DESIGN AND PROGRAMMING OF A LOBSTER ARM ROBOT

By

VAHID TABAN

Bachelor of Science

Oklahoma State University

Stillwater, Oklahoma

1982

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements
for the degree of
MASTER OF SCIENCE
May, 1986

DESIGN AND PROGRAMMING OF A

LOBSTER ARM ROBOT

Thesis Approved:

Thesis Adviser

Dean of the Graduate College

PREFACE

The objective of this project is to design and fabricate a four degrees of freedom Lobster arm robot. Furthermore, it is intended to develop proper software for a powerful and efficient operating system. The model is to serve as a test-bed for future research concerning kinematics, dynamics, and control of closed-loop robots. Task-programming techniques of open-loop vs. closed-loop robots is to be studied to evaluate the difficulties that may be encountered. Methods to achieve speed and efficiency of actuator control will be studied in detail. Once the robot design is complete and detailed research has been carried out, it is expected to have many industrial applications.

I would like to express my sincere appreciation to my thesis advisor Dr. A. H. Soni for his continuous guidance and encouragement throughout this project. Also I would like to extend thanks to my major advisor Dr. B. L. Basore for his kind assistance and generous contributions which made this project possible. Thanks are also due to Dr. R. L. Lowery and Dr. J. Nazmetz for their active support on this project.

A special thanks to my parents for their enduring love and moral support.

TABLE OF CONTENTS

Chapter Page

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Closed-loop robots often have the advantage of higher speed capability and higher precision in comparison to open-loop robots. Also the closed-loop robots tend to handle high inertia load conditions more satisfactorily than open-loop robots. However, a closed-loop robot does suffer from some important drawbacks. Most closed-loop robots have a more limited and somewhat smaller workspace and their highly nonlinear equations of motion are time consuming to solve demanding powerful computers to control the robot. Applications where the speed of operation is critical, make closed-loop robots a more attractive alternative.

The Oklahoma Crawdad which is often called a Lobster arm robot has four degrees of freedom. As shown in Figure 1, it consists of a three degrees of freedom planar mechanism and a prismatic elbow with a gripper mounted at one end. The mechanism was previously modeled mathematically and computer programs were developed to simulate the mechanism graphically [1]. The mechanism has two ternary links (including the ternary ground link) and six binary links. The elbow is installed on the ternary

output link of the mechanism such that it can slide up and down along the vertical axis. As shown in Figure 2, the entire assembly is mounted on a rigid and sufficiently large work surface. A small teach panel is mounted on one corner of the work surface within easy access of the user. Shown in Figure 3 is the separate box containing power supply and the drive circuits necessary to control the various motors. The computer interface board, which also contains the A/D, D/A, and decoding circuits for shaft-encoder, is mounted inside the computer. Connecting cables are used to attach the power supply box to the computer and the robot. Three stepper motors are used to drive the input links of the mechanism whereas DC servo motors are used for the elbow and gripper. Menu driven software is developed for easy interaction with the user. In control modules of the software where speed of execution is critical, assembly language is used. However, Pascal is used for the remaining software modules.

