WORKING PAPER 173

STEPPING MOTOR CONTROL SYSTEM

by

Noble G. Larson

Massachusetts Institute of Technology
Artificial Intelligence Laboratory

February, 1979

Abstract

This paper describes a hardware system designed to facilitate position and velocity control of a group of eight stepping motors using a PDP-11. The system includes motor driver cards and other interface cards in addition to a special digital control module. The motors can be driven at speeds up to 3000 rpm. Position feedback is provided by shaft encoders, but tachometers are not used.

A.I. Laboratory Working Papers are produced for internal circulation, and may contain information that is, for example, too preliminary or too detailed for formal publication. Although some will be given a limited external distribution, it is not intended that they should be considered papers to which reference can be made in the literature.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Office of Naval Research under Office of Naval Research contract N00014-77-C-0389

TABLE OF CONTENTS

1.0 Introduction

The control system described in this paper has been built specifically for driving a manipulator arm designed at the A.I. Lab by John Purbrick. This requires position control to within 1 mil and stable velocity control for speeds up to 3000 RPM. The manipulator has three rectilinear joints and three rotational joints, two of the latter driving a differential wrist. Two additional motors are provided to control a vise, which is part of the manipulator. Since this system is quite general in structure, it should lend itself to a fairly wide variety of stepping motor applications.

Several features of stepping motors make them attractive for this application. First, the discrete and repeatable nature of their motion makes precise positioning easy. Second, shaft encoders are available with outputs commensurate with the motor steps, facilitating the entire design. Third, in addition to being capable of high speeds, stepping motors can move very slowly without "stick-and-slip" problems. Fourth, the holding torque of the motors eliminates the need for brakes. Finally, open loop stepping provides a means of achieving smooth velocities without heavy dependence on velocity feedback, in particular, tachometers.

1.1 Stepping Motors

The control system is designed around Superior Electric M Series stepping motors. The latter are 200 step/revolution 4 phase DC permanent magnet motors, with bifilar windings. In this type of winding, the four coils are grouped into two pairs, the coils within a pair being tightly linked, inductively. The motors are used here as shown in fig. 1. One coil in each pair is always activated, resulting in four distinct states as shown below (coils are identified by lead color, referring to fig. 1.):

Coil State	G	G/W	R	R/W
0	0	1	0	1
1	0	1	1	0
2	1	0	1	0
3	1	0	0	1

The arithmetic code on the left is used throughout the control system. An incrementing sequence of coil states(0,1,2,3,0,...)represents forward motion of the motor shaft. This in turn is defined as counter clockwise rotation, viewed from the motor label.

Associated with the four coil states are four static torque curves, which represent torque as a function of position for a given coil state, assuming a fixed rotor. Typical curves are shown in fig. 2(the position representation used in figs. 2 and 3 is explained in the next section). Here as elsewhere in this paper, it is assumed that these curves are sinusoidal.

An important factor in this system is the back EMF generated by the motor itself. This quantity, a voltage, is also roughly a sine wave, with amplitude proportional to the(signed)angular velocity. Measured with respect to the center taps(see fig. 1), the voltages at the four end leads, for forward motion, are as shown in fig. 3.

1.2 Feedback

Position feedback is provided by pulse generating shaft encoders. These devices are mounted on each motor, and output two TTL square waves in quadrature as the shaft rotates. Each square wave has 200 cycles per shaft revolution. By virtue of the quadrature relationship, the two phases partition a complete shaft rotation into 800 segments. These occur as a repetitive sequence of the following encoder states:

Encoder State	B Phase	A Phase
0	0	0
1	0	1
2	1	1
3	1	0

Both the arithmetic code and the gray code are used in the system. A single XOR gate suffices to convert between codes. Unless otherwise stated, it is the binary code that is referred to in this paper. An incrementing sequence of encoder states(0,1,2,3,0,...--arithmetic code)represents forward motion.

It should be noted that the four encoder states--0,1,2,3--constitute exactly one step. The encoders are aligned on the motor shafts so that the 1->2 encoder transition occurs at the rest or *detent* position of the motor. This convention gives rise to a natural position representation, with the encoder state serving as the rightmost two bits, and an arbitrary number of bits to the left(14 in the case of this system), corresponding to the step. The step portion of the position is programable, but the encoder state or quarter step portion is strictly dictated by the encoder. By convention we will assume that the detent position of coil state 0 corresponds to the 1->2 encoder transition of step 0. This must be effected by initialization.

The shaft encoders also output an index pulse, or mark, once per revolution. This is used for calibration of the manipulator during initialization.

Except for limit switches no other feedback is used in the stepping motor control system. In particular, there are currently no plans for using tachometers.

1.3 Phase Angle and Lead Angle

In discussing stepping motors it very useful to have a simple terminology to describe the relationship between the coil state and the motor shaft. The concepts of *phase angle* and *lead angle* are introduced for this purpose. The phase angle is defined as the amount by which the coils are leading the shaft,or, more specifically,as the rightmost 4 bits of the quantity(PA)obtained by subtracting position(POS)from four times the coil state(CS):

PA = 4*CS - POS [rightmost 4 bits]

The multiplicative factor is necessary since a motor step consists of four encoder steps. This representation allows for 16 phase angles: 0, 1 through 7, -1 through -7, 8, and -8, the last two being equal and ambiguous. Since the unit is a quarter step, we can represent phase angles up to +2 or -2 steps. Phase angles with magnitudes greater than 2 motor steps are equivalent to smaller phase angles, namely, those represented by the rightmost 4 bits. Phase angles are arbitrary in so much as the step portion of position is arbitrary, prior to initialization. In other words, even though it may seem reasonable to choose the detent position of step 0 as corresponding to coil state 0 at initialization, resulting in an initial phase angle of -1 or -2, it is an arbitrary choice. The phase angle, a periodic function of position and coil state, is always defined.

The lead angle is one of the two phase angles associated with the instant that the motor coils are switched from one state to the next. In the case of forward motion, the phase angle <u>after</u> the coil transition is indicated. In the case of reverse motion, it is the phase angle <u>before</u> the transition. Negative lead angles are sometimes referred to as *lag angles*. Lead angles are discussed further in the next section.

1.4 Low Level Control

An important method for controlling stepping motors can now be introduced. This method will be called lead angle control since it consists of imposing a fixed lead angle(LA)upon the motor. This amounts to solving the phase angle equation for coil state:

$$CS = (POS + LA)/4 \qquad \text{[rightmost 2 bits]}$$

Static torque curves are also meaningful in reference to lead angles. In fig. 4 the static torque curve for LA = 5 is superimposed on those of the four coil states. Associated with each lead angle is an *average static torque*, proportional to the integral of a single period. These average static torques are sinusoidal as functions of lead angle. A circular representation, suggested by Matt Mason, is shown in fig. 5. The angle θ is equal to $LA*\pi/8$; the y-axis represents torque. The static nature of this quantity must be emphasized. The *average dynamic torque*, is a much more difficult quantity to derive, other than qualitatively. The main effects to consider are the delay time in turning one coil off and another on, and motor back EMF, which subtracts from the driving voltage. At even moderate speeds these factors significantly affect motor torque. The net result is contraction and rotation(clockwise for forward motion and counter clockwise for reverse motion)of the torque diagram. Both effects increase with motor speed. An important consequence of this is that lead angles with negative static torques can have positive dynamic torques at high speeds, and vice versa. Control schemes must take this effect into account.

Associated with the average dynamic torque is a *terminal velocity*. This is the velocity at which the average torque exactly compensates for the losses due to friction and other effects. Assuming that the static torque associated with a lead angle is sufficient to start the rotor in motion in the right direction, the velocity will slew up to this value and stay there. Terminal velocities for one of the motors used here are plotted in fig. 6. Unfortunately, the terminal velocity corresponding to a particular lead angle has a tendency to drift in time. As a result, lead angles, while providing a convenient way of slewing between velocities, do not constitute, in themselves, a means for stable velocity control. This does not rule out higher level control schemes, such as *lead angle modulation*, which Matt Mason has employed with considerable success. Fig. 7 shows phase angle plotted against time for a motor running at a terminal velocity, under lead angle control.

Probably the most common method of controlling stepping motors is the open loop method. This consists of stepping the coils at uniform intervals, determined by the desired step rate. The result, in general, is rotation of the motor shaft in the direction of the coils. Unfortunately, this is not necessarily the case. First, the rotor may not be able to follow the coils at all, the result being stalling. Second, whereas the coils may be advancing forward at rate X, the rotor may actually move backward at rate 3X, or even forward at rate 5X. This is due to the existence of harmonics, as illustrated in fig. 9. They do not pose much of a problem, since the higher order harmonics are increasingly unstable.

It is instructive to plot phase angle vs. time for a motor running under open loop control. Some examples are shown in fig. 8. It can be seen from these that open loop control also results indirectly in a lead angle. Presumably these lead angles are precisely those required to achieve the open loop stepping rate as a terminal velocity, under lead angle control. At resonant frequencies the phase angle goes through more complicated gyrations.

