMANDS FROM THE HEX KEYBOARD.
 151                       ;
 152   0257   20 AF 88    WAITLP JSR GETKEY
 153   025A   C9 2B              CMP #$2B
 154   025C   F0 0A              BEQ POSV
 155   025E   C9 2D              CMP #$2D
 156   0260   F0 0A              BEQ NEGV
 157   0262   C9 47              CMP #$47      ; 'G' KEY.
 158   0264   F0 40              BEQ SETANG
 159   0266   D0 EF              BNE WAITLP
 160                       ;
 161                       ;  READ A NEW VELOCITY SETPOINT FROM THE HEX KEYBOARD.  IF IT'S A
 162                       ;  NEGATIVE MAGNITUDE, TAKE ITS 2'S COMPLEMENT.  THEN STORE IT
 163                       ;  IN TWN AND THRS.
 164                       ;
 165   0268   A9 00       POSV   LDA #$00
 166   026A   10 02              BPL GETVEL
 167   026C   A9 FF       NEGV   LDA #$FF
 168   026E   85 5D       GETVEL STA TEMP3
```

```
CARD # LOC    CODE         CARD
  169  0270   20 D9 81          JSR INBYTE      ; READ 2 CHARS AND ASSEMBLE INTO A BYTE.
  170  0273   B0 E2             BCS WAITLP
  171  0275   AA                TAX
  172  0276   20 D9 81          JSR INBYTE      ; READ LOW-ORDER VELOCITY BYTE.
  173  0279   B0 DC             BCS WAITLP
  174  027B   24 5D             BIT TEMP3
  175  027D   10 0D             BPL SAVW
  176                     ;  TAKE 2'S COMPLEMENT OF THE DATA THAT WAS INPUTTED.
  177  027F   49 FF             EOR #$FF
  178  0281   69 01             ADC #1
  179  0283   A8                TAY
  180  0284   8A                TXA
  181  0285   49 FF             EOR #$FF
  182  0287   69 00             ADC #0
  183  0289   AA                TAX
  184  028A   98                TYA
  185  028B   38                SEC             ; C IS NOW THE SIGN OF THE VELOCITY SETPOINT.
  186                     ;  STORE THE NEW SETPOINT IN TWR AND TWRS.  SINCE WE CAN BE SCREWED UP
  187                     ;  IF WE RECEIVE AN INTERRUPT NOW, DISABLE INTERRUPTS.
  188  028C   78          SAVW  SEI
  189  028D   86 57             STX TWR+1
  190  028F   85 56             STA TWR
  191  0291   A8                TAY
  192  0292   8A                TXA
  193  0293   6A                .BYT $6A  ; ROR A
  194  0294   85 5A             STA TWRS+2
  195  0296   98                TYA
  196  0297   6A                .BYT $6A  ; ROR A
  197  0298   85 59             STA TWRS+1
  198  029A   A9 00             LDA #0
  199  029C   6A                .BYT $6A  ; ROR A
  200  029D   85 58             STA TWRS
  201  029F   A9 00             LDA #0
  202  02A1   85 5E             STA PFLAG  ; SWITCH TO VELOCITY CONTROL.
  203  02A3   58                CLI
  204  02A4   90 81             BCC WAITLP
  205                     ;   READ AND STORE A REQUESTED MOTOR ANGLE, THEN SWITCH TO POSITION CONTROL.
  206  02A6   20 D9 81    SETANG JSR INBYTE
  207  02A9   B0 AC             BCS WAITLP
  208  02AB   AA                TAX
  209  02AC   20 D9 81          JSR INBYTE
  210  02AF   B0 A6             BCS WAITLP
  211  02B1   78                SEI
  212  02B2   86 53             STX THREF+1
  213  02B4   85 52             STA THREF
  214  02B6   A9 80             LDA #$80
  215  02B8   85 5E             STA PFLAG   ; SWITCH TO POSITION CONTROL.
  216  02BA   58                CLI
  217  02BB   30 9A             BMI WAITLP  ; B7 OF A IS STILL 1.
  218  02BD               *     =   $0300
  219                     ;
  220                     ;   INTHDL - MAIN INTERRUPT HANDLER.
  221                     ;   THIS ROUTINE HANDLES THE IRQ INTERRUPTS.  PRESENTLY THERE ARE
  222                     ;   TWO TYPES OF INT. TO HANDLE:
  223                     ;      ZREF - ZERO REFERENCE FROM SHAFT ENCODER, ON CA1 LINE OF VIA #3
  224                     ;      T1   - INTERVAL TIMER FOR DISCRETE CONTROL, T1 IS IN VIA #3.
  225                     ;             FOR A T1 INTERRUPT WE BRANCH TO THE ROUTINE CONTRL.
```