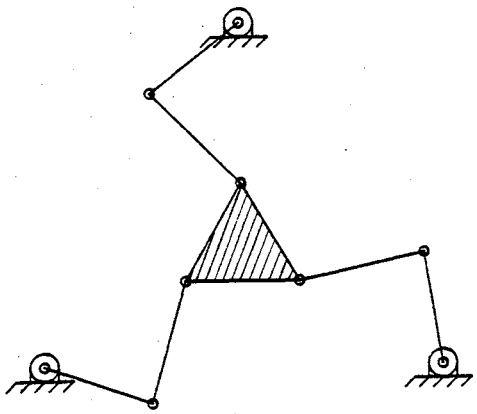


Figure 1. Lobster Mechanism

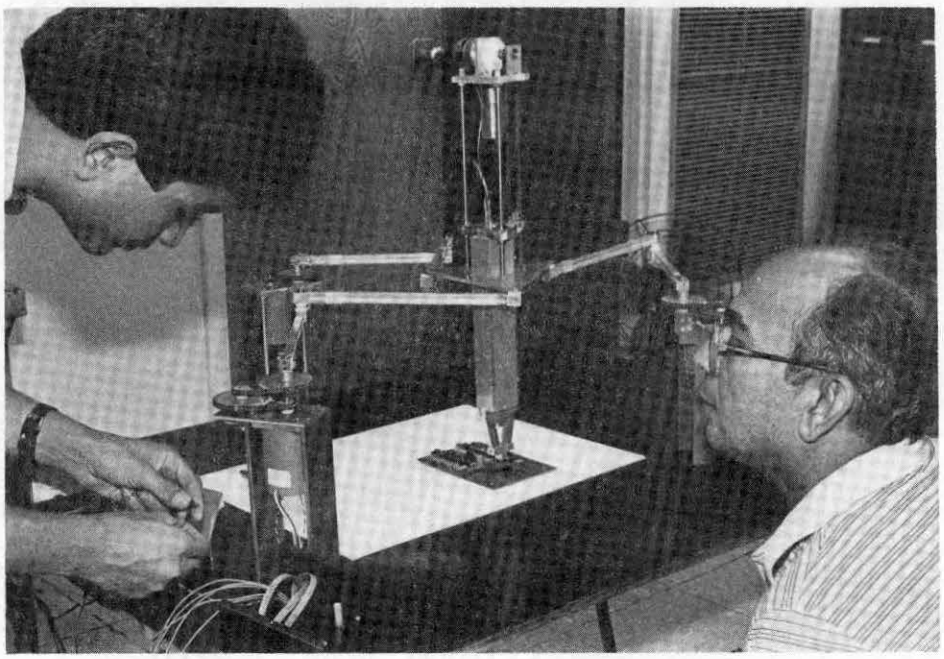


Figure 2. Crawdad Robot

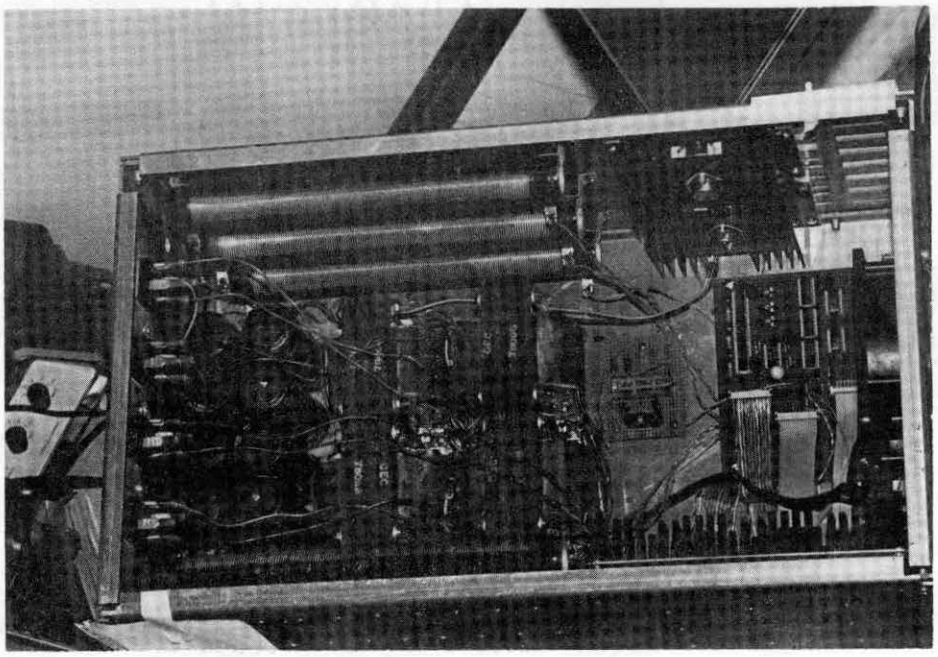


Figure 3. Power Supply and Motor Controllers

CHAPTER II

DESIGN AND CONSTRUCTION

Three stepper motors are used to drive the base inputs of the lobster mechanism. The movements of the Z-axis and the gripper are controlled by employing DC servo motors. For detailed description of drive electronics and software refer to appropriate sections.

A. Bases

Aluminum sheets of 3/16" thickness are used to construct the bases. Each motor is mounted on a horizontal piece which itself is supported by two long vertical plates. A 54" by 36" by 3/4" high strength phenolic fiber board is used as a mounting table for the three bases.

In order to gear down the motors, two identical pairs of chain and sprockets, each with a gear ratio of 3:1 are used to obtain a total gear ratio of 9 for each base. Precision sprockets and specially constructed chains (combination of steel belt and plastic jacket) produced a zero backlash gear drive which is critical when using open loop (no feedback from the actual output) devices such as stepper motors. The output bearing block is mounted on one of the vertical plates. The idler bearing block is mounted on the horizontal plate such that its position can be

adjusted to allow for tightening of the chains.

B. Links

Three pairs of binary links connect the center equilateral triangle link and the three bases or inputs of the mechanism. Input links are slightly shorter than the coupler links, but they all have "I" beam type cross section of the same size. Double ball bearings are used at each end to ensure rigidity of the mechanism.

C. Elbow

The elbow consists of a 1.5" by 1.25" by 8.5" piece and a gripper mounted at the bottom end. The elbow is constructed from four pieces of 1/8" sheet aluminum to make a box shape housing for the gripper motor and the drive components. Weight is reduced wherever possible.

The gripper is constructed from two sets of parallelograms to form two fingers such that the gripping surfaces remain parallel independent of the position of the fingers. Pulleys are used to amplify the gripper force. A worm is mounted on the motor shaft, which turns a worm gear at 1/25 of motor speed. A special high strength cable is wrapped around the worm-gear's 1/4" shaft. The cable passes through an idler pulley amplifying the force twice. The other end of the cable is fixed to the hand. The pulley is connected to another cable which in turn wraps around the feed back potentiometer and finally goes to the

gripper. The two gripper fingers are spring loaded to provide a return action.

D. Z-Axis

This part of the robot is responsible for up and down movement of the elbow. Three 1/4" steel rods are mounted on the ternary link. The other end of the shafts are fixed to another plate about 8" above the ternary link so that the rods remain parallel rigidly. The top plate is also used to mount the servo motor and the shaft encoder. A gear head with gear ratio of 5 is mounted on the motor.

The vertical output shaft of the gear head is engaged to a horizontal shaft by means of a pair of bevel gears reducing the gear ratio four more times. The shaft encoder is connected to this shaft by a small steel coupler. A small sprocket is mounted on this shaft to drive a loop chain up and down. The other end of the chain passes over an identical idler sprocket which is mounted just below the ternary link. The elbow itself is mounted on another plate which translates along the three vertical shafts using three linear ball bearings. This plate is fixed to the chain such that the motor can drive the elbow up and down.

E. Sensors

A number of different sensors are necessary to achieve proper operation of the robot. The obvious sensors are the feedback potentiometer for gripper and the shaft encoder for the Z-axis servo system. The gripper employs a single

turn wire wound 5 K Ohm potentiometer and the Z-axis has a 4800 pulse per revolution shaft encoder. A limit switch is used to sense the gripper force. This design prevents over-tightening of the gripper.

Three pairs of semiconductor strain gages are mounted on each of three binary input links, so that when in the teach mode the user can apply a small force in the desired direction, causing a small stress in the binary links. The computer can read the strain gages and drive the particular motors in the direction which the force is being applied. This will enable the user to move the robot from one place to another without having to solve the complex sets of equations for the Crawdad arm in real time.