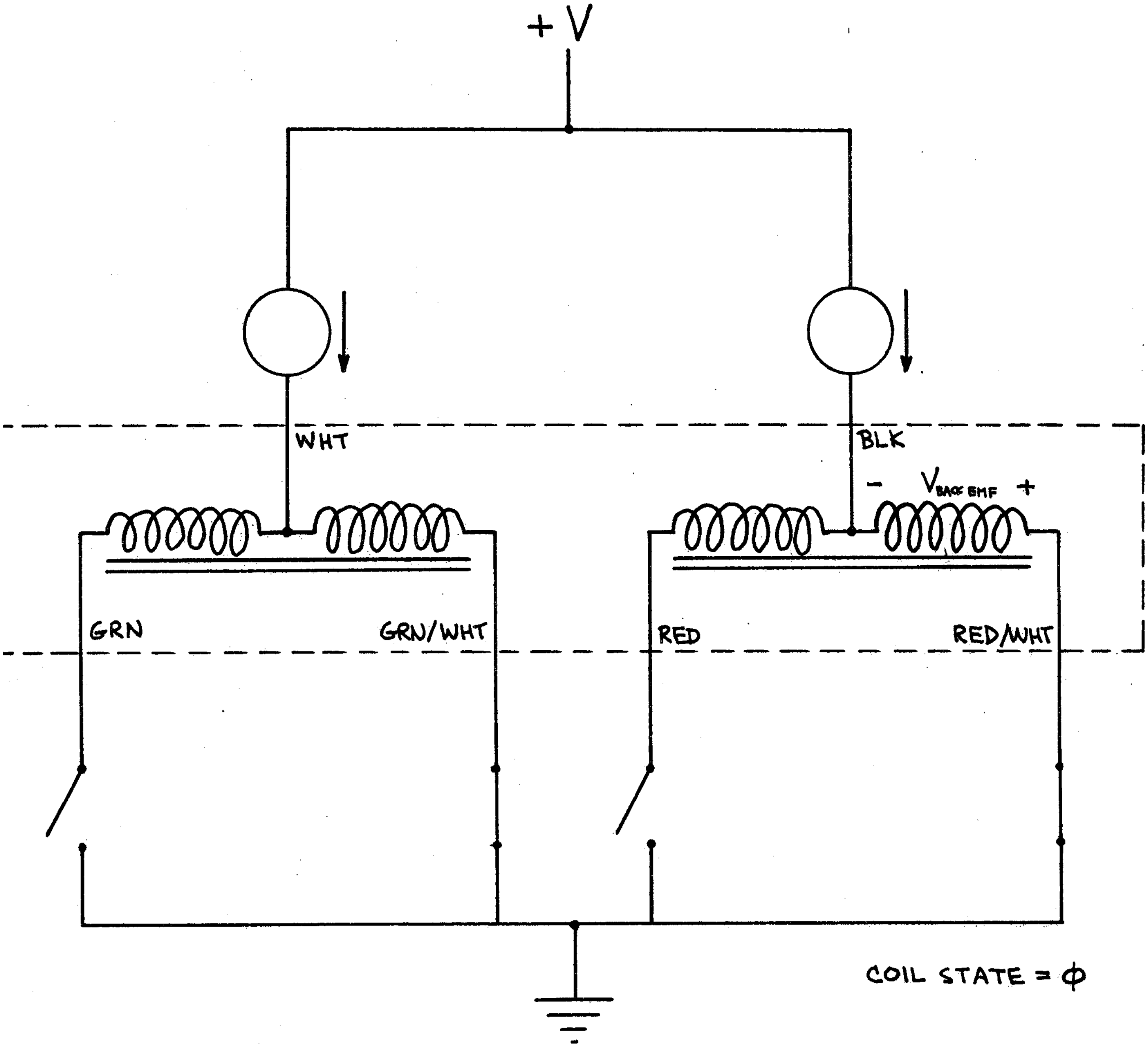


Figure 1. 4 Phase Stepping Motor in Drive Circuit

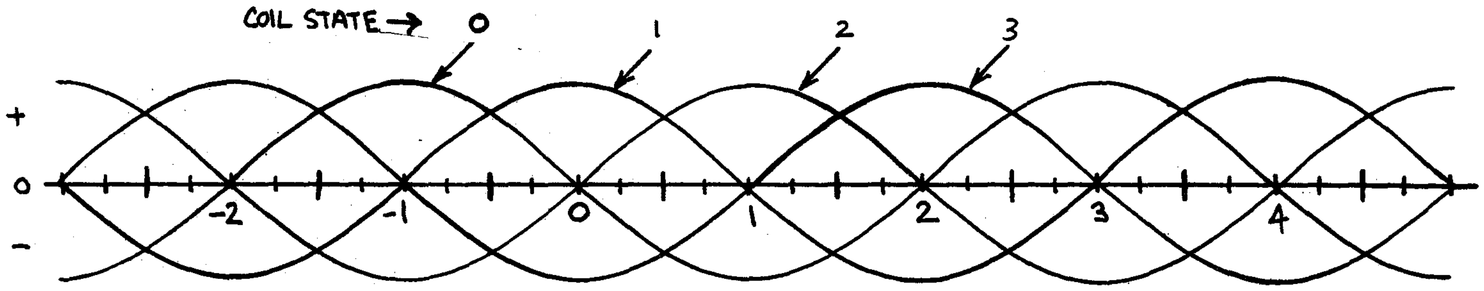
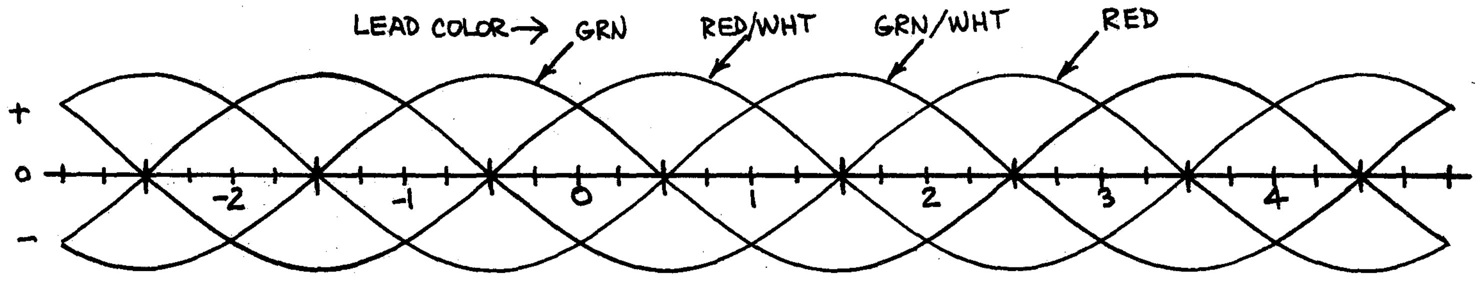


Figure 2. Static Torque Curves



(VOLTAGES SHOWN FOR FORWARD MOTION
INVERT CURVES FOR REVERSE MOTION)