```
CARD # LOC      CODE          CARD
  226                         ;
  227   0300    48            INTHDL PHA
  228   0301    A9 02                LDA #$02
  229   0303    2C 0D AC             BIT IFR
  230   0306    D0 05                BNE ZREF
  231   0308    50 4B                BVC BYE1
  232   030A    4C 57 03             JMP CONTRL
  233   030D    A9 02         ZREF   LDA #$02
  234   030F    8D 0D AC             STA IFR       ; TURN OFF THE INTERRUPT FLAG.
  235                         ;  DETERMINE THE DIRECTION OF MOTION BY EXAMINING B7 OF THE POSITION REG.
  236   0312    AD 00 A8             LDA IRB2
  237   0315    0A                   ASL A
  238   0316    B0 20                BCS CW
  239                         ;  CORRECT THE ( = FOLLOWING ANGLE ) BY ADDING 3599.
  240   0318    A5 4A         CCW    LDA THF
  241   031A    69 3C                ADC #$3C
  242   031C    85 4A                STA THF
  243   031E    A5 4B                LDA THF+1
  244   0320    69 38                ADC #$38
  245   0322    85 4B                STA THF+1
  246                         ;  IF POSITIONING IS IN PROGRESS, CORRECT THREF BY SUBTRACTING 3600.
  247   0324    24 5E                BIT PFLAG
  248   0326    10 2D                BPL BYE1
  249   0328    38                   SEC
  250   0329    A5 52                LDA THREF
  251   032B    E9 40                SBC #$40
  252   032D    85 52                STA THREF
  253   032F    A5 53                LDA THREF+1
  254   0331    E9 38                SBC #$38
  255   0333    85 53                STA THREF+1
  256   0335    18                   CLC
  257   0336    90 1D                BCC BYE1
  258                         ;  CORRECT THF ( = FOLLOWING ANGLE ) BY SUBTRACTING 3599.
  259   0338    A5 4A         CW     LDA THF
  260   033A    E9 3C                SBC #$3C
  261   033C    85 4A                STA THF
  262   033E    A5 4B                LDA THF+1
  263   0340    E9 38                SBC #$38
  264   0342    85 4B                STA THF+1
  265                         ;  IF POSITIONING IS IN PROGRESS, CORRECT THREF BY ADDING 3600.
  266   0344    24 5E                BIT PFLAG
  267   0346    10 0D                BPL BYE1
  268   0348    18                   CLC
  269   0349    A5 52                LDA THREF
  270   034B    69 40                ADC #$40
  271   034D    85 52                STA THREF
  272   034F    A5 53                LDA THREF+1
  273   0351    69 38                ADC #$38
  274   0353    85 53                STA THREF+1
  275   0355    68            BYE1   PLA
  276   0356    40                   RTI
  277                         ;
  278                         ;  CONTRL - MAIN CONTROL PROGRAM
  279                         ;  THIS ROUTINE SERVICES THE TIMER INTERRUPTS. IT COMPUTES A NEW
  280                         ;  VALUE FOR THE MOTOR CONTROL AND THEN UPDATES THE OBSERVER
  281                         ;  STATE VARIABLES.
  282                         ;
```

```
CARD # LOC      CODE       CARD
  283  Ø357   AD Ø1 A8  CONTRL LDA IRA2       ; READ MOTOR ANGLE AND
  284  Ø35A   85 48            STA THM        ; STORE IN MEMORY.
  285  Ø35C   AD ØØ A8         LDA IRB2
  286  Ø35F   85 49            STA THM+1
  287                   ; RESET THE INTERRUPT FLAG.
  288  Ø361   A9 4Ø            LDA #MSKT1
  289  Ø363   8D ØD AC         STA IFR3
  290                   ; SAVE REGISTERS.
  291  Ø366   8A               TXA
  292  Ø367   48               PHA
  293  Ø368   98               TYA
  294  Ø369   48               PHA
  295                   ;
  296                   ;  COMPUTE NEW VALUE FOR CONTROL.
  297                   ;
  298  Ø36A   A2 1A            LDX #K         ; LOAD POINTER TO FEEDBACK VECTOR.
  299  Ø36C   AØ 4Ø            LDY #XH        ; LOAD POINTER TO OBSERVER STATE VECTOR.
  300  Ø36E   A9 Ø3            LDA #NSTATE    ; LOAD DIMENSION OF
  301  Ø37Ø   85 5F            STA DIM        ; VECTORS.
  302  Ø372   2Ø 7F Ø4         JSR VCMULT     ; MULTIPLY THE VECTORS.
  303                   ;
  304                   ;  ADD A (NONLINEAR) BIAS VOLTAGE TO HELP CANCEL STATIC FRICTION.
  305                   ;
  306                   ;  TEST XH2 (OBSERVER VELOCITY) TO SEE IF WE THINK WE ARE MOVING.
  307  Ø375   A5 43            LDA XH2+1
  308  Ø377   DØ 1E            BNE MOVING
  309  Ø379   A5 42            LDA XH2
  310  Ø37B   DØ 1A            BNE MOVING
  311  Ø37D   7Ø 5B            BVS OVFLW      ; SEE IF VCMULT PRODUCED AN OVERFLOW.
  312                   ;  TEST SUM (THE COMPUTED CONTROL) TO SEE IF IT IS ZERO.
  313  Ø37F   A5 19            LDA SUM+3
  314  Ø381   DØ ØE            BNE TSTSGN
  315  Ø383   A5 18            LDA SUM+2
  316  Ø385   DØ ØA            BNE TSTSGN
  317  Ø387   A5 17            LDA SUM+1
  318  Ø389   DØ Ø6            BNE TSTSGN
  319  Ø38B   A5 16            LDA SUM
  320  Ø38D   DØ Ø2            BNE TSTSGN
  321  Ø38F   FØ 2A            BEQ NOVFLW
  322                   ;   IF SUM > Ø WE ADD A BIAS VOLTAGE, ELSE WE SUBTRACT ONE.
  323  Ø391   24 19     TSTSGN BIT SUM+3
  324  Ø393   1Ø Ø6            BPL ADDVB
  325  Ø395   3Ø 15            BMI SUBVB
  326                   ;   HERE WE ARE MOVING, SO ADD THE BIAS IN THE DIRECTION OF MOTION.
  327  Ø397   24 49     MOVING BIT THM+1      ; HIGH BIT OF MOTOR ANGLE GIVES DIRECTION OF MOTION.
  328  Ø399   3Ø 11            BMI SUBVB
  329                   ;   ADD THE BIAS VOLTAGE TO THE RELEVANT BYTES OF SUM.
  330  Ø39B   18        ADDVB  CLC
  331  Ø39C   A5 18            LDA SUM+2
  332  Ø39E   65 2Ø            ADC VBIAS
  333  Ø3AØ   85 18            STA SUM+2
  334  Ø3A2   A5 19            LDA SUM+3
  335  Ø3A4   65 21            ADC VBIAS+1
  336  Ø3A6   85 19            STA SUM+3
  337  Ø3A8   5Ø 11            BVC NOVFLW
  338  Ø3AA   7Ø 2E            BVS OVFLW
  339                   ;  SUBTRACT A BIAS VOLTAGE FROM THE RELEVANT BYTES OF SUM.
```