CHAPTER III

STEPPER MOTORS

A stepper motor provides precisely controllable speed or position. Since the motor increments a precise amount with each control pulse, it easily converts digital information to exact incremental rotation without the need for any feedback device such as a shaft encoder or potentiometer. Any stepper system requires initial calibration when power is first applied; therefore, a home or a reference position is necessary. Each motor has four phases which must be turned on and off in a determined pattern in order to rotate the motor shaft in a clockwise or counter clockwise direction.

There are many different approaches to drive a stepper motor each with some advantages or disadvantages. Unipolar phase energizing with half step phase pattern generation is used here because of superior speed capabilities [2].

A series resistor R is used to limit the steady state current to the stepper motor windings to the rated value (4.7 amps) and also to provide means to decrease the time for bringing the current up to full value. The time constant (time for current to rise to 63% of max) is equal to L divided by Rtotal. Rtotal is equal to series resistance plus the transistor on-resistance (.18 Ohm) plus

the motor winding resistance (0.34 Ohm), where L is the motor winding inductance (0.81 mH). 10 ohm 250W resistors are used for series resistor R. Power MOSFETs (Metal Oxide Semiconductor Field Effect Transistor) are used to switch the power. A special power down feature is employed to reduce the power when in steady state.

CHAPTER IV

IV- SERVO MOTORS

The purpose of a closed-loop position control system is fast and accurate control of the rotational position of the motor shaft. Rotational position is given the notation of θ , which signifies the angle deviation from a set reference angle [3]. θ is equal to integral of the angular velocity. The block diagram of a position control system is shown in figure 4.

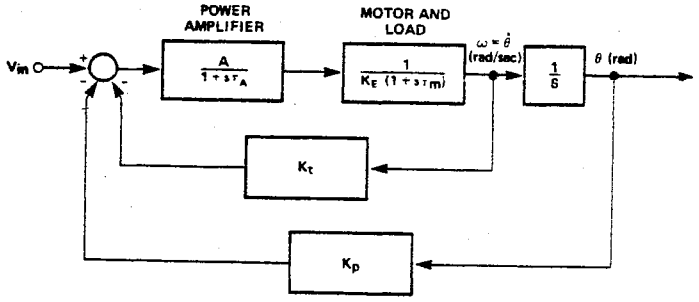


Figure 4. Position Control System

And the transfer function is as follows:

$$\frac{\theta}{V_{in}} = \frac{\frac{1}{S}\left(\frac{A}{1+S\tau_A}\right)\left[\frac{1}{K_E(1+S\tau_m)}\right]}{1+\frac{K_tK_p}{S}\left(\frac{A}{1+S\tau_A}\right)\left[\frac{1}{K_E(1+S\tau_m)}\right]}$$

Where:

Kp = Integrator constant
(position feedback gain)

Kt = Velocity feedback gain

Ke = Voltage constant of motor

T_m = Motor time constant

T_A = amplifier time constant

A = D.C. gain of amplifier

Velocity feedback increases system damping and allows for faster response while preventing overshoot which is highly undesirable in robot applications of servo systems [4] (electrical, hydraulic or otherwise).

To employ velocity feedback in a servo system a tachometer is necessary. In addition to cost, a tachometer will increase the weight and size of the elbow. Another approach is to differentiate the signal from position feedback device, but practical differentiation of a signal electronically is not an easy task. This discussion is pursued in more detail in the next sections.

A. Gripper Servo

Since the gripper employs a relatively high gear ratio and low inertia, velocity feedback is not used and position feedback is considered to be sufficient. This makes the hardware necessary to control the motor simple.

B. Z-Axis Servo

The Z-axis joint has considerable inertia since it must support the elbow inertia including the bearings, the plate housing them, and the load carried by the gripper. This could cause an overshoot. Overshoot can cause serious damage to the arm or to the object being handled. Therefore some means of velocity feedback must be employed to obtain sufficient damping and fast response. A tachometer however is not used for this purpose. Instead, the position is read by computer at equal time intervals, then the previous position is subtracted from present position. This time interval can be very short (about 5 ms) since the shaft encoder for position feedback is a very high resolution device.

The velocity feedback gives good dynamic characteristics, but the steady state errors must be reduced. The steady state errors can be considerably high when heavy objects are being handled. Therefore a simple integrator is employed in the controller to minimize these errors.

CHAPTER V

ROBOT OPERATION

A. Teach Mode

Probably the most critical part of operating a robot is teaching it to do a series of operations sequentially. To do this a user must have total control over each function of the robot arm while in teach mode including moving the arm around, bringing the elbow up and down, opening and closing the gripper and finally being able to tell the computer to remember a particular state of the robot (position of the three bases, position of the Z-axis, and position of the gripper). A typical teach session would produce a set of numbers stored in a large array in the following format. Three 16-bit binary numbers to store the position of the three bases, one 8-bit number to store the position of the gripper and one 16-bit number to store the position of the elbow.

The length of the array determines the number of different points (robot states) that can be stored. When the teach mode is completed, this array can be stored on disk as a file to be called by the main program later, in order to execute the previously taught sequence. Full

editing capability is available to delete or insert one or more points in an existing file.

Three global 16-bit counters are incremented or decremented accordingly to be stored in current position of the array.

B. Teach Pendent

A Teach pendent shown in Figure 5 is used to control the rest of the robot functions. DPST (Dual Position Single Terminal) rocker switches with temporary action control the two servos in either direction. A decimal thumbwheel selector switch is mounted on the bottom side of the teach pendent. This selector determines the velocity factor (0-9). There is a push button to remember the current state of the robot.

User can move the servo in any direction by pressing the rocker switches, simultaneously if desired. The speed of the motors depend on the setting of the thumbwheel switch (0-9) and varies from very slow to maximum speed. This speed factor can be varied in different stages of teach session. It controls not only the speed of the servo motors, but also the stepper motors. For instance, when moving from one side of the workspace to the other side a fast speed may be used, but for positioning a rivet in a small hole very slow speed settings should give better results.