Figure 3. Back EMF's

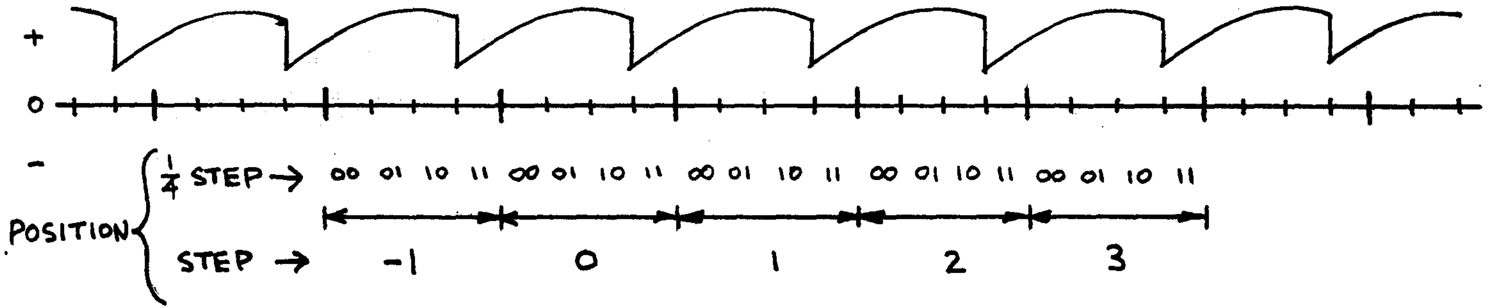


Figure 4. Static Torque Curve for LA = 5

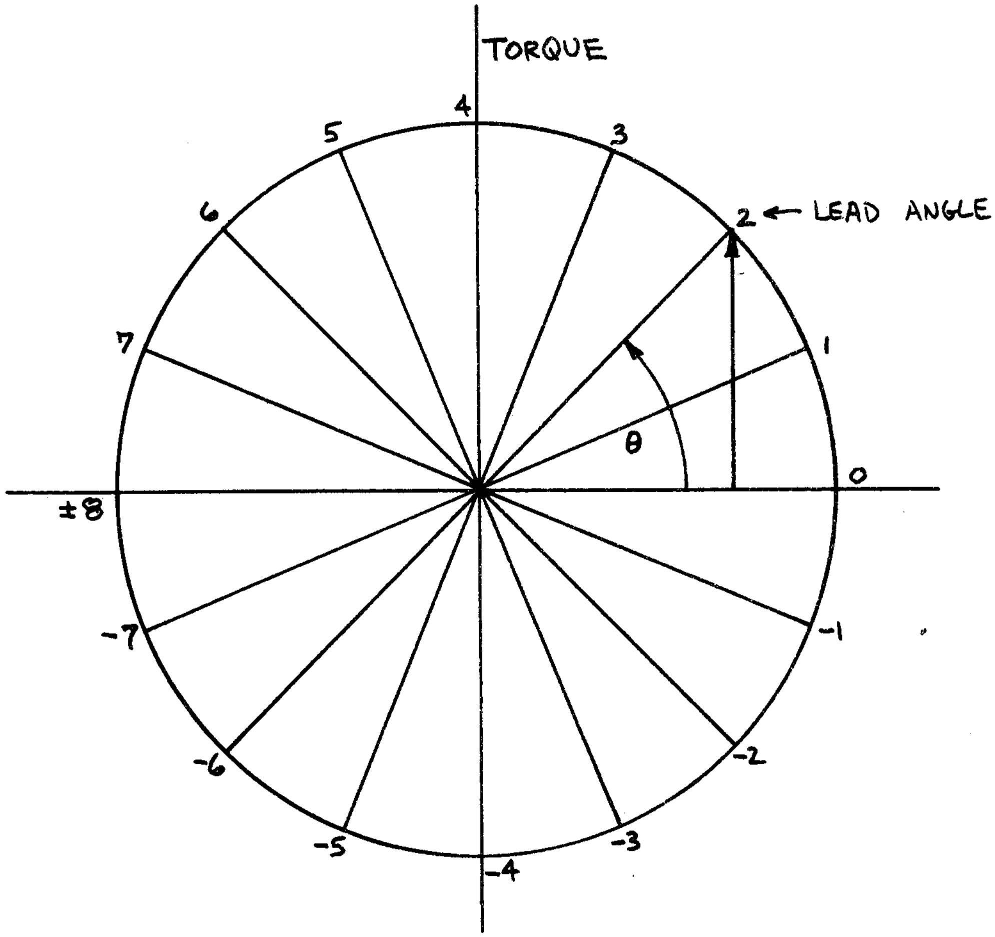


Figure 5. Circular Representation of Average Static Torque

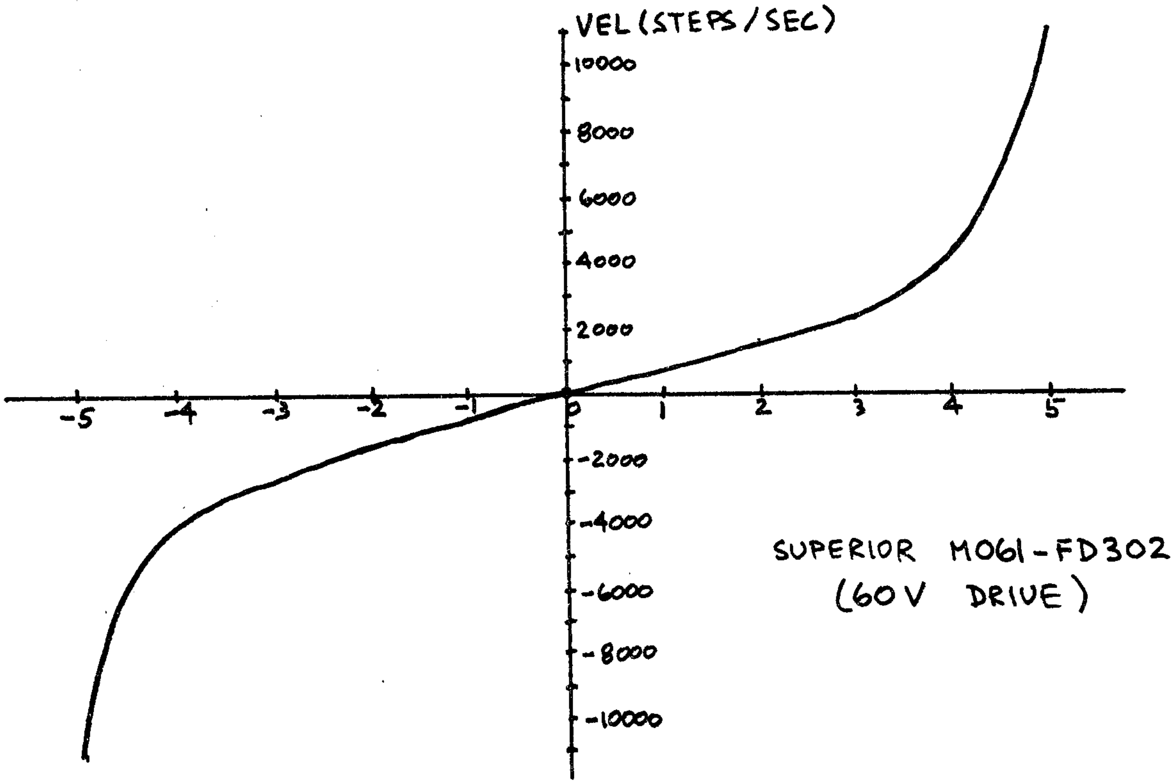


Figure 6. Terminal Velocity vs. Lead Angle

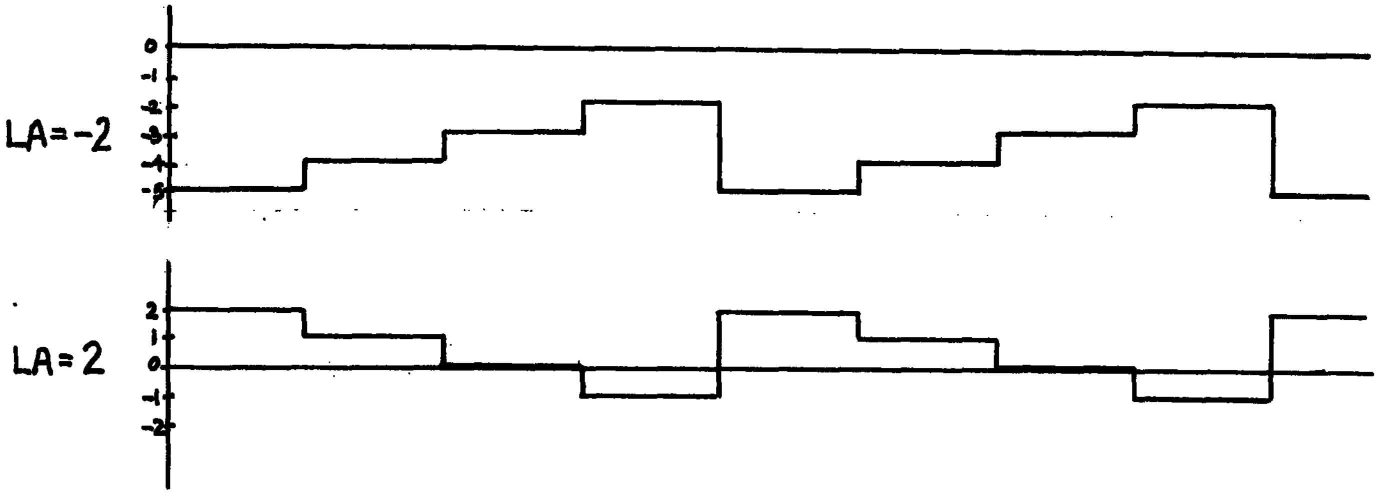


Figure 7. Phase Angle vs. Time - Lead Angle Control

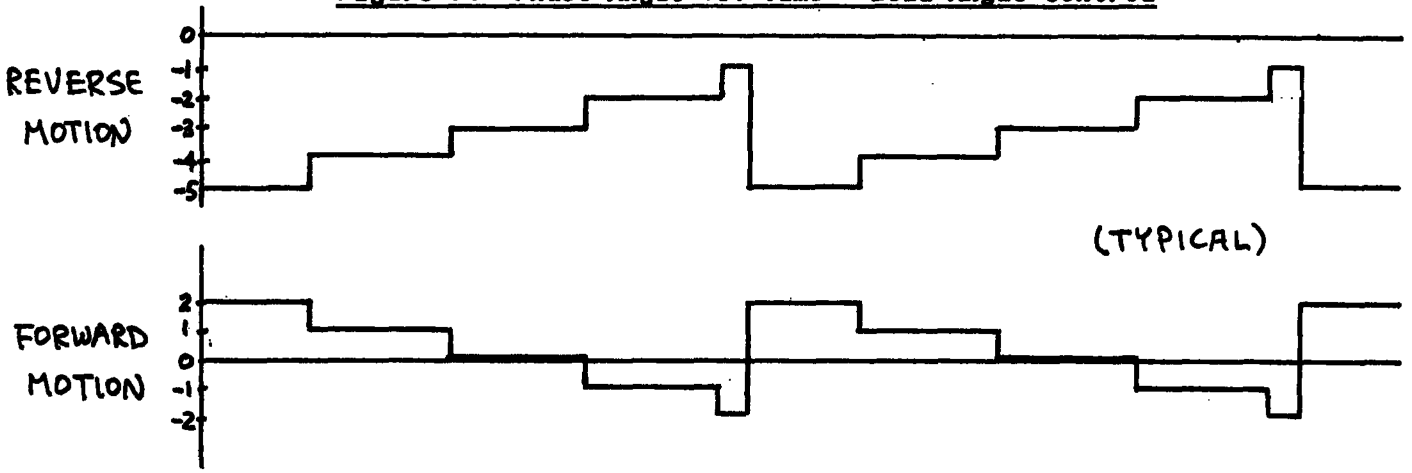


Figure 8. Phase Angle vs. Time - Open Loop Control

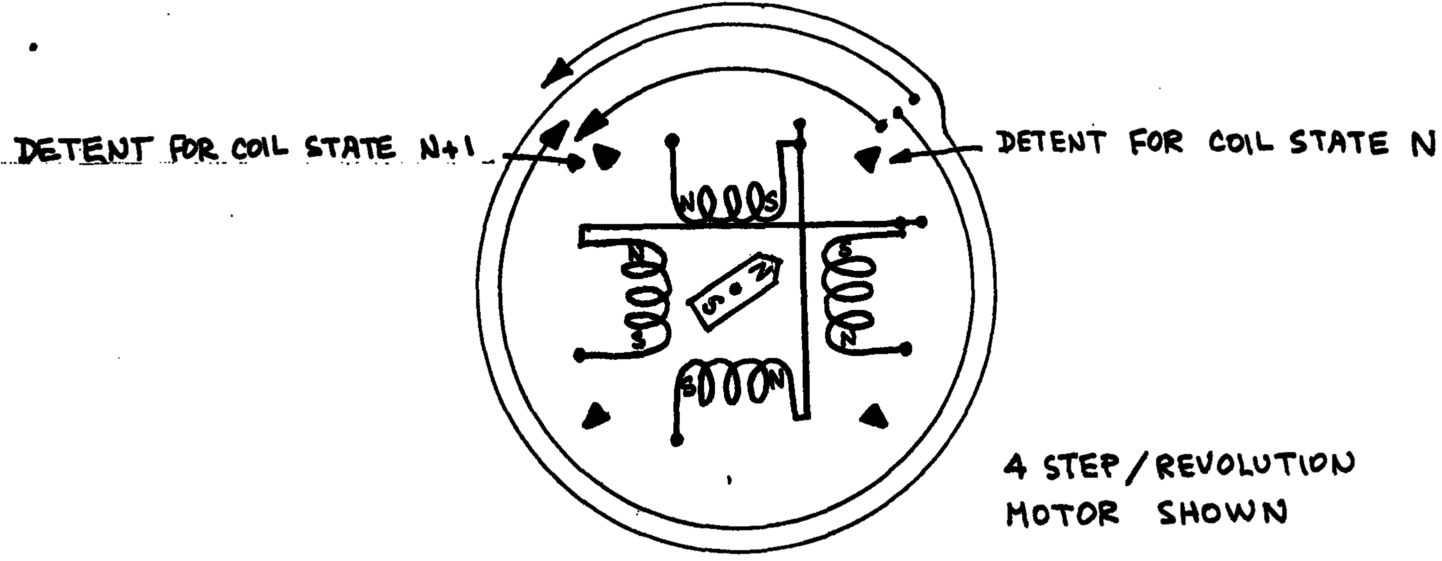


Figure 9. Harmonics

1.5 Motor Drivers

The foremost design consideration in this, as in any such system, is the motor driver circuit. A simplified diagram of the type used here is shown in fig. 1.

In order to achieve even moderate speeds, it is necessary to rapidly switch coil currents. Rapid current switching implies putting a fairly high voltage(40V-100V in this case)across the coils in the process of turning them on. It also implies large voltage spikes in the same place when shutting them off. These spikes may have to be limited in order to protect the driving circuitry. The duration of the spike depends on the current, the coil inductance, and the spike voltage. If the driving circuit is designed to block current coupling between the bifilar wound coil pairs, then the coil current will have to be shut off entirely by the spike. The transistors used to switch the coils must be able to withstand these conditions without secondary breakdown occuring. It is significant that in this type of circuit the coil being shut off will induce a voltage in the coil being turned on. This voltage tends to delay the buildup of current in the latter until the former has been shut off. On the other hand, if the driving circuit is designed to allow current coupling, then most of the current in the coil being shut off will be transferred instantly into the other coil. It will be flowing backwards, i.e. toward the center tap, but since the paired coils are 180 degrees out of phase, the resulting torque is the same. This reverse current must, of course, be stopped before current can begin to flow in the proper direction in the coil being turned on. The remaining, uncoupled current must be dissipated through the spike. This scheme has the advantage of greatly reducing power dissipation, in particular, during the spike, when secondary breakdown of the transistor can occur.

Back EMF's have a significant effect on coil switching. In the case of the Superior Electric M061-FD302, the peak-to-peak EMF amplitude increases by roughly 20 volts for every 1000 steps/sec of velocity. Careful examination of figs. 3 and 4 shows that this voltage tends to oppose both the switching on and switching off of coil currents.

The voltages and currents involved can cause a power dissipation problem, calling either for large heat sinks, or power-efficient current control.

1.6 Limit Switches

A very important practical consideration in this design is that of *limit switches*. Since the speeds desired and the gearing employed raise the possibility of severe damage to the mechanical system in the event of a software failure, protection is required. In particular it is essential to prevent the control hardware from running a joint with finite travel into one of the stops. The approach here has been to run signals from mechanical or optical limit switches directly to the motor driver circuits, and to cause this signal to inhibit further motion toward the stop. It is also desirable to be able to back out of a limit switch condition without manual intervention.

The existence of the differential wrist in this manipulator arm, poses special problems in the limit switch area. This wrist has two joints--rotation and elevation--driven by two motor channels. Both channels are required to move either of these joints. Rotation has no travel limits, but elevation requires limit detection at two extremes. For each of these limits, only one direction of motion in each channel can result in further travel into the limit. It is sufficient to cause each elevation limit to inhibit motion in one direction in both motor channels.

2.0 Control System Overview

Fig. 10 shows a fairly general picture of a motor control algorithm. It is organized as a nested heirarchy of control processes. The inner process performs low level control, described in section 1.4, in addition to position tracking. It is nested within a velocity control process, which is in turn nested in a position controller. The outer process performs the higher level tasks such as trajectory control.

2.1 The Need for Special Hardware

Although it is possible to control the motors with a PDP-11, by simply making the encoder state and coil state directly accessible to the processor, the consequent limitations are too severe for our application. These limitations originate, at least partly, in three problems, associated with the inner control process.

At maximum motor speed, 10000 steps/sec, encoder transitions can occur at 18 μs intervals (this figure takes into account certain departures from the ideal in encoder behavior). In order to keep track of positions it is necessary to sense every such transition. Clearly, monitoring even one such motor is barely feasible for an unaided minicomputer. In short, keeping track of joint position is a very time consuming job for the processor.

The implementation of lead angles is another demanding task. It requires computing coil states from joint position at frequency comparable to the encoder step rate, i.e. four times the motor step rate.

Open loop stepping also presents a problem. Stepping rates are most conveniently generated by a clock driven interrupt routine. In order to generate a reasonable spectrum of step rates, this routine would have to be run at an unfeasibly high frequency(typically, 100 KHz).

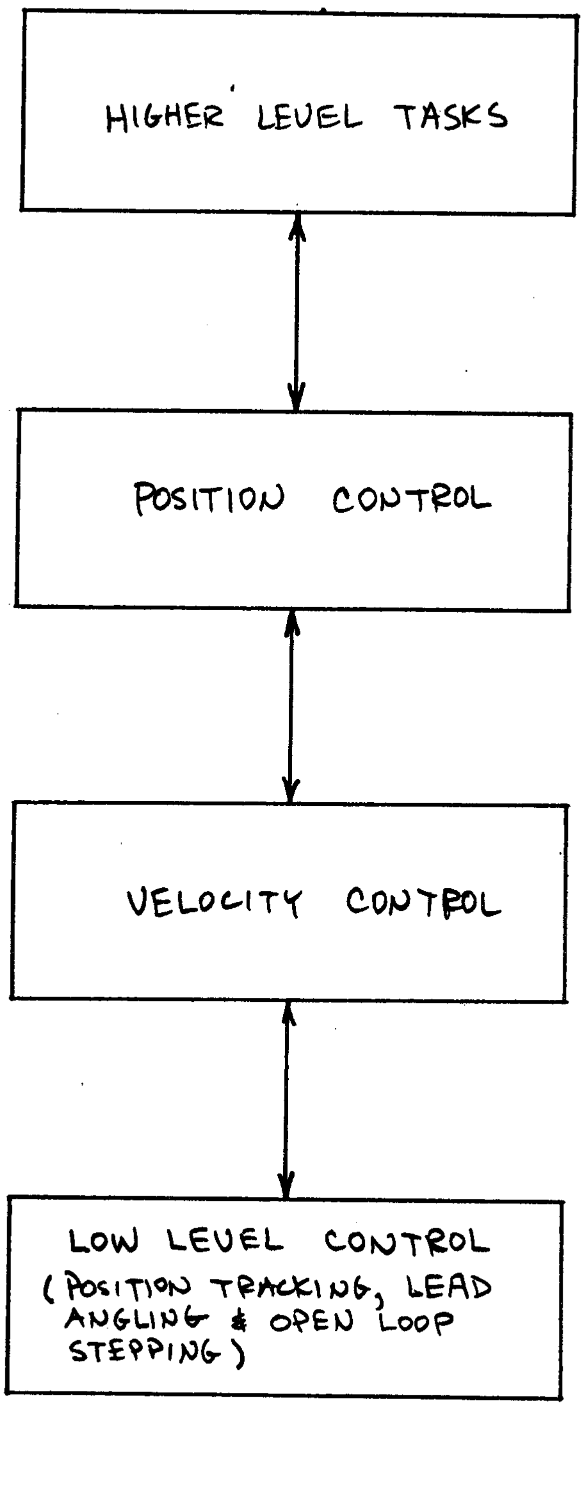


Figure 10. Motor Control Algorithm

2.2 Hardware Requirements

The most difficult problem which has been considered in this design is that of velocity control. It is complicated somewhat by the absence of tachometers, requiring that velocity be computed by differentiation. This computation is affected by encoder resolution, joint inertia, and sampling rate. In the course of the design, a number of velocity control schemes have been studied. The common denominator of all of them is the need for a hardware solution to the three problems described in the last section. In essence the goal is to relieve the PDP-11 of the time consuming drudgery, allowing it to maintain tight control over the motors through relatively infrequent commands to hardware. While doing this, it is important to retain maximum flexibility as to the nature of the higher lever control processes. In addition to this, the hardware must support: joint calibration, detection of limit conditions, detection of encoder tracking errors(these will arise if the frequency of the encoder transitions, i.e. motor speed, overruns the position tracker), and rotary joints with periodic position values(i.e., unlimited travel).