```
CARD # LOC     CODE        CARD
 340   03AC    38         SUBVB   SEC
 341   03AD    A5 18              LDA SUM+2
 342   03AF    E5 20              SBC VBIAS
 343   03B1    85 18              STA SUM+2
 344   03B3    A5 19              LDA SUM+3
 345   03B5    E5 21              SBC VBIAS+1
 346   03B7    85 19              STA SUM+3
 347   03B9    70 1F              BVS OVFLW
 348                      ;   STORE THE SIGN OF THE CONTROL VOLTAGE IN SOUT, AND ITS MAGNITUDE
 349                      ;   IN SUM ( TAKE COMPLEMENT IF NEGATIVE ).
 350   03BB    85 23      NOVFLW  STA SOUT
 351   03BD    10 0D              BPL NONEG
 352   03BF    38                 SEC
 353   03C0    A9 00              LDA #$00
 354   03C2    E5 18              SBC SUM+2
 355   03C4    85 18              STA SUM+2
 356   03C6    A9 00              LDA #$00
 357   03C8    E5 19              SBC SUM+3
 358   03CA    85 19              STA SUM+3
 359                      ;   TEST THE RELEVANT BYTES OF SUM TO SEE IF THEY EXCEED THE
 360                      ;   MAXIMUM ALLOWABLE CONTROL VOLTAGE (MVOLTS).
 361   03CC    D0 0E      NONEG   BNE MAXV       ; HIGH BYTE NOT = 0, SO IT MUST BE TOO BIG.
 362   03CE    A5 18              LDA SUM+2
 363   03D0    C9 A0              CMP #MVOLTS
 364   03D2    B0 08              BCS MAXV
 365                      ;   SCALE THE CONTROL AND STORE IN VOUT.
 366   03D4    20 6A 04           JSR SCALE
 367   03D7    18                 CLC
 368   03D8    90 06              BCC OUTV
 369                      ;   HERE WE HANDLE AN OVERFLOW CONDITION OR THE CONTROL BEING TOO BIG.
 370                      OVFLW   ; ROR SOUT     ; MOVE C (THE OVERFLOW SIGN) TO SOUT.
 371   03DA    66                 .BYT $66,SOUT
 371   03DB    23
 372   03DC    A9 FF      MAXV    LDA #$FF
 373   03DE    85 22              STA VOUT
 374                      ;   OUTPUT THE CONTROL VOLTAGE MAGNITUDE AND DIRECTION, AFTER
 375                      ;   CLEARING THE EXTRA 'GARBAGE' IN SOUT.
 376   03E0    A5 23      OUTV    LDA SOUT
 377   03E2    29 80              AND #$80
 378   03E4    A6 22              LDX VOUT
 379   03E6    8E 01 A0           STX ORA1
 380   03E9    8D 00 A0           STA ORB1
 381   03EC    85 23              STA SOUT
 382                      ;
 383                      ;   COMPUTE THE NEW OBSERVER STATE VECTOR, XHKP1 .
 384                      ;
 385                      ;   FIRST COMPUTE DELTH = THM + THF .
 386   03EE    18                 CLC
 387   03EF    A5 48              LDA THM
 388   03F1    65 4A              ADC THF
 389   03F3    85 46              STA DELTH
 390   03F5    A5 49              LDA THM+1
 391   03F7    29 7F              AND #$7F          ; CLEAR THE DIRECTION
 392   03F9    85 49              STA THM+1         ; FLAG (B7 OF HIGH BYTE).
 393   03FB    65 4B              ADC THF+1
 394   03FD    85 47              STA DELTH+1
 395                      ;   COMPUTE THE 1ST NEW STATE (XH1KP1) .
```