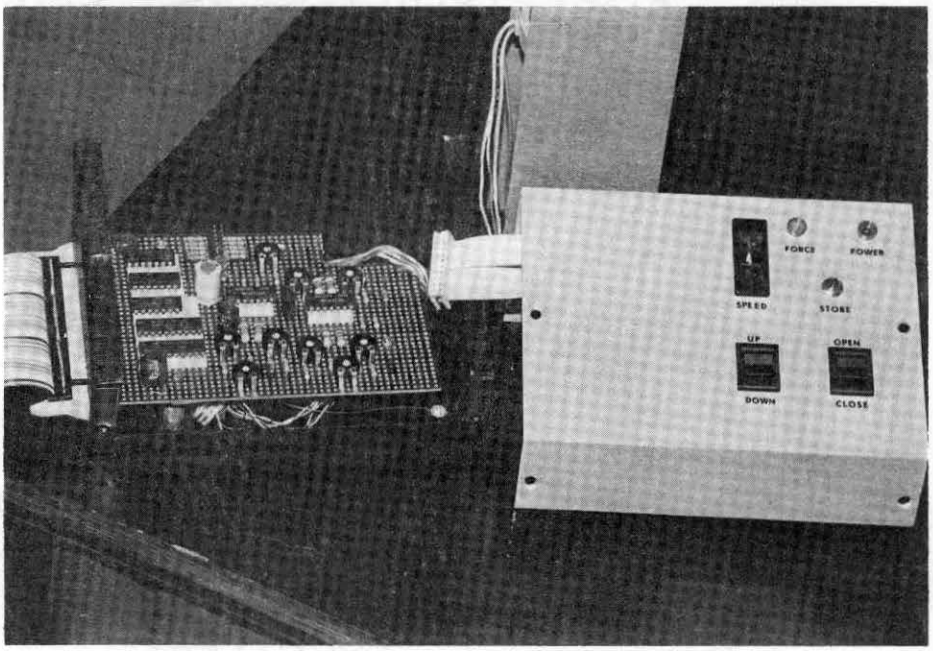


Figure 5. Teach Pendent

CHAPTER VI

SOFTWARE

Software is structured using Pascal language. Disk file handling and most of the user interactions are also done by Pascal. However all control functions, such as scanning the servo motors, stepper motor control, and various data acquisition routines were written in 8086-88 assembly language.

A. Servo Interrupt Service Routine (ISR)

This interrupt service routine reads the current position of each motor, subtracts this value from the desired position of the motor to calculate the position error and outputs the value of the error to corresponding D/A, which in turn connects to the motor's power amplifier. A timer chip on the interface board is used to interrupt the computer at 5ms intervals so that the above sequence is repeated exactly 200 times per second. The elbow incorporates an additional velocity feedback although it does not have a tachometer. Velocity is calculated by subtracting the current position of the motor from that of the previous scan. Since the time interval between each scan is fixed, the average velocity of the motor in 5ms

intervals can be calculated. This is possible only because a relatively high resolution position feedback device is used (about 20,000 counts for a full stroke of the elbow). This gives a sizable difference between the two scans even at slow speeds. The velocity is then multiplied by some gain Kv and then subtracted from position error. Higher damping is achieved by this technique without any additional cost, and the software overhead is minimal.

Additionally, to minimize the steady state errors an integrator is implemented on elbow servo system [4]. This is done by storing a few of the previous position errors and adding them together before sending the total error to the D/A. The number of these previous errors determines the time constant of the integrator.

This interrupt service routine is activated when the power is first turned on and repeated 200 times per second until the power is turned off.

B. Stepper Motors

Software necessary to drive a stepper motor is relatively simple. Since half stepping is used, the four phases of each motor must be turned off and on sequentially according to a table of length eight upward or downward depending on the direction desired. This is done by decrementing or incrementing a pointer variable for each motor in module 8. This variable is then used to look up the four bit phase values from the phase table mentioned

above.

However, the timing between the steps is critical. A Pascal module is developed to calculate the time interval necessary between each step according to a symmetrical Trapezoidal acceleration motion program. This module prompts the user to enter the slope of acceleration (jerk), the maximum acceleration and, the maximum velocity in kilo steps per unit of time. The program calculates the time between the steps to accelerate from zero velocity to the maximum velocity in micro seconds and stores that in a table called ACC/DEC table (acceleration/deceleration). These values are later used to load the external hardware timer with the correct number needed to generate desired delays between each step. Depending on the next or previous number loaded into the timer, acceleration or deceleration is obtained. Furthermore, loading the same value in the timer yields a constant velocity. This method produces smooth and jerk free start/stop characteristics which allows high speed operation (as fast as 8000 steps/sec.), also the user has total software control on motion characteristics of the stepper motors.

C. Drive

This procedure accepts one unsigned 16 bit number for desired position of the elbow servo motor, one unsigned 8 bit number for desired position of the gripper servo motor,

and three signed 16 bit numbers for the direction and number of steps each stepper motor must take. Servo motors are handled by their own interrupt service routine. For stepper motors the directions are set depending on the sign of the arguments. The procedure Drive steps the stepper motors and decrements the absolute value of the three arguments until they all become zeros. A special algorithm [5] is used to coordinate the motions of all motors such that they all reach their destination at the same time regardless of the difference in the number of steps necessary for each motor. This ensures smoother overall operation. The ACC/DEC table is used to accelerate or decelerate the motor with the highest number of steps only. The rest of the motors are moved accordingly. The drive procedure performs a number of checks to decide whether to accelerate, go constant velocity or to decelerate the motors.

D. Teach

Acceleration or deceleration is not used in teach mode of operation, since it makes the software complex, and slower speeds are sufficient in teach mode. The speed selector on teach panel decides the constant velocity of all three motors. A polling technique is used to scan the signals from the force sensors on each input link of the mechanism. Depending on existence of an external force

(user) the corresponding motor is stepped in the proper direction and a software counter is incremented or decremented to keep track of each motor. The polling routine is repeated from 50 to 250 times per second depending on the speed selector. This gives complete control over the stepper motors. The servo motors are controlled by two rocker switches on the teach panel as mentioned previously. If these switches are pressed, the corresponding servo moves in the proper direction with the speed selected. When the store push button is pressed, the necessary arguments are passed to the main program in the same format as the Drive procedure. These arguments are stored in an array to be recalled by the Drive procedure later.