2.3 Controller Functionality

The most basic function of the controller is position tracking. It maintains 16 bits of position information, in the format described in section 1.2., for each of the eight motors. The controller can accurately track motors at speeds up to 15000 steps/sec.(4500 RPM). Motor positions are always directly accessible by the processor via the UNIBUS(Trademark of Digital Equipment Corporation).

The controller provides software with four different control modes, for each motor channel. *Direct* mode allows software to force load a particular coil state in any motor, or to step a motor forward or backward. Loading is necessary for initialization and calibration. Direct stepping is a programming convenience. *Lead Angle* mode activates lead angle control for a particular motor. Separate modes are defined for *Open Loop Forward* and *Open Loop Reverse*. In either mode, the step rate is determined by a programmable 16-bit *set count*. Steps are generated by repeatedly decrementing this quantity in a counter. Both the set count and the counter are accessible by the processor, via the UNIBUS^{TM}.

The coil state, limit condition, tracking error status, and index mark for each motor, is accessible by the processor at all times.

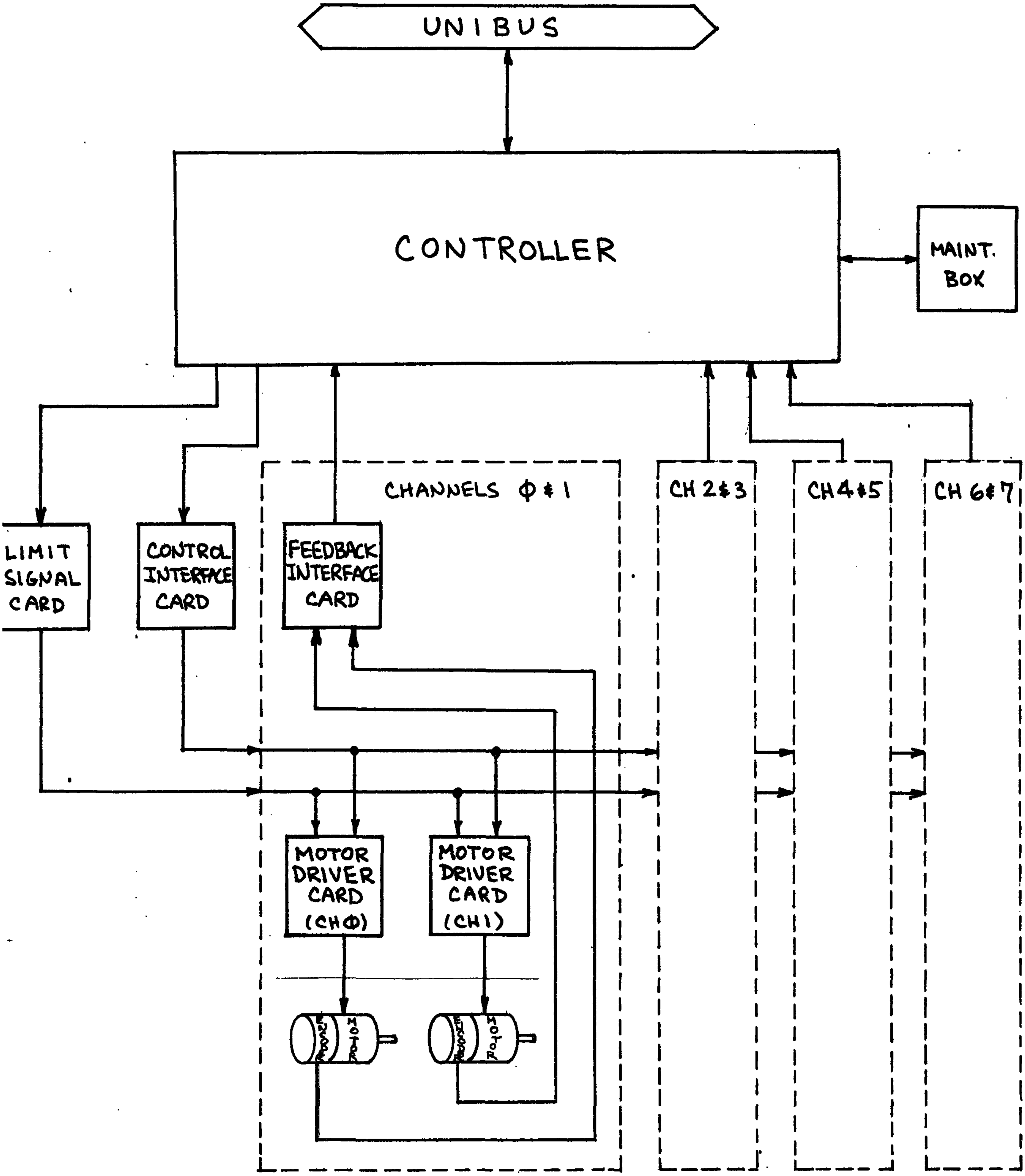


Figure 11. Hardware Block Diagram

3.0 Hardware Description

An overall block diagram of the hardware is shown in fig. 11.

The controller contains the logic for UNIBUS^{TM} interfacing, position tracking, lead angle control, and open loop step generation. It also collects information on limit and error conditions, coil states, and index pulses. All of this is can be accessed by the processor via the UNIBUS^{TM}. The limit signals also get passed on to the driver cards for joint protection.

Feedback signals(shaft encoder and limit switch lines)are collected at the manipulator arm in a junction box. This box contains 4 occurances of the same card including +5 volt supplies for powering the cards and the encoders. Each of these feedback interface cards services two motors, providing support circuitry for limit switches in addition to line drivers for shaft encoder signals.

The motor driver cards are plugged into a separate DEC backpanel. Each card contains the driving circuitry necessary for a single motor, in addition to a small amount of coil state logic.

The control interface card allows the controller to operate on the coil state logic of any one driver card. It is plugged into the motor driver backpanel.

A limit signal cable card is necessary for bringing the limit signals into the motor driver card cage. It is also plugged into the motor driver backpanel.

A maintenance box can be connected to the controller for trouble-shooting. It is not normally plugged in.

Except for the controller itself, all cards in the system are printed circuit cards. Both the artwork layout and fabrication of these was done by Fred Drenckhahn.