```
CARD # LOC    CODE          CARD
 396   03FF   A2 26               LDX #AD1
 397   0401   A0 40               LDY #XH
 398   0403   A9 04               LDA #NSTATE+1
 399   0405   85 5F               STA DIM
 400   0407   20 7F 04            JSR VCHULT
 401   040A   A2 4C               LDX #XH1KP1    ; STORE IN XH1KP1
 402   040C   20 3B 05            JSR CLIP       ; AFTER CLIPPING.
 403                        ;  COMPUTE THE 2ND NEW STATE (XH2KP1) .
 404   040F   A2 2E               LDX #AD2
 405   0411   A0 40               LDY #XH
 406   0413   A9 04               LDA #NSTATE+1
 407   0415   85 5F               STA DIM
 408   0417   20 7F 04            JSR VCHULT
 409   041A   A2 4E               LDX #XH2KP1
 410   041C   20 3B 05            JSR CLIP
 411                        ;  COMPUTE THE 3RD NEW STATE (XH3KP1) .
 412   041F   A2 36               LDX #AD3
 413   0421   A0 40               LDY #XH
 414   0423   A9 04               LDA #NSTATE+1
 415   0425   85 5F               STA DIM
 416   0427   20 7F 04            JSR VCHULT
 417                        ;  SUBTRACT THE CORRECTLY-SHIFTED DISTURBANCE FROM THIS STATE.
 418   042A   38                  SEC
 419   042B   A5 17               LDA SUM+1
 420   042D   E5 58               SBC TWR8
 421   042F   85 17               STA SUM+1
 422   0431   A5 18               LDA SUM+2
 423   0433   E5 59               SBC TWR8+1
 424   0435   85 18               STA SUM+2
 425   0437   A5 19               LDA SUM+3
 426   0439   E5 5A               SBC TWR8+2
 427   043B   85 19               STA SUM+3
 428   043D   50 03               BVC OK1
 429   043F   20 C2 04            JSR OVFLOW
 430                        ; CLIP AND STORE IN XH3KP1 .
 431   0442   A2 50         OK1   LDX #XH3KP1
 432   0444   20 3B 05            JSR CLIP
 433                        ;
 434                        ;  SEE IF IN POSITION-CONTROL MODE,IF SO ENTER POSITION CONTROLLER.
 435                        ;
 436   0447   24 5E               BIT PFLAG
 437   0449   10 03               BPL NOPOS
 438   044B   20 79 05            JSR POSITN
 439                        NOPOS
 440                        ;
 441                        ;  COMPUTE THF = THF - TWR
 442                        ;
 443   044E   38                  SEC
 444   044F   A5 4A               LDA THF
 445   0451   E5 56               SBC TWR
 446   0453   85 4A               STA THF
 447   0455   A5 4B               LDA THF+1
 448   0457   E5 57               SBC TWR+1
 449   0459   85 4B               STA THF+1
 450                        ;
 451                        ;  MOVE XHAT( K+1 ) TO XHAT( K )  ( XHKP1 TO XH ).
 452                        ;
```

```
CARD # LOC      CODE       CARD
  453  045B   A2 05               LDX #$05
  454  045D   B5 4C      LP1      LDA XHKP1,X
  455  045F   95 40               STA XH,X
  456  0461   CA                  DEX
  457  0462   10 F9               BPL LP1
  458                    ;
  459                    ;  RESTORE REGISTERS, ENABLE INTERRUPTS, AND RETURN.
  460                    ;
  461  0464   68                  PLA
  462  0465   A8                  TAY
  463  0466   68                  PLA
  464  0467   AA                  TAX
  465  0468   68                  PLA
  466  0469   40                  RTI
  467                    ;
  468                    ;  SCALE - MULTIPLY THE NUMBER IN A BY A SCALE FACTOR (SC) AND
  469                    ;  STORE IN VOUT.
  470                    ;
  471  046A   85 60      SCALE    STA SMQ
  472  046C   A9 00               LDA #$00
  473  046E   A2 08               LDX #8
  474                    LP2      ; ROR SMQ
  475  0470   66                  .BYT $66,SMQ
  475  0471   60
  476  0472   90 03               BCC SHIFT1
  477  0474   18                  CLC
  478  0475   65 61               ADC SC
  479                    SHIFT1   ; ROR A
  480  0477   6A                  .BYT $6A
  481  0478   CA                  DEX
  482  0479   D0 F5               BNE LP2
  483  047B   2A                  ROL A            ; RESTORE 8 BITS OF PRECISION,
  484  047C   85 22               STA VOUT         ; AND STORE WITH NO SIGN BIT.
  485  047E   60                  RTS
  486                    ;
  487                    ;  VCMULT - VECTOR MULTIPLY
  488                    ;
  489  047F   86 5C      VCMULT   STX TEMP2        ; SAVE PTR TO VECTOR1 SINCE X IS MODIFIED BY MULT
  490  0481   20 E0 04            JSR MULT
  491  0484   A5 10               LDA MQ
  492  0486   85 16               STA SUM
  493  0488   A5 11               LDA MQ+1
  494  048A   85 17               STA SUM+1
  495  048C   A5 12               LDA AC
  496  048E   85 18               STA SUM+2
  497  0490   A5 13               LDA AC+1
  498  0492   85 19               STA SUM+3
  499  0494   C6 5F      NEXT     DEC DIM
  500  0496   F0 29               BEQ DONE
  501  0498   A6 5C               LDX TEMP2
  502                    ;  GO TO NEXT ELEMENT OF EACH VECTOR (EACH ELEMENT IS TWO BYTES).
  503  049A   E8                  INX
  504  049B   E8                  INX
  505  049C   C8                  INY
  506  049D   C8                  INY
  507  049E   86 5C               STX TEMP2
  508  04A0   20 E0 04            JSR MULT
```