CHAPTER VII

CIRCUIT DESCRIPTION

This section contains hardware description for each particular circuit board used in the Crawdad robot.

A. Stepper Motor Controller

Stepper motors are powered by three separate power supplies. One of the three identical power supplies is presented in Figure 6. Each power supply consists of a transformer, a bridge rectifier, and two 10,000 uF capacitors connected in parallel, which yield a 48 Volt unregulated power at 10 Amperes. Motor windings are rated at 4.7 Amps each. Since half stepping is used, either one or two of the motor windings are energized at all times. With the circuit parameters used, the current through each phase is approximately 4.0 Amps when the two phases are energized, but when only one phase is energized, the drop in load current results in a voltage rise in the unregulated power supply. This voltage rise in turn increases the winding current to approximately 15% above the rated value. This is due to the nature of unregulated power supplies, however, the motor specifications supplied by the manufacturer, allow an overdrive of up to 20% when only one phase is energized [2]. This yields a cooler

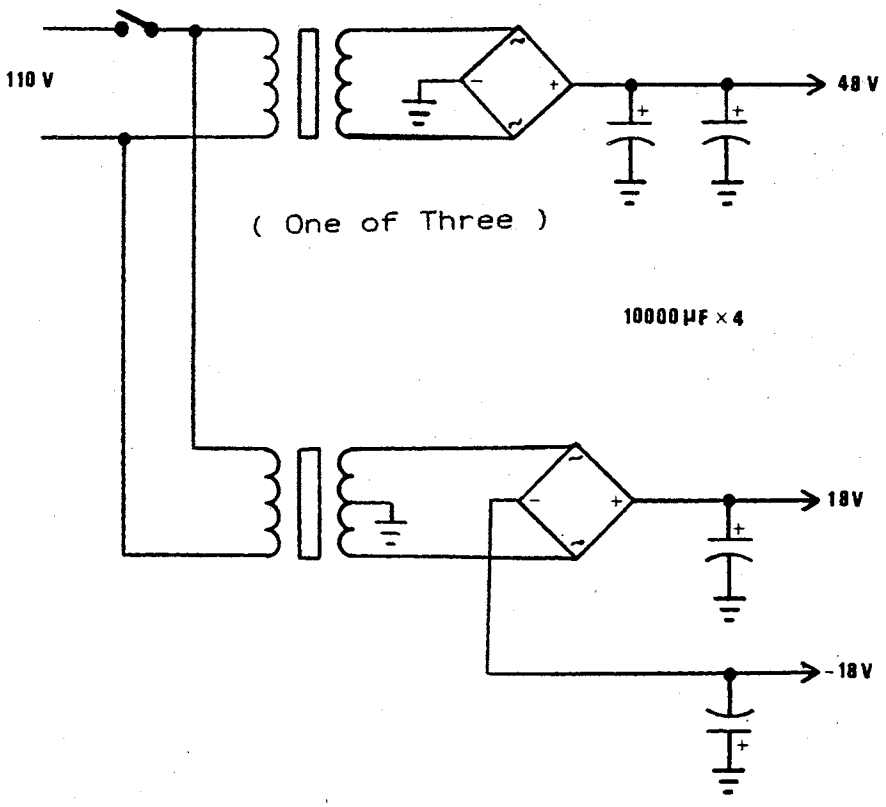


Figure 6. Motor Power Supplies

operation of the motor for a longer period of time while torque fluctuation of the motor shaft is minimized.

Figure 7 Shows the circuit diagram for one of the three identical stepper motor controllers. Power MOSFETs are used to switch the power to the motor windings for their superior speed and ease of implementation [6]. A zener diode is used across the gate and drain terminals of each MOSFET device along with a 1 kilo-ohm resistor to ensure safe operation. Four data flip/flops are used to latch the desired phase pattern. Open collector inverters are used to drive the gates of the power transistors. A one-shot timer is triggered each time a new phase pattern is latched (motor is stepped). The output of this timer is connected via a voltage divider to an op-amp's input which has a gain of 3. The time constant of the one-shot is slightly less than a second. This set-up produces about 10 volts on the gate of the MOSFETs for as long as the motors are being stepped more than once every second. But when a motor is not stepped for longer than one second the gate voltage of the power transistor drops to a value adjustable by a potentiometer. This allows for power reduction when the motor is in steady state conditions and the maximum torque is not required. Care must be taken when adjusting the low current of the motor windings because the power transistors may get too hot if the steady state current is reduced too much. Of course, the MOSFETs reduce their output current when they get overheated reaching a stable