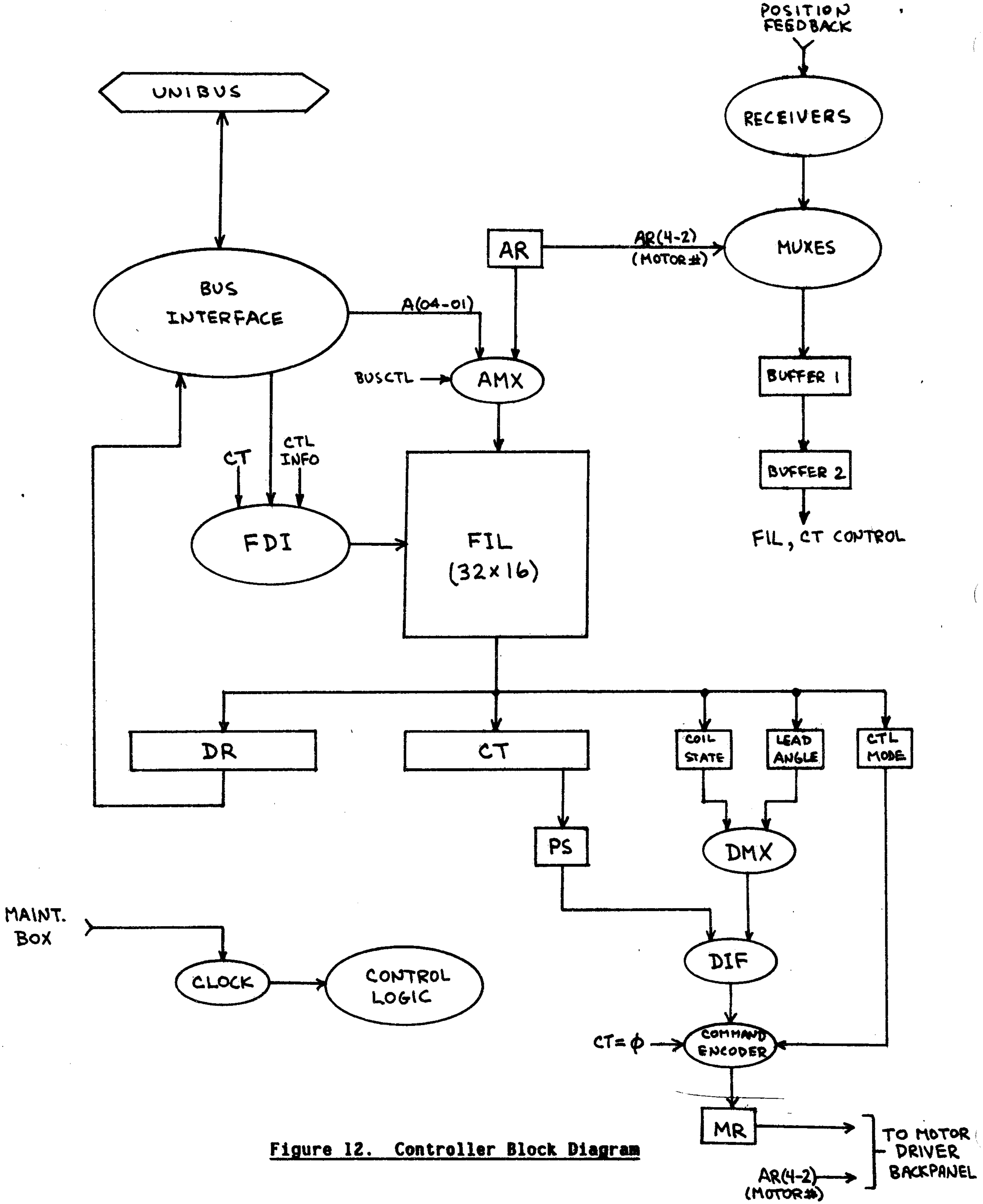


Figure 12. Controller Block Diagram

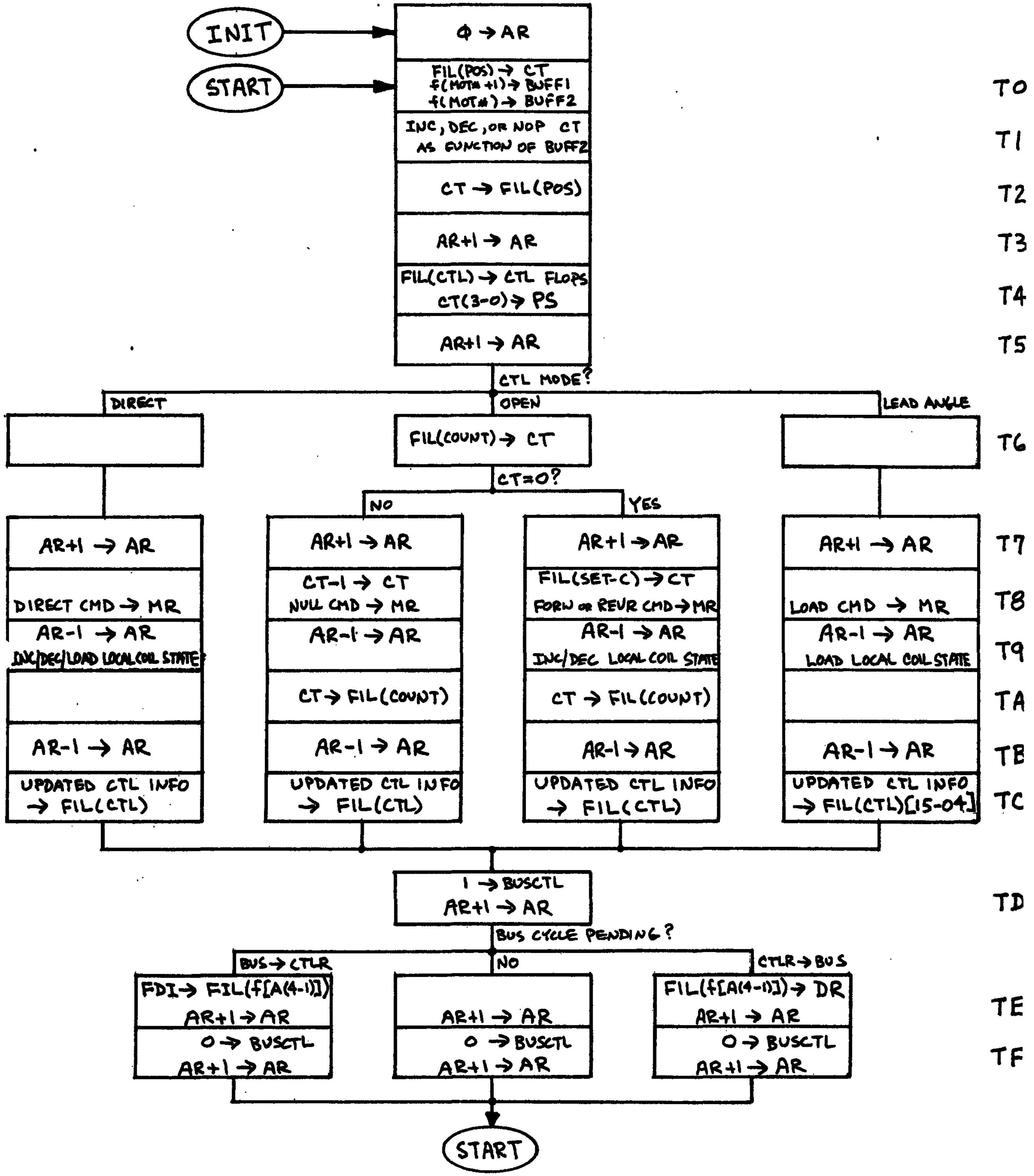


Figure 13. Controller Cycle Flow

3.1 Controller

The controller is implemented on a single Augat LG-576 wire wrap board. It is a quad high board and plugs directly into a standard DEC small peripheral slot. This board has 110 dip locations, of which 100 are used. All external connections are made via the seven 26-pin cable connectors provided on the LG-576.
These are allocated as follows:

4 - Cabling to the four feedback interface cards
1 - Cabling of limit lines to motor driver card cage
1 - Cabling of motor commands to the control interface card
1 - Internal clock control(goes to maintenance box)

Connection to the UNIBUS^{TM} is made in the standard fashion via the four edge connectors, which also provides power(+5 volts, only).

A block diagram of the controller is given in fig. 12.

All signal names used below refer to the logic drawings.

The unit is a synchronous device, driven by a delay line oscillator clock, with an 82 ns. cycle time. The clock can be stopped, stepped, and restarted by the maintenance box logic. During normal operation, the clock always free-runs at a constant frequency. The main timing reference is the signal, TIM000+0. From this are derived the main clock, CLOCK+0, and several other offset timing signals: CLOCK-0, ENCLK+0, and ENCLK-0.

The initialization logic consists mainly of two one-shots, MCLR-0 and ICLR-0. These can be triggered by the UNIBUS^{TM}(BINIT-0)or by the maintenance box logic(MINIT-0). MCLR-0, a 200 ns negative going pulse, is used for resetting the oscillator. ICLR-0, a 240 ns negative going pulse, is used for resetting certain synchronous elements. The length of MCLR-0 is fixed. As it is necessary that ICLR-0 be between 30 and 50 ns longer in duration, a trimpot(Z007)is provided to allow for adjustment.

Control logic timing is derived from the state counter, SC(3-0). This element is cleared at initialization, and increments repetitively through 16 states, 0-F. For each state, both a level(e.g. T0-0)and a pulse(e.g. P0-0)are created. These are generated from SC and ENCLK-0, by four

74S138's. These signals, together with machine state are used to create the various control signals such as register strobes, counter clocks, counter enables, and write pulses. The control logic is totally hardwired and repetitive in nature.

The controller is organized around a 16x32 register file, FIL(15-00), constructed from eight 82S21's. Four 16-bit words of storage are provided for each of the 8 motor channels. All relevant information on the motors is kept in this file, in the layout shown in fig. 20. It serves as the buffer between software and the control system hardware. The file is addressed by a 2-input mux, AMX(4-0). This permits access either from the UNIBUS^{TM} interface, A(5-1), or from the internal address register, AR(4-0). File data is written from a 4-input mux, FDI(15-00). This allows data to be written from the UNIBUS^{TM}, BD(15-11); the main counter, CT(15-00); or from various control elements.

The 16-bit counter, CT(15-00), is provided for performing the arithmetic needed for position tracking and open loop stepping. The associated quantities in the file can be loaded into CT, operated on, and written back.

The coil state logic generates commands for the motor driver cards, in addition to maintaining a local copy of motor coil states, accessible to software. The commands can either be coil state load(used in lead angle control and direct load)or coil step(used in open loop control). The coil state logic has storage elements for control mode, direct load state, direct load command, coil state, tracking error, lead angle, and bits 3-0 of position. As a hardware convenience, lead angle and phase angle are stored in <u>negated</u> form. In lead angle mode a 4-bit subtractor, DIF(3-0), is used to generate coil state from position and lead angle. For all other modes, the same subtractor is used to compute the phase angle from position and current coil state. A 4-bit mux, DMX(3-0), determines which operation is occuring. In essence, the following versions of the phase angle equation are being used:

$$-PA = POS - 4*CS$$

and $$CS = (POS - (-LA))/4$$

The command to the motor drivers is encoded into 7 signals:

MR0+0 - MR1+0	- Command type
MR2+0 - MR3+0	- Coil load state
AR2+0 - AR4+0	- Motor channel #

where the command type codes are:

00	- Null
01	- Load
10	- Step forward
11	- Step backward

These, together with a strobe(STRB+0)are transmitted to the motor driver card cage by line drivers(26LS31).

The feedback interface contains 24 line receivers(26LS32)for the encoder signals(A phase, B phase, and index pulse), five 8-to-1 muxes, and two sets of flip-flops for synchronizing these external signals with the controller clock. The five(multiplexed)external signals are strobed into the first set of flip-flops by the clock. The latter are in turn strobed into the second set of flip-flops, also by the clock. This double bufferring is done to minimize "synchronizer" problems. In order to compensate, the signals going into the muxes are rotated once to the right. This way the first set of flops gets loaded with the signals for motor N+1 at the time that motor N is being worked on.

The bus logic contains the UNIBUS^{TM} drivers and receivers together with address decoder and a data output register. The controller can respond to data-in and data-out cycles, but never becomes bus master. Its UNIBUS^{TM} address block consists of 32 consecutive words in I/O space. These words are precisely the 32 register file locations. All can be read and written via the bus, although writing is inhibited in certain bits. This block can be located on any 32-word boundary in I/O space by means of the address select switches, X(12-06). Controller bus cycles cause the flop BUSCYC+0 to be set. This in turn is passed through synchronizing logic to create BUSCYC+G, which either causes bus data to be written into the file, or data to be read from the file to the data-out register, DR(15-00). The controller processes bus cycles only at one point in its cycle flow, resulting in a bus delay of up to 2 μs(worst case).

One pass of the controller cycle flow is shown in fig. 13. A single motor, and the associated file locations are updated in the process. Eight such passes constitute a complete update cycle(10.71 μs). The cycle being executed is determined by the incrementing state counter, SC(3-0); the motor being operated on, by AR(4-2). The rightmost two bits of AR address the 4 file locations of given motor.

3.2 Feedback Interface Card

A block diagram for this card is given in fig. 14.

Three outputs(A and B phases and index mark, X)from each shaft encoder come in on short twisted pair lines. These signals are fed into line drivers(26LS31)which transmit the information to the controller over a 26-conductor flat cable through series terminated(90 ohm)lines.

Current mode limit switch signals(FL and RL)also come in on short twisted pair lines. These signals are received by current sensing devices(74LS63)and then transmitted to the controller over the flat cable. The optical limit switches are mounted on small specially designed printed circuit cards, which facilitate wiring.

The feedback interface cards are provided with a separate 5 volt power supply, which also services the shaft encoders and the optical switches.

3.3 Control Interface Card

A block diagram for this card is given in fig. 15. It occupies a single slot in the motor driver backpanel.

Eight pairs of parallel terminated(180 ohm)lines come in on a 26-conductor flat cable from the controller. These signals, described in section 3.1, are received by 26LS32's. In each pass of the controller flow one of the 8 select lines is pulsed, sending a decoded command to the corresponding motor driver card. This information is transmitted over the wire-wrapped backpanel. The decoded signals are: FORW*(step forward), REVR*(step reverse), LOAD*(load coil state), ST0 and ST1(coil state), and SEL0* through SEL7*(channel select pulses).

3.4 Limit Switch Cable Card

This card occupies a single slot in the motor driver backpanel. The limit signals, originating on the encoder/limit switch cards, are wired straight through the controller onto a single 26-conductor flat cable, and out to the backpanel pins via this card. There are 16 limit signals: RL0 through RL7(reverse limits)and FL0 through FL7(forward limits). Signals from different channels are separated by ground wires. The forward and reverse limit signals from each channel are carried on adjacent wires. Since a transition of either one implies recalibration of the channel, cross talk between the two does not present a problem. Moreover, since the limit signals are active high, all signals on the cable are normally low. The system is fail-safe in that pull-ups will cause a limit signal to be asserted should a cable come unplugged.

3.5 Motor Driver Card

A block diagram for the motor driver card is given in fig. 16.

It exploits the coil pairing scheme referred to in section 1.1, and places each pair in series with a current source, making separate the problems of current control and coil switching.

The artwork of the motor driver PC card supports three different current control schemes: chopper, dual voltage, and resistive.

In the chopper scheme, the current source is a transistor connected to a voltage source between 40V and 100V. A current sensing circuit, with hysteresis, provides a chopper mode control for the transistor. When the transistor is on, the supply voltage is applied to the coil. When the transistor is off, the coil current free-wheels from ground through a diode. In order to prevent the chopped current from being coupled into the off coil, a diode is placed in series with each phase. This circuit is borrowed from a driver card designed by John Roe. The chopper operates in the 2KHz-4KHz range.

The dual voltage scheme is identical to the chopper, with the exception that the free-wheeling diode is connected to low voltage supply rather than ground. The value of the low voltage depends on the motor type, and must be such that the steady state coil current lies in the dead-band of the current sensing circuit. There is no power sequencing requirement on the supply voltages, due to the chopper.