```
CARD # LOC     CODE        CARD
 509   04A3   18                  CLC
 510                        ; ADD PRODUCT OF THIS ELEMENT-PAIR TO SUM.
 511   04A4   A5 16               LDA SUM
 512   04A6   65 10               ADC MQ
 513   04A8   85 16               STA SUM
 514   04AA   A5 17               LDA SUM+1
 515   04AC   65 11               ADC MQ+1
 516   04AE   85 17               STA SUM+1
 517   04B0   A5 18               LDA SUM+2
 518   04B2   65 12               ADC AC
 519   04B4   85 18               STA SUM+2
 520   04B6   A5 19               LDA SUM+3
 521   04B8   65 13               ADC AC+1
 522   04BA   85 19               STA SUM+3
 523   04BC   50 06               BVC NEXT
 524   04BE   20 C2 04            JSR OVFLOW
 525   04C1   60           DONE   RTS
 526                       ;
 527                       ;  OVFLOW - OVERFLOW HANDLER.
 528                       ;  IF THERE WAS A POSITIVE OR NEGATIVE OVERFLOW FROM ADDING TO SUM, SET IT
 529                       ;  TO THE MAXIMUM POSITIVE OR NEGATIVE VALUE, RESPECTIVELY.
 530                       ;
 531   04C2   B0 0D        OVFLOW BCS MAXNEG
 532   04C4   A9 FF               LDA #$FF
 533   04C6   85 16               STA SUM
 534   04C8   85 17               STA SUM+1
 535   04CA   85 18               STA SUM+2
 536   04CC   A9 7F               LDA #$7F
 537   04CE   85 19               STA SUM+3
 538   04D0   60                  RTS
 539   04D1   A9 01        MAXNEG LDA #$01
 540   04D3   85 16               STA SUM
 541   04D5   A9 00               LDA #$00
 542   04D7   85 17               STA SUM+1
 543   04D9   85 18               STA SUM+2
 544   04DB   A9 80               LDA #$80
 545   04DD   85 19               STA SUM+3
 546   04DF   60                  RTS
 547                       ;
 548                       ;  MULT - 4-QUADRANT MUTIPLY.  THIS ROUTINE MULTIPLIES TWO 2-BYTE,
 549                       ;  2'S COMPLEMENT NUMBERS TO GIVE A 4-BYTE, 2'S COMPLEMENT PRODUCT.
 550                       ;
 551   04E0   B5 00        MULT     LDA 0,X       ; GET 2 BYTES OF MULTIPLIER
 552   04E2   85 10                 STA MQ        ; AND STORE
 553   04E4   B5 01                 LDA 1,X       ; IN MQ 'REGISTER'
 554   04E6   85 11                 STA MQ+1
 555   04E8   B9 00 00              LDA 0,Y       ; GET 2 BYTES OF
 556   04EB   85 14                 STA MC        ; MULTIPLICAND AND
 557   04ED   B9 01 00              LDA 1,Y       ; STORE IN
 558   04F0   85 15                 STA MC+1      ; MC 'REGISTER'
 559   04F2   A9 00                 LDA #$00      ; CLEAR THE
 560   04F4   85 12                 STA AC        ; AC 'REGISTER' OR
 561   04F6   85 13                 STA AC+1      ; ACCUMULATOR
 562   04F8   85 58                 STA TEMP
 563   04FA   A2 0F                 LDX #$0F      ; LOAD A COUNTER
 564                       LOOP     ; ROR MQ+1      SHIFT LOW BIT OF PRODUCT TO MQ
 565                                ; ROR MQ        ; AND LOW BIT OF MULTIPLIER TO C
```