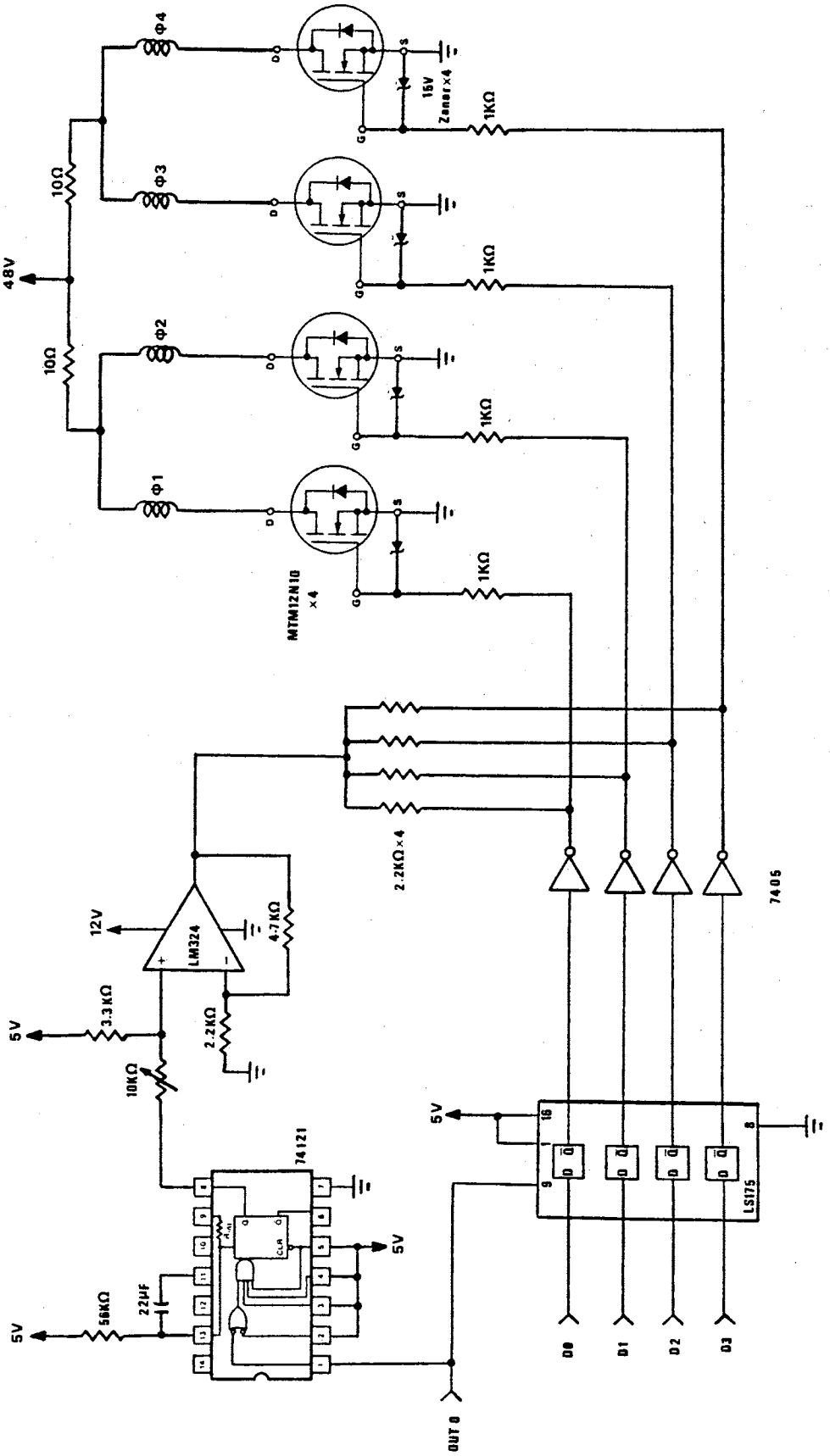


Figure 7. Stepper Motor Controller
(One of Three)

power dissipation due to their inherent characteristics which makes them almost foolproof [6].

B. Servo Motor Driver

Power supply for the servo motors is also shown in Figure 6. Plus and minus 18 volt unregulated power at 10 Amps is obtained from the circuit which is enough for three additional motors (the two motors used draw 2 Amps each). One of the two identical servo motor power amplifiers is presented in Figure 8. Power op-amps were used to simplify amplifier design for servo motors [7]. Tantalum capacitors are used near power supply pins of the op-amps to prevent oscillations. The gain of the devices are fixed to 12 for both of the motors. A special divider network is used at the input stage of each amplifier to obtain a full swing of the output voltage from -15V to +15V for an input voltage of -5V to 0v which is supplied by the digital to analog converters. Short circuit resistors of .47 ohms each are used to limit the current to a maximum of approximately 2.2 Amps.

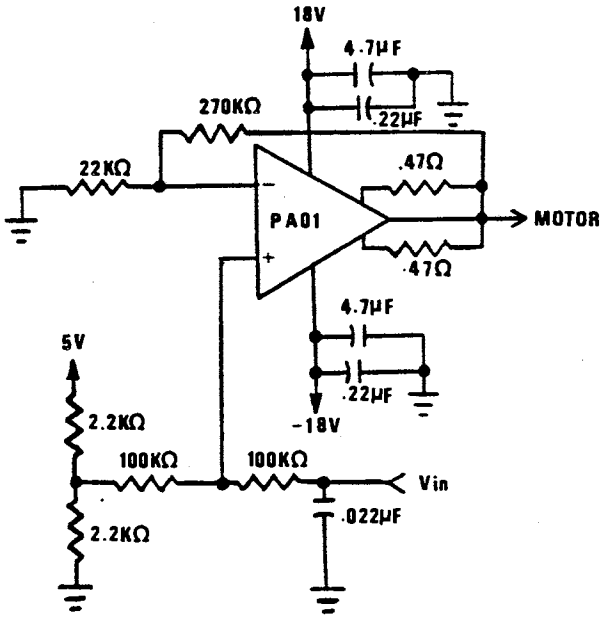


Figure 8. Servo Motor Power Amplifier

C. FORCE SENSORS

Semiconductor strain-gages are utilized to achieve high sensitivity. These strain-gages have a gage factor of approximately 100. This is roughly 50 times the sensitivity of resistive type strain-gages which have gage factors of 2 to 2.10 [8]. Nonlinear behavior of semiconductor gages is of no importance here, because only a preset level of force must be sensed not the magnitude of force at different times.

Wheatstone bridge and voltage comparators are used to amplify the signals from the force sensors so they can be monitored by the computer. The circuit diagram is shown in Figure 9. The bridge is temperature compensated and one potentiometer is used to balance each bridge and two more to adjust the sensitivity of each force direction individually.

D. Computer Interface

The complete circuit diagram for the interface board which is installed inside the computer is presented in Figure 10. It contains the components necessary to decode the address bus, components necessary to read the shaft encoder, the hardware timer, A/D and D/A's, and data bus buffer. Each section is discussed in detail here.