In the resistive scheme, the current sensing circuit is omitted and the current source transistor is replaced by a resistor. The value and power rating of the resistor depends on the supply voltage and the motor current.

The coil switching circuitry is the same for all three current schemes. High voltage darlington transistors are used to switch the individual coils. A zener diode between base and collector limits inductive spikes. Diodes between the collector of each darlington and ground provide a path for current coupling within the bifilar windings. Some transistors, such as Motorola MJ10002, have built in collector-emitter diodes, eliminating the need for the external diodes. Optionally, the diodes can be placed in series with the coils to block current coupling. This is necessary only for the chopper scheme. Typical coil current and voltage waveforms are shown in fig. 18.

A small amount of logic resides on the card. A counter is provided for storing the coil state in addition to logic to permit incrementing, decrementing, and loading of the counter. The limit lines can inhibit these operations.

Voltage levels for driving the coils come in on lugs on the rear edge of the card. The high voltage should not exceed 100 volts. The low voltage(dual supply scheme only)depends on the DC voltage rating of the motor. +5 volts is provided via the backpanel.

3.6 Motor Driver Backpanel

Slot allocation is shown in fig. 17.

The backpanel provides power(+5 V), mounting and interconnect for driver cards, the control interface card, and the limit switch cable card. It it constructed out of double height DEC blocks.

3.7 Maintenance Box

The maintenance box provides the means for controlling the delay line oscillator clock, in addition to displaying machine state and creating sync pulses.

A block diagram is given in fig. 19.

The box is connected to the controller by a 26-conductor flat cable, with alternating signals and grounds. Eight bits of machine state(in this case, SC(3-0), AR(4-2), and one spare)are received and buffered. The buffered signals are displayed as hex or octal digits depending on a jumper dip(A11). The same signals also feed two comparators. Here they are compared against two sets of switches to provide syncs and stop conditions. The clock(i.e. controller)can be either stopped or cleared on a given machine state. Once stopped, it can be single stepped or restarted. A local initialize can be triggered by a pushbutton on the box.