```
CARD # LOC     CODE       CARD
 566   04FC    66                 .BYT $66,MQ+1,$66,MQ
 566   04FD    11
 566   04FE    66
 566   04FF    10
 567   0500    90 0F              BCC SHIFT    ; DON'T ADD MC IF LOW BIT IS 0
 568                              ; ROR TEMP   ; NOTE THE FACT THAT WE ARE DOING AN ADD
 569   0502    66                 .BYT $66,TEMP
 569   0503    5B
 570   0504    18                 CLC          ; ADD THE 2-BYTE
 571   0505    A5 12              LDA AC       ; MULTIPLICAND TO
 572   0507    65 14              ADC MC       ; THE 2-BYTE
 573   0509    85 12              STA AC       ; ACCUMULATOR.
 574   050B    A5 13              LDA AC+1
 575   050D    65 15              ADC MC+1
 576   050F    85 13              STA AC+1
 577                      ;  SHIFT THE 2-BYTE AC RIGHT 1 BIT, PUTTING THE CORRECT SIGN IN AC(H).
 578   0511    A5 15      SHIFT   LDA MC+1     ; SIGN = S(MC) IF WE
 579   0513    25 5B              AND TEMP     ; ALREADY PERFORMED AN ADD,
 580   0515    0A                 ASL A        ; ELSE SIGN = 0
 581                              ; ROR AC+1   ; SHIFT AC RIGHT WITH
 582                              ; ROR AC     ; CORRECT SIGN EXTENSION.
 583   0516    66                 .BYT $66,AC+1,$66,AC
 583   0517    13
 583   0518    66
 583   0519    12
 584   051A    CA                 DEX          ; COUNT ITERATIONS
 585   051B    D0 0F              BNE LOOP     ; AND REPEAT.
 586                      ;  TEST AND ADJUST FOR NEGATIVE MULTIPLIER.
 587                              ; ROR MQ+1
 588                              ; ROR MQ
 589   051D    66                 .BYT $66,MQ+1,$66,MQ
 589   051E    11
 589   051F    66
 589   0520    10
 590   0521    90 0C              BCC NOSUB    ; IF S(MQ) = 0 , NO NEED TO ADJUST.
 591   0523    A5 12              LDA AC
 592   0525    E5 14              SBC MC
 593   0527    85 12              STA AC
 594   0529    A5 13              LDA AC+1
 595   052B    E5 15              SBC MC+1
 596   052D    85 13              STA AC+1
 597                      ;  EXTEND THE SIGN OF AC IN THE FINAL SHIFT.
 598   052F    A5 13      NOSUB   LDA AC+1
 599   0531    0A                 ASL A
 600                              ; ROR AC+1
 601                              ; ROR AC
 602                              ; ROR MQ+1
 603                              ; ROR MQ
 604   0532    66                 .BYT $66,AC+1,$66,AC
 604   0533    13
 604   0534    66
 604   0535    12
 605   0536    66                 .BYT $66,MQ+1,$66,MQ
 605   0537    11
 605   0538    66
 605   0539    10
 606   053A    60                 RTS
```

```
CARD # LOC      CODE        CARD
   607                      ;
   608                      ;  CLIP - 'CLIP' THE VALUE IN SUM IF IT EXCEEDS THE CAPACITY OF
   609                      ;  THE S13.2 FORMAT OF THE XH VECTOR.
   610                      ;
   611 053B   18            CLIP   CLC
   612                      ;  ROUND SUM FROM S14.17 FORMAT TO S14.2 .
   613 053C   A5 17                LDA SUM+1
   614 053E   69 40                ADC #$40
   615 0540   85 17                STA SUM+1
   616 0542   A5 18                LDA SUM+2
   617 0544   69 00                ADC #$00
   618 0546   85 18                STA SUM+2
   619 0548   A5 19                LDA SUM+3
   620 054A   69 00                ADC #$00
   621 054C   85 19                STA SUM+3
   622 054E   70 15                BVS MAXBIG
   623                      ;  DETERMINE IF THE VALUE IN SUM (PRESENTLY S14.2) IS TOO GREAT TO BE
   624                      ;  STORED AS S13.2 .  THIS IS THE CASE IF THE SIGN BIT IS NOT THE SAME
   625                      ;  AS THE BIT PRECEEDING IT.
   626 0550   4A                   LSR A
   627 0551   45 19                EOR SUM+3
   628 0553   0A                   ASL A
   629 0554   30 0D                BMI TOOBIG      ; B7(A) EXOR B6(A) = 1 IF B7 NOT = B6 .
   630                      ;  SHIFT THE RELEVENT BYTES OS SUM LEFT 1 BIT AND STORE IN THEIR
   631                      ;  DESTINATION, POINTED TO BY X.
   632 0556   26 17                ROL SUM+1
   633 0558   A5 18                LDA SUM+2
   634 055A   2A                   ROL A
   635 055B   95 00                STA $00,X
   636 055D   A5 19                LDA SUM+3
   637 055F   2A                   ROL A
   638 0560   95 01                STA $01,X
   639 0562   60                   RTS
   640                      ;  THE VALUE IN SUM IS TOO GREAT TO REPRESENT IN S13.2 FORMAT, SO
   641                      ;  STORE THE MAXIMUM POS OR NEG VALUE IN DEST.
   642 0563   06 19         TOOBIG ASL SUM+3        ; PUT THE SIGN OF SUM IN C .
   643 0565   B0 09         MAXBIG BCS MAXNG
   644 0567   A9 FF                LDA #$FF
   645 0569   95 00                STA 0,X
   646 056B   A9 7F                LDA #$7F
   647 056D   95 01                STA 1,X
   648 056F   60                   RTS
   649 0570   A9 01         MAXNG  LDA #$01
   650 0572   95 00                STA 0,X
   651 0574   A9 80                LDA #$80
   652 0576   95 01                STA 1,X
   653 0578   60                   RTS
   654                      ;
   655                      ;  POSITN - THIS ROUTINE IMPLEMENTS A SIMPLIFIED POSITIONING SCHEME.
   656                      ;  IT IS TO BE INVOKED AT EVERY CONTROLLER ITERATION, AND COMPUTES A
   657                      ;  NEW REQUESTED VELOCITY TWR PROPORTIONAL TO THE MOTOR'S ANGULAR
   658                      ;  ERROR THERR.
   659                      ;
   660 0579   AD 01 A8      POSITN LDA IRA2
   661                      ;  START BY READING AND STORING THE MOTOR POSITION.
   662 057C   85 48                STA THM
   663 057E   AD 00 A8             LDA IRB2
```