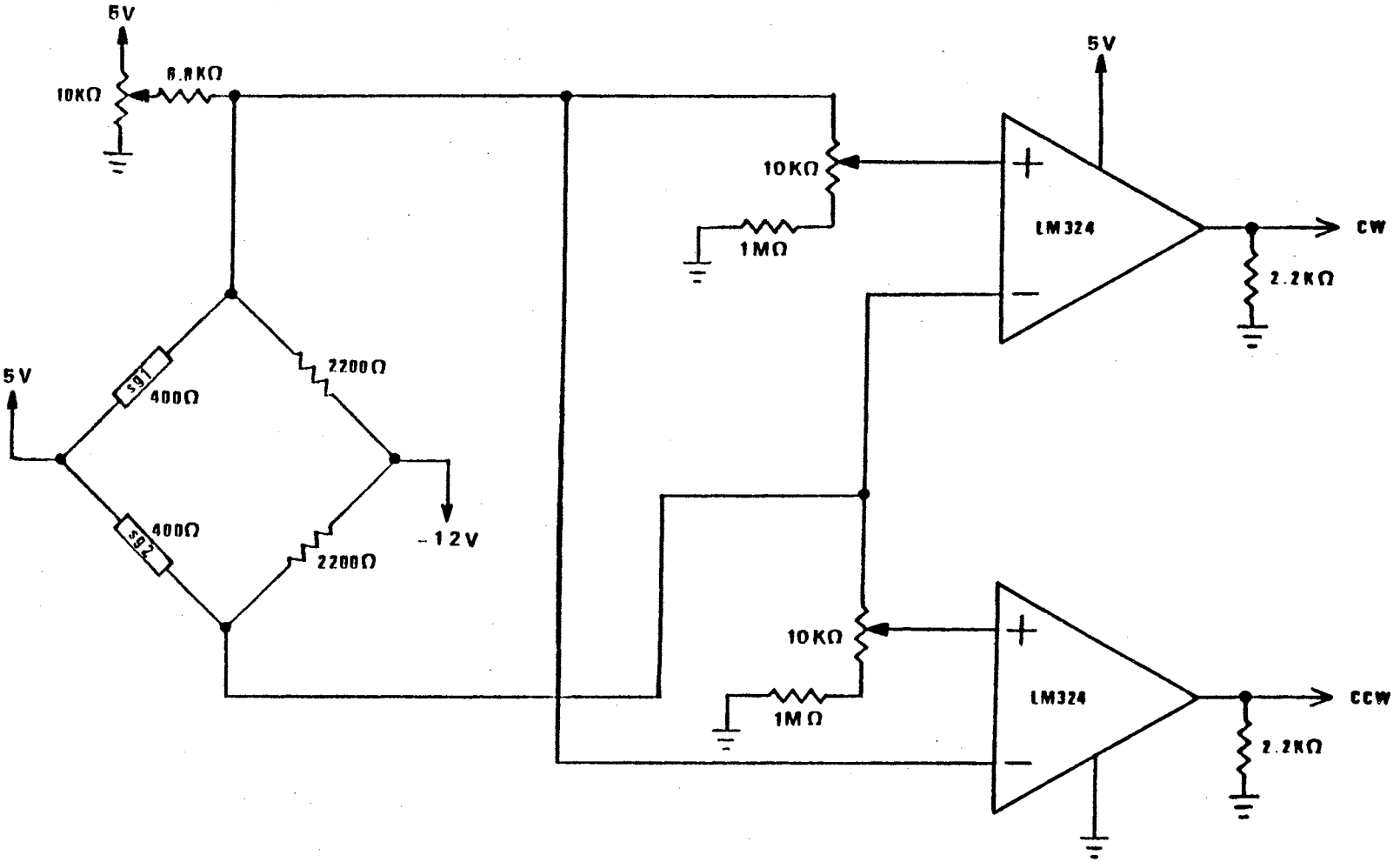


Figure 9. Force Sensor Interface Diagram

(One of Three)