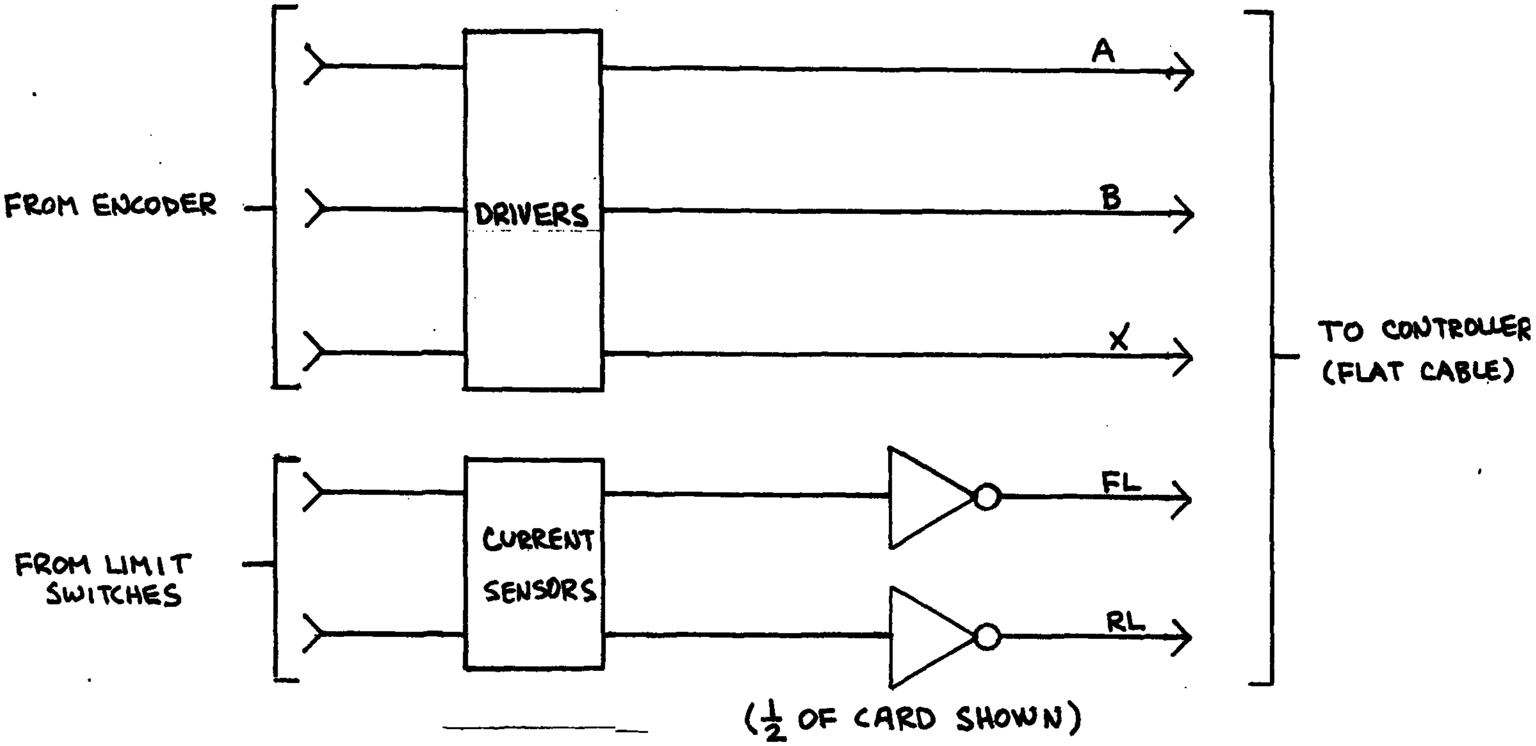


Figure 14. Feedback Interface Card

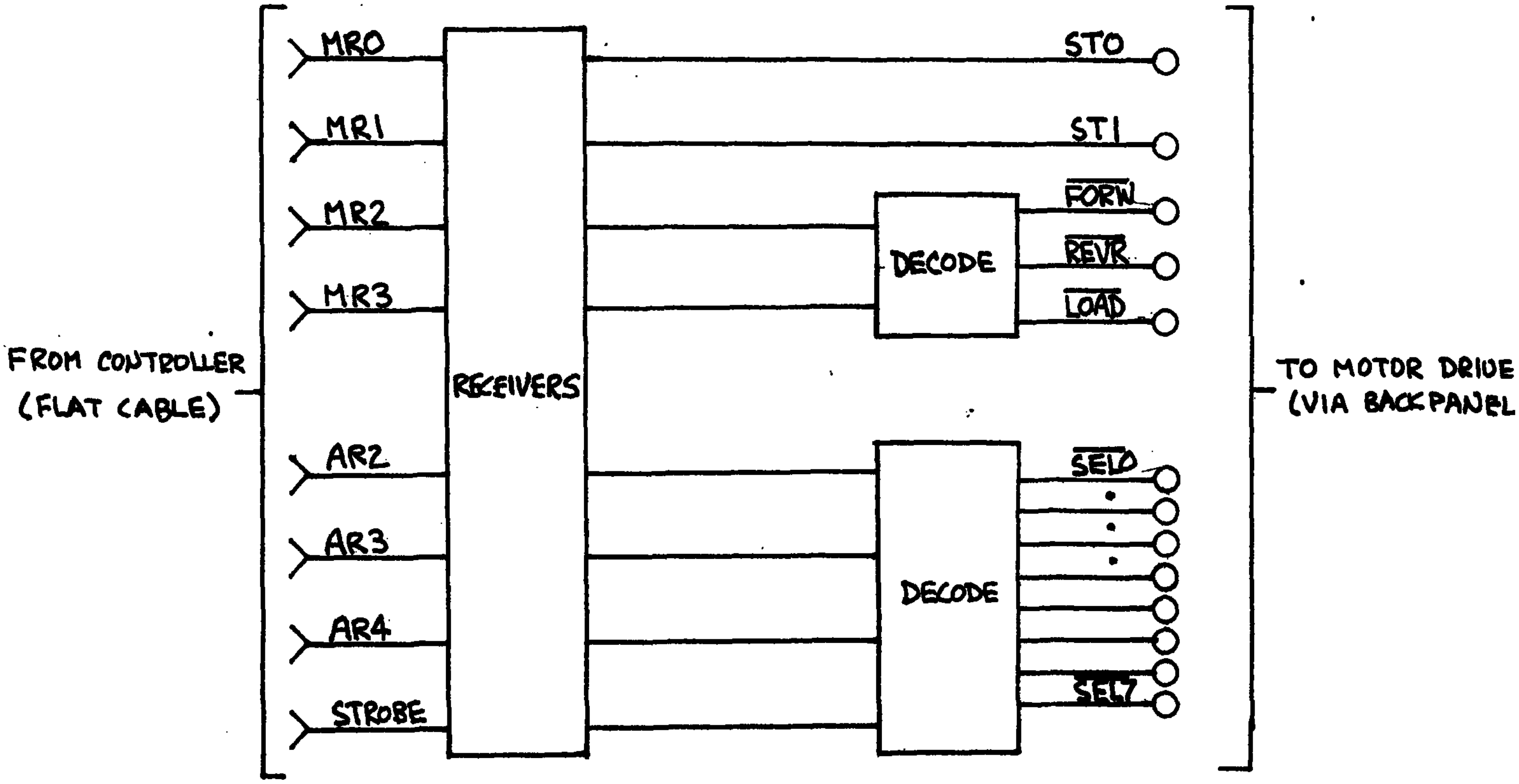


Figure 15. Control Interface Card

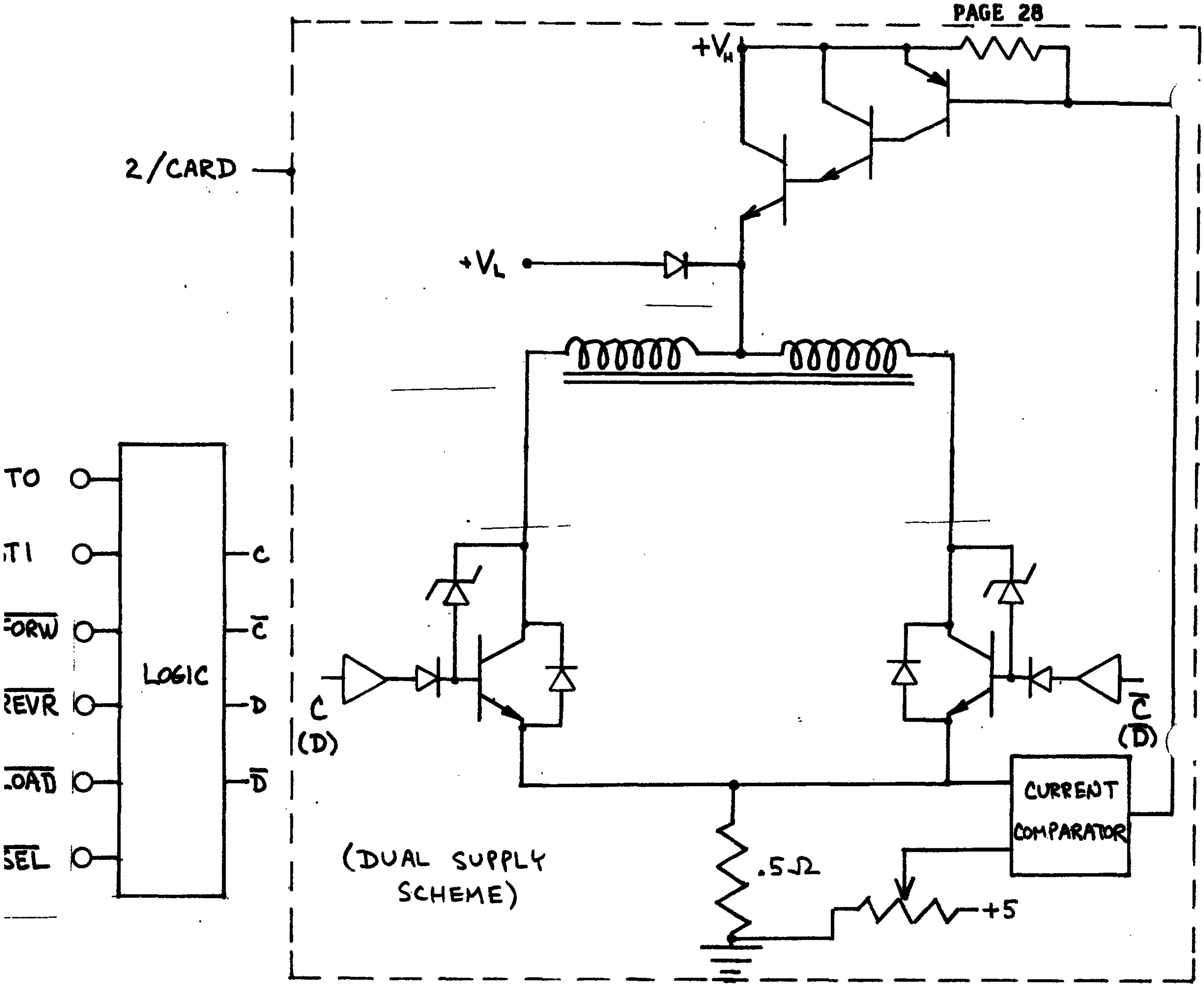


Figure 16. Motor Driver Card

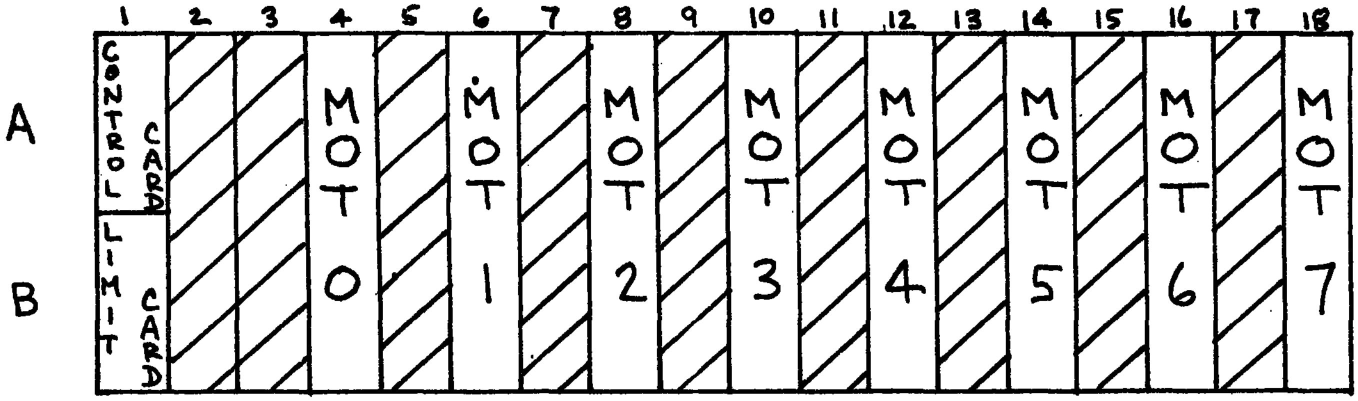


Figure 17. Motor Driver Backpanel(pin side)

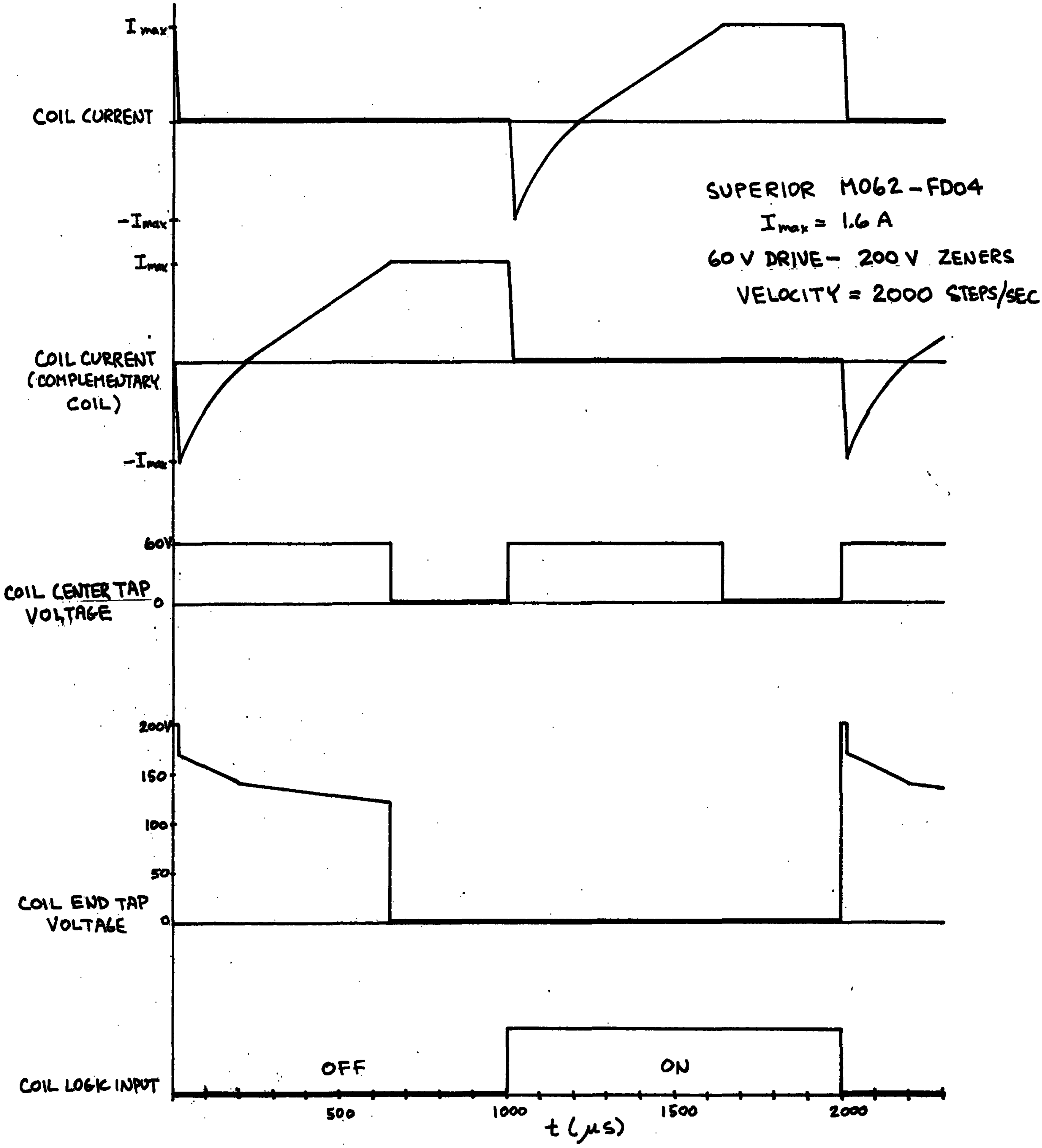


Figure 18. Coil Current and Voltage Waveforms

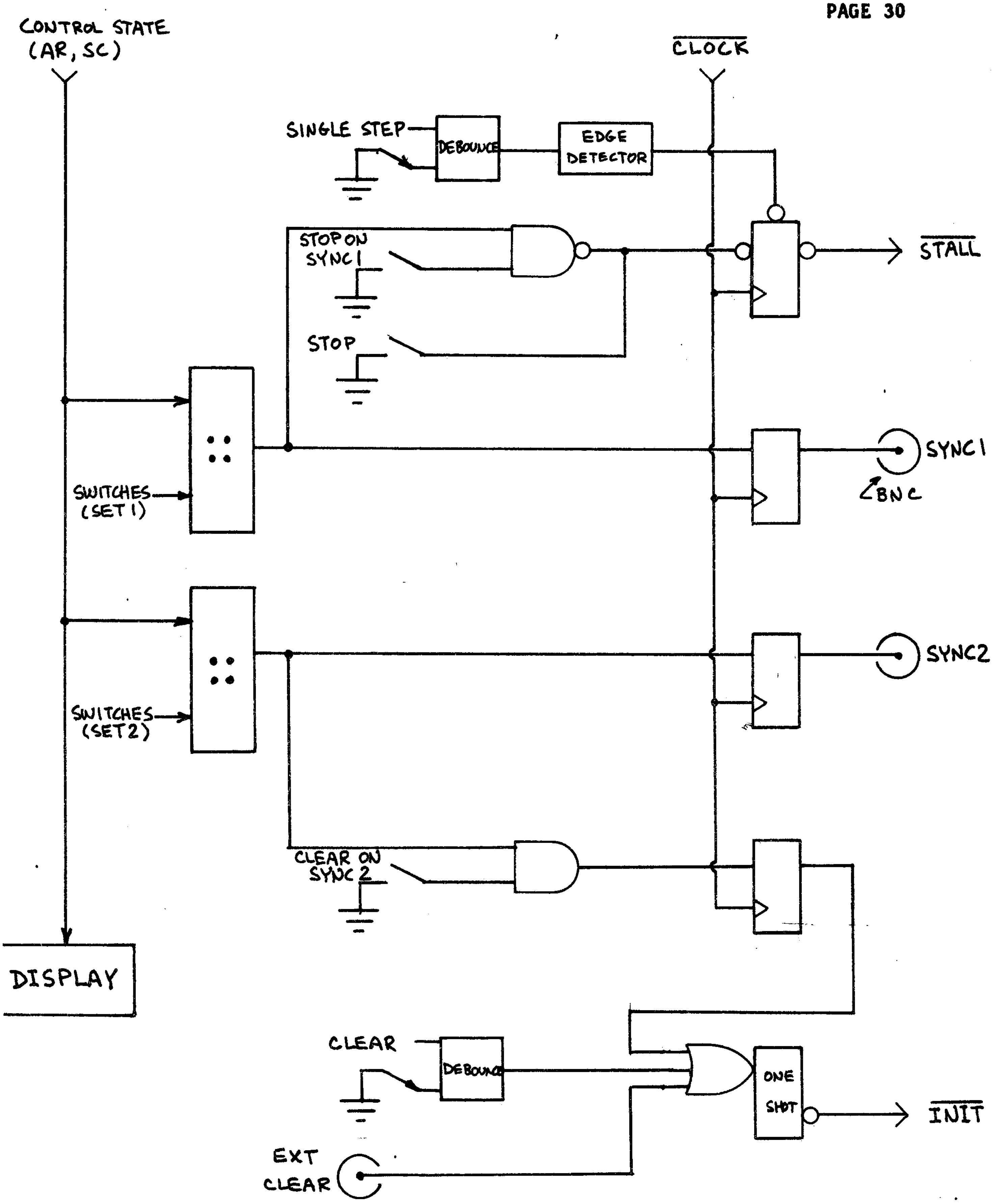


Figure 19. Maintenance Box Block Diagram

4.0 Programming

4.1 Software Interface

Software monitors and controls the stepping motors via a 32-word control block in I/O space. This block is divided into 8 equal sections, each of which corresponds to a motor channel. The layout of this block is shown in fig. 20.

The first word of each section represents the position, in quarter steps(encoder steps), of the channel. All except the rightmost two bits can be written. This is normally done only during initialization, and when the motor is at rest. If necessary, software can extend position to an arbitrary number of bits by maintaining an actual position record, separate from the hardware record.

The second word is the control word and is partitioned as shown in Table 1. The mode bits determine the control, if any, being imposed on the motor. Direct mode is further defined by the sub-mode field and can call for a no-op, a single step or an direct coil state load. The latter two operations will be performed exactly once, the sub-mode bits being cleared automatically by hardware. Lead angle mode causes the quantity in bits 3-0 to be enforced as a lead angle. The two open loop modes cause stepping to occur at regular intervals, determined by the count and the set count(see below). The tracking error indicator will stay on, once set, until either an initialize occurs or a byte write(e.g. CLRB)is done to this word with an operand having bit 7 off. The limit bit goes high in the event of either a forward or a reverse limit condition. The F/R/X bit allows software to distinguish between these extremes. In the absence of either it is equal to the index mark. When a limit signal is activated, further movement into the limit is inhibited. The joint can be moved out of the limit only by open loop or direct stepping. Since the limit signal may have blocked coil sequencing on the driver card, the coil state stored in the control word may no longer agree with the actual coil state. After moving out of the limit, a direct load should be done to remedy this situation. The use of the A-bit is explained in section 4.2. The meaning of bits 3-0 during mode 01 was explained above. In the other three modes it represents the phase angle as defined in section 1.3. However, upon leaving mode 01, this field may not be valid for up to 11 µs.