```
                                                                  PAGE   13
CARD # LOC     CODE       CARD
  664  0581   29 7F            AND #$7F
  665  0583   85 49            STA THM+1
  666                    ;  COMPUTE THERR = THREF - THM
  667  0585   38               SEC
  668  0586   A5 52            LDA THREF
  669  0588   E5 48            SBC THM
  670  058A   85 54            STA THERR
  671  058C   A5 53            LDA THREF+1
  672  058E   E5 49            SBC THM+1
  673  0590   85 55            STA THERR+1
  674                    ;  COMPUTE REQUESTED VELOCITY TWR = CPOS * THERR
  675  0592   A2 54            LDX #THERR
  676  0594   A0 24            LDY #CPOS
  677  0596   20 E0 04         JSR MULT
  678                    ;  MOVE TWO MIDDLE BYTES OF PRODUCT TO TWR.  IF PRODUCT'S HIGH BYTE HAS
  679                    ;  SIGNIFICANCE, TOUGH - THIS IS A 'SLAP-DASH' APPROACH.
  680  0599   78               SEI             ; CAN'T BE INTERRUPTED IN THE MIDDLE OF THIS.
  681  059A   A5 12            LDA AC
  682  059C   85 57            STA TWR+1
  683  059E   A5 11            LDA MQ+1
  684  05A0   85 56            STA TWR
  685                    ;  SHIFT TWR RIGHT 1 BIT, EXTENDING THE SIGN, AND STORE IN TWRS.
  686  05A2   A5 57            LDA TWR+1
  687  05A4   0A               ASL A
  688  05A5   A5 57            LDA TWR+1       ; CARRY HAS SIGN OF TWR.
  689  05A7   6A               .BYT $6A        ; ROR A
  690  05A8   85 5A            STA TWRS+2
  691  05AA   A5 56            LDA TWR
  692  05AC   6A               .BYT $6A        ; ROR A
  693  05AD   85 59            STA TWRS+1
  694  05AF   A9 00            LDA #0
  695  05B1   6A               .BYT $6A        ; ROR A
  696  05B2   85 58            STA TWRS
  697  05B4   58               CLI
  698  05B5   60               RTS
  699                          .END

END OF MOS/TECHNOLOGY 650X ASSEMBLY VERSION 7.0

NUMBER OF ERRORS= 0,  NUMBER OF WARNINGS= 0
```

```
        SYMBOL TABLE

SYMBOL    VALUE  LINE DEFINED       CROSS-REFERENCES

ACCESS    8B86         34    104
ACR2      A80B         22   ****
ACR3      AC0B         29    136
AC        0012         45    495   497  518  521  560  561  571  573  574  576
                             583   591  593  594  596  598  604  661
ADDVB     039B        330    324
A011      0026         58   ****
A012      0028         59   ****
A013      002A         60   ****
A01       0026         57    396
A021      002E         63   ****
A022      0030         64   ****
A023      0032         65   ****
A02       002E         62    404
A031      0036         68   ****
A032      0038         69   ****
A033      003A         70   ****
A03       0036         67    412
BYE1      0355        275    231   248  257  267
CCW       0318        240   ****
CLIP      0518        611    402   410  432
CONTRL    0357        283    232
CPOS      0024         55    676
CW        0338        259    230
DDRA1     A003         16    113
DDRA2     A803         21    121
DDRB1     A002         15    115
DDRB2     A802         20    122
DELTA     0046         78    389   394
DIM       005F         95    301   399  407  415  499
DONE      04C1        525    500
DRIVEN    0200        104   ****
GETKEY    8BAF         35    152
GETVEL    026E        168    166
H1        002C         61   ****
H2        0034         66   ****
H3        003C         71   ****
IER3      AC0E         31    146
IFR3      AC0D         30    145   229  234  289
INBYTE    81D9         36    169   172  206  209
INTADH    0003         39    109
INTADL    0000         38    107
INTHDL    0300        227   ****
IRA2      A801         19    126   283  660
IRB2      A800         18    129   236  285  663
K1        001A         49   ****
K2        001C         50   ****
K3        001E         51   ****
K         001A         48    298
LOOP      04FC        564    585
LP1       045D        454    457
LP2       0470        474    482
MAXBIG    0565        643    622
MAXNEG    0401        539    531
MAXNG     0570        649    643
MAXV      030C        372    361   364
MC        0014         46    556   558  572  575  578  592  595
```