The third word is the open loop counter, used for decrementing set counts to generate steps. It can be read or written at any time. Upon entering open loop mode, it is first decremented to zero, at which time the motor is stepped. The set-count(fourth word)is then repeatedly loaded into the open loop counter and decremented to zero, the motor being stepped each time around. The counter is decremented at 10.71 µs intervals. Before entering open loop mode it should be loaded with the number of such intervals desired prior to the first step.

The fourth word is the open loop set-count. It determines the number plus one of 10.71 µs intervals between steps. The step rate(steps per sec.)can be calculated from the formula:

Rate = 93388.8/(Set-count + 1)

During open loop mode, it can be changed at any time. In this case, the current value in the open loop counter is first exhausted before the new stepping interval goes into effect. Set counts less than 5 represent stepping rates which exceed the tracking logic capability and should not be used.

4.2 Initialization and Calibration

The following steps will result in proper initialization and calibration:

1. Issue a master clear with all joints at rest.
   (This step should only be necessary following power-up.)
2. Move all joints slowly into reverse limits.
3. Move each joint 5 steps away from limit.
4. Direct load state 0 to all driver cards.
   (This guarantees that coil states are consistent with values in control word.)
5. Wait for all joints to settle
6. Move each joint slowly forward until index mark is on.
   At this point, all joints should be in predictable positions.
7. Load the desired initial value into the position word.
   Bits 1,0(R/O)are dictated by the encoder phases.
   Bits 3,2(R/W)are affected by the encoder index mark.
   E.g., if the encoder is aligned such that the index mark is on during coil state 0, then defining this position as step 0 would result in lead angle terminal velocities similiar to those in fig. 6.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	CHANNEL
BASE+0	POSITION														¼ STEPS		0
+2	MODE		SUB-MODE		LOAD STATE		COIL STATE		ERR	LIM	F/R/X	A	PHASE ANGLE				
+4	COUNT																
+6	SET COUNT																
+10																	1
+20																	2
+30																	3
+40																	4
+50																	5
+60																	6
+70																	7

Figure 20. Control Block

Bits	Name	Access	Use
15-14	Mode	R/W	Control Mode 00 - Direct 01 - Lead Angle 10 - Open Loop(forward) 11 - " " (reverse)
13-12	Sub-mode	R/W	Sub-mode for Direct Mode only 00 - Null 01 - Step Forward 10 - Step Backward 11 - Load Coil State (bits 13-12 are self-clearing)
11-10	State	R/W	State to be loaded(Direct,Load only)
09-08	Coil State	R/O	Current state of motor coils
07	Error	R/W	Tracking error(active high)
06	Limit	R/O	Limit condition(active high)
05	F/R/X	R/O	Meaning depends on state of bit 6: If bit 6 = 0, then bit 5 = index pulse If bit 6 = 1, then bit 5 = forward limit
04	A	R/O	Encoder A-phase When on, motor near middle of a step.
03-00	Phase Ang. (negation)	R/W	Meaning depends on mode(bits 01-00): If mode = 01, then bits(03-00) are the negation of lead angle(R/W) For all other modes, bits(03-00)are the negation of phase angle(R/O). Upon leaving mode 01, this field may not be valid for 11 µs.

Table 1. Control Word Format

4.3 Rotational Joints

Rotational joints can pose a problem in the position record keeping area. In particular, it may be desirable to define position as a periodic quantity. This can be done by maintaining actual position separate of the hardware record. Alternatively, the hardware record can be made to reflect the actual position. If the joint period is exactly 2**15 quarter steps, then there is no problem, since the position word will wrap around at exactly the right point. A similar situation exists for smaller powers of 2. For other periods it is necessary for software to change the position record dynamically, at the wrap-around point. In order to avoid hardware race conditions, this must be done in the middle of a step (encoder state 01 or 10, i.e. A-phase on). To facilitate testing for this condition, the A-phase is made directly accessible(R/O) in the control word. Once this condition has been detected, software has an interval of time dependent on motor speed to write the position register. At 15000 steps/sec., this interval is at least 12 μs. The following code will test for the condition and either load the position word, if the condition is present, or exit:

```
WRAP:   MOV     #POS,R4         ;address of position reg -> R4
        MOV     #CTL,R5         ;address of control reg  -> R5
        MOV     NEWVAL,R1       ;new position value      -> R1
        MOV     #20,R0          ;A-bit mask              -> R0
        BIT     R0,(R5)         ;test A-bit
        BEQ     EXIT            ;on ?
        MOV     R1,(R4)         ;yes
EXIT:   ....                    ;no
```

The last three instructions execute in less than 12 μs, on an 1145, as required.

Software should never change bits 3-2 of position after initialization. This will not be necessary, provided that rotational joints have periods which are exact multiples of four steps. If this is not the case, lead angle terminal velocities may change at wrap-around points.

APPENDIX

A Velocity Control Program

As an example of how the controller can be used, a velocity control program will be described. Its objective is simply to maintain a constant(programmable)velocity on a given joint, assuming this is physically possible, and in any event, apply torque in the right direction. Lead angle mode is used to accelerate or decelerate the motor to approximately the right velocity. The controller is then switched to open loop mode in the appropriate direction.

At the instant of mode transition, the phase angle is computed. An approximate lead angle for the desired velocity is compared against the computed phase angle. The difference is used to compute an initial count, prior to the first step. This procedure achieves a smooth transition to open loop mode.

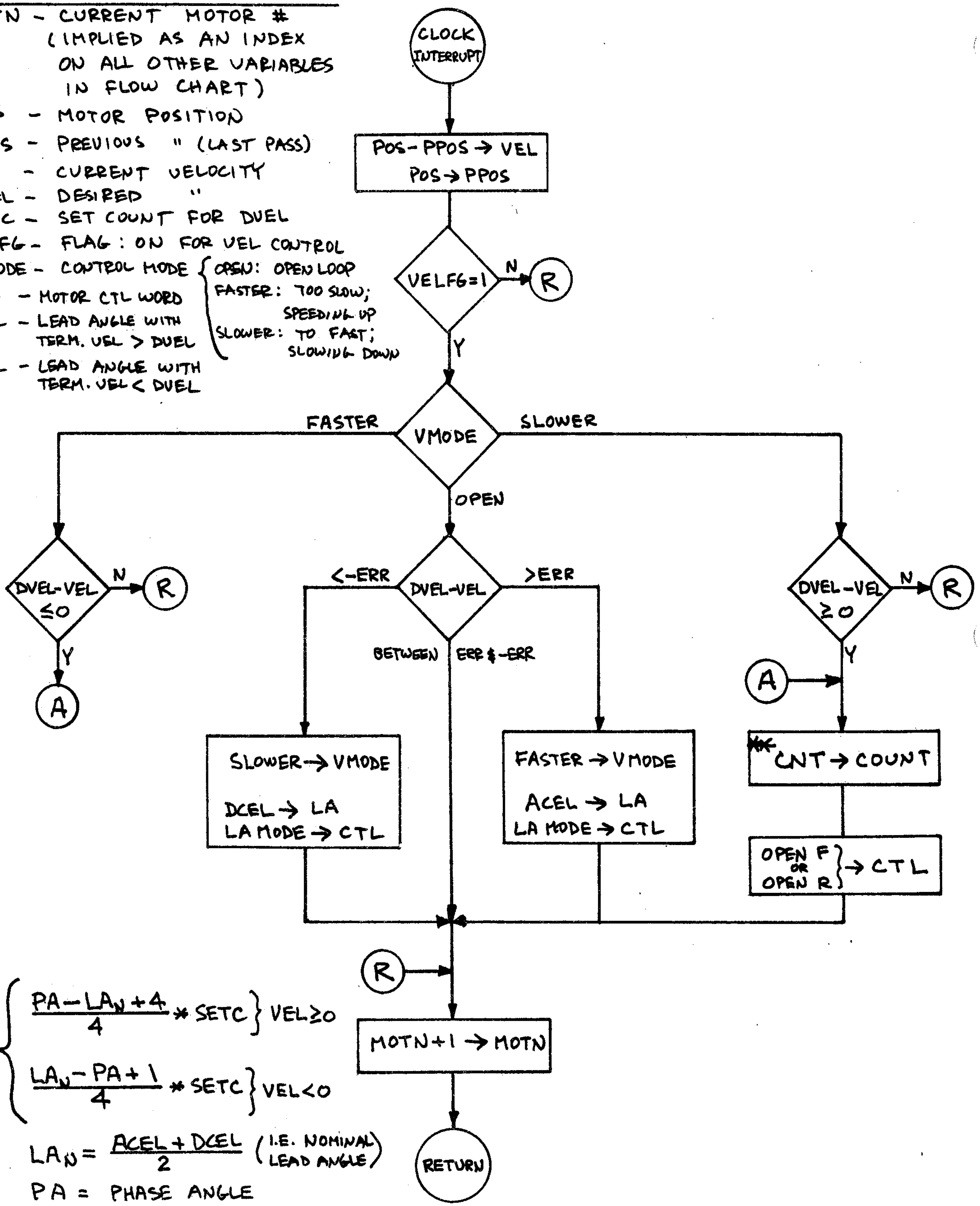
The use of lead angle mode to achieve velocity transitions avoids stalling. While the controller is in open loop mode, software continues to monitor motor velocity. Should the rotor show signs of slowing down or speeding up, lead angle mode is resumed to correct the error. This scheme takes advantage of the smoothness of open loop velocities, without sacrificing closed loop control. Moreover, the usual requirement of tachometers is eliminated. However, there is the drawback of being limited to the open loop stepping rates, which are spaced rather far apart at the fast end(see Table 2).

The program can control eight motors simultaneously, achieving stable velocities up to 10000 steps per sec. The degree of stability is dependent on the open loop behavior of the driver cards and the motors. Transitions between velocities are quite rapid and exhibit little if any overshoot.

The program was implemented on an 1145. It is driven by the programmable clock at 2000 Hz, servicing one motor on each tick (i.e., each motor is serviced at 250 Hz). Approximately 15 % of the processor time is used by the program. A flowchart is given in fig. 21.

VARIABLES

MOTN	- CURRENT MOTOR # (IMPLIED AS AN INDEX ON ALL OTHER VARIABLES IN FLOW CHART)
POS	- MOTOR POSITION
PPOS	- PREVIOUS " (LAST PASS)
VEL	- CURRENT VELOCITY
DVEL	- DESIRED "
SETC	- SET COUNT FOR DVEL
VELFG	- FLAG : ON FOR VEL CONTROL
VMODE	- CONTROL MODE { OPEN: OPEN LOOP FASTER: TOO SLOW; SPEEDING UP SLOWER: TO FAST; SLOWING DOWN
CTL	- MOTOR CTL WORD
ACEL	- LEAD ANGLE WITH TERM. VEL > DVEL
DCEL	- LEAD ANGLE WITH TERM. VEL < DVEL



$$* \quad CNT = \begin{cases} \dfrac{PA - LA_N + 4}{4} * SETC \Big\} \; VEL \geq 0 \\ \dfrac{LA_N - PA + 1}{4} * SETC \Big\} \; VEL < 0 \end{cases}$$

$$LA_N = \frac{ACEL + DCEL}{2} \quad \text{(I.E. NOMINAL LEAD ANGLE)}$$

PA = PHASE ANGLE

Figure 21. Velocity Control Algorithm

Velocity(SPS)	Set Count(octal)
15565	5
13341	6
11674	7
10377	10
9339	11
8490	12
7782	13
7183	14
6671	15
6226	16
5837	17
5493	20
5188	21
4915	22
4669	23
4447	24
4244	25
4060	26
3891	27
3735	30
3591	31
3459	32
3335	33
3220	34
3112	35
3012	36
2918	37
2830	40
2747	41
2668	42
2594	43
2524	44

Table 2. Open Loop Velocities(> 2500 SPS)