```
SYMBOL  VALUE  LINE DEFINED      CROSS-REFERENCES

MOVING   0397          327   308   310
MQ       0010           44   491   493   512   515   552   554   566   589   605   683
MSKCA1   0002           10  ****
MSKT1    0040           11   288
MULT     04E0          551   490   508   677
MVOLTS   00A0            6   363
NEGV     026C          167   156
NEXT     0494          499   523
NONEG    03CC          361   351
NOPOS    044F          439   432
NOSUB    052F          598   590
NOVFLW   03BB          350   321   337
NSTATE   0003            5   300   398   406   414
OK1      0442          431   428
ORA1     A001           14   117   379
ORB1     A000           13   118   380
OUTV     03E0          376   368
OVFLOW   04C2          531   429   524
OVFLW    03DA          370   311   338   347
PCR2     A00C           23  ****
PCR3     AC0C           29   133
PFLAG    005E           64   202   215   247   266   436
POSITN   0570          660   458
POSV     0260          165   154
SAVW     028C          188   175
SCALE    046A          471   366
SC       0061           97   478
SETANG   02A6          206   158
SHIFT1   0477          479   476
SHIFT    0511          578   567
SMQ      0060           96   471   475
SOUT     0023           94   350   371   376   381
SUBVB    03AC          340   325   328
SUM      0016           47   313   315   317   319   323   331   333   334   336   341
                             343   344   346   354   355   357   358   362   419   421
                             422   424   425   427   492   494   496   498   511   513
                             514   516   517   519   520   522   533   534   535   537
                             540   542   543   545   613   615   616   618   619   621
                             627   632   633   636   642
TEMP2    005C           92   489   501   507
TEMP3    005D           93   168   174
TEMP     005B           91   562   569   579
THERR    0054           88   670   673   675
THP      004A           80   127   130   240   242   243   245   259   261   262   264
                             388   393   444   446   447   449
THM      0048           79   284   286   327   387   390   392   662   665   669   672
THMAXH   0038            7  ****
THMAXL   003C            8  ****
THREF    0052           87   212   213   250   252   253   255   269   271   272   274
                             668   671
TOOBIG   0563          642   629
TPER     003E           72   138   140
TSTSGN   0391          323   314   316   318   320
TWRS     0058           90   194   197   200   420   423   426   690   693   696
TWR      0056           89   189   190   445   448   682   684   686   688   691
T1CH3    AC05           25   142
T1LH3    AC07           27   141
T1LL3    AC06           26   139
UIRQVC   A678           33   108   110
```

```
VBIAS    0020         52   332  335  342  345
VCMULT   047F        489   302  400  408  416
VOUT     0022         53   373  378  484
WAITLP   0257        152   159  170  173  204  207  210  217
XHKP1    004C         82   454
XH1      0040         75  ****
XH1KP1   004C         83   401


SYMBOL  VALUE  LINE DEFINED        CROSS-REFERENCES

XH2      0042         76   307  309
XH2KP1   004E         84   409
XH3      0044         77  ****
XH3KP1   0050         85   431
XH       0040         74   299  397  405  413  455
ZREF     030D        233   230
```

REFERENCES

1. Greenleaf, J. F., Johnson, S. A., Lee, S. L., Herman, G. T., and Wood, E. H. (1974) Algebraic reconstruction of spatial distributions of acoustic absorption within tissue from their two-dimensional acoustic projections. Acoustical Holography, (Philip S. Green, editor) Plenum Press, New York, 5, 591-603.

2. Greenleaf, J. F., Johnson, S. A., Samayoa, W. F., and Duck, F. A. (1975) Algebraic reconstruction of spatial distributions of acoustic velocities in tissue from their time-of-flight profiles. Acoustical Holography (Newell Booth, editor), Plenum Press, New York, 6, 71-90.

3. Greenleaf, J. F., and Johnson, S. A. (1975) Algebraic reconstruction of spatial distribution of acoustic speed and attenuation in tissues from time-of-flight and amplitude profiles. Proceedings, Seminar on Ultrasonic Tissue Characterization, NBS Special Publication 453, Gaithersburg, Maryland, May 28-30, pp. 109-110.

4. Greenleaf, J. F., Johnson, S. A. Samayoa, W. F., and Hansen, C. R. (1976) Refractive index by reconstruction: use to improve compound B-scan resolution. Proceedings, Seventh International Symposium on Acoustical Holography and Imaging, (L. W. Kessler, editor), Plenum Press, New York, 7, 263-273.

5. MOS Technology, Inc., MCS6500 Microcomputer Family Programming Manual, MOS Technology, Inc., 1975.

6. Synertek Systems Corporation, SYM Reference Manual, Synertek Systems Corporation, Santa Clara, California, 1975.

7. Kuo, B. C., Automatic Control Systems, 3rd Edition, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1975.

8. Bingulac, S., "Linsys Conversational Software for Analysis and Design of Linear Systems", Report T-17, Coordinated Science Laboratory, University of Illinois, Urbana, Illinois, June, 1975.

9. Electro-Craft Corporation, DC Motors, Speed Controls, Servo Systems, 4th Edition, Electro-Craft Corporation, Hopkins, Minnesota, July, 1978.

10. International Business Machines Corporation, Continuous Systems Modelling Program III (CSMP III) Program Reference Manual, Program Number 5734-X59, Publication No. SH19-7001-3, December, 1975.

11. Electro-Craft Corporation, Product Catalog, Electro-Craft Corporation, Hopkins, Minnesota, p. 76.