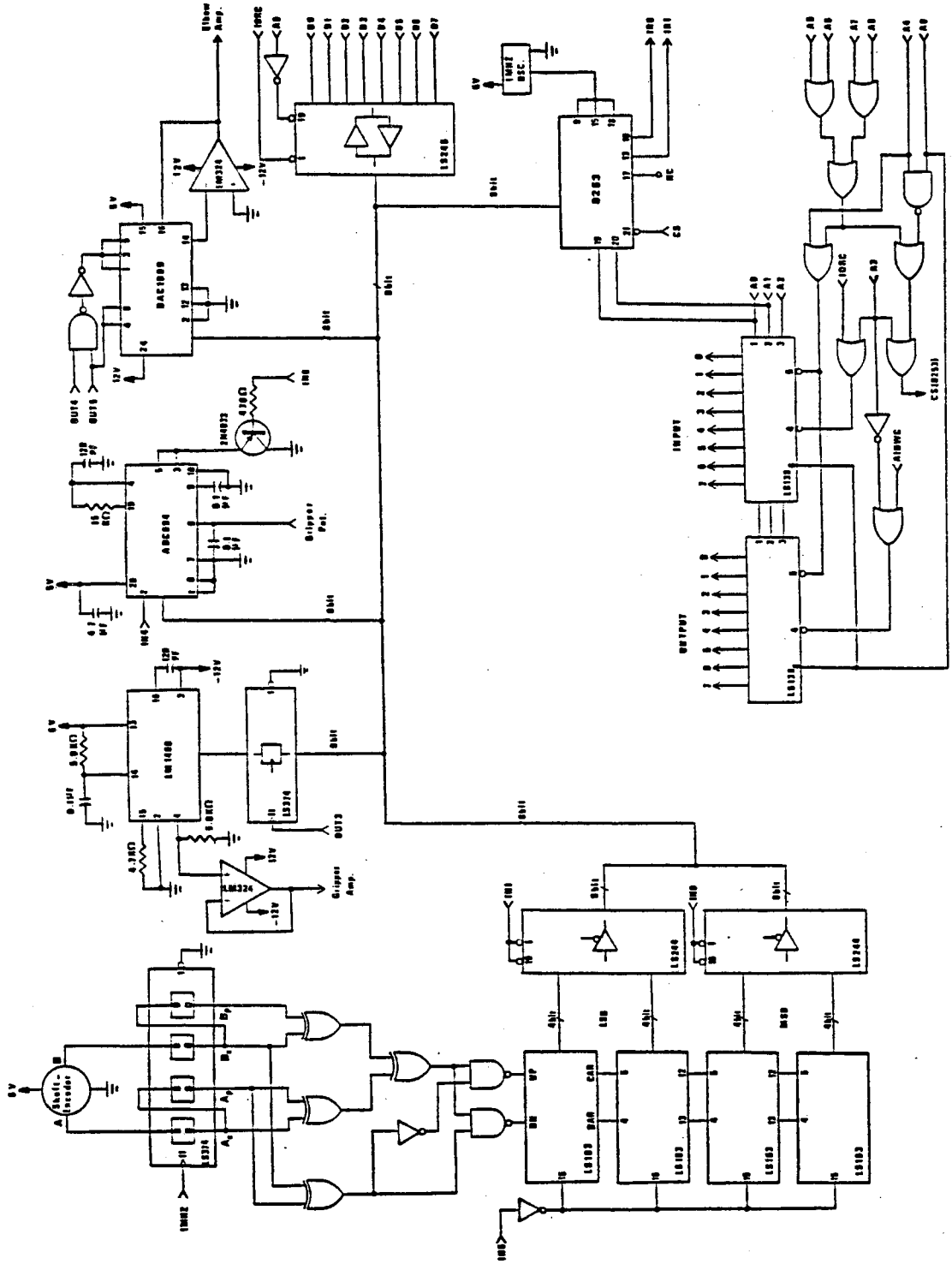


Figure 10. Computer Interface Board Schematic

i. Address Decoder

Two 3 to 8 bit decoders and a few logic gates are used to decode the address bus to obtain 8 input ports and 8 output ports for various devices necessary. Texas Instruments Professional Computer uses only 10 of the address bus pins for input / output purposes which allow for a total of 1000 I/O devices [9] . Therefore address lines A0 to A9 are used along with IORC and AIOWC pins to decode the necessary locations for I/O devices. A list of the location assignments in the I/O map is given in Figure 11.

ii. Shaft-Encoder

A two channel relative type shaft-encoder is used for the Z-axis position feedback. It produces 1200 pulses on each channel for a complete revolution. The pulses are 90 degrees out of phase which with proper decoding yield a resolution of 1/4800 of revolution. Refer to Figure 10 for shaft encoder circuit diagrams. Decoding is accomplished by comparing the present and previous logic states of channels A and B. The previous states of the channels are latched using four data type flip/flops which are clocked with a 1 MHz square wave. These four bits of digital information are decoded with proper logic to achieve a count-up and a count-down outputs. A 16 bit up/down counter is made by

cascading four 4 bit counters. It is used to keep track of the position of the shaft-encoder to more than 13 revolutions. The counter can be cleared whenever desired to allow a home position for the elbow. To clear the counters, an input command must be executed for number 6 input port. For proper address of this port refer to Figure 11.

iii. A/D, D/A, and Timer

An 8 bit A/D (ADC 804) is used in continuous conversion mode to read the position of the gripper servo. This chip must be initialized when power is first applied so that the continuous conversion process is started. This is done at the same time that the counters for the shaft-encoders are cleared.

A 10 bit (DAC 1000) and an 8 bit (LM 1408) D/A are used to drive the power amplifiers for the elbow and the gripper servo motors respectively. Design is straight forward and simple. A programmable timer chip (8253) is used with a clock speed of 1 MHz to allow precise timing functions necessary to control the motors [10]. Interrupt lines IR0 and IR1 are connected to output pins of channel 0 and 1 of the timer chip. Channel 2 of the timer is not used but the circuit board can be easily modified to use this channel for future applications.

		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		**********	I	N	P	U	T	P	O	R	T	S	**********
200H	S.E. High	F	E	D	C	B	A	9	8
201H	S.E. Low	7	6	5	4	3	2	1	0
202H	Panel	Open	Close	Down	Up	Speed Selector
203H	Force Sen.	F3CW	F3CCW	F2CW	F2CCW	F1CW	F1CCW	Limit	Store
204H	8 Bit A/D	7	6	5	4	3	2	1	0
205H		Not Used
206H	Reset	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
207H		Not Used
		*********	O	U	T	P	U	T	P	O	R	T	S	**********
208H	Motor 1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	Phi 4	Phi 3	Phi 2	Phi 1
209H	Motor 2	X	X	X	X	X	X	X	X	X	X	X	X	X	X	Phi 4	Phi 3	Phi 2	Phi 1
20AH	Motor 3	X	X	X	X	X	X	X	X	X	X	X	X	X	X	Phi 4	Phi 3	Phi 2	Phi 1
20BH	8 Bit D/A	7	6	5	4	3	2	1	0
20CH	10 Bit D/A	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	9	8
20DH	10 Bit D/A	7	6	5	4	3	2	1	0
20EH		Not Used
20FH		Not Used
		**********	T	I	M	E	R	**********
210H	Timer 0	7	6	5	4	3	2	1	0
211H	Timer 1	7	6	5	4	3	2	1	0
212H	Timer 2	7	6	5	4	3	2	1	0
213H	Control	7	6	5	4	3	2	1	0

Figure 11. Input / Output Memory Map

E. Miscellaneous

Figure 12 shows circuit diagrams used to decode the decimal thumbwheel selector. Also Shown in Figure 12 are diagrams for interfacing the rocker switches. Two 8 bit ports are used to interface the force sensors and other components housed in the teach pendent box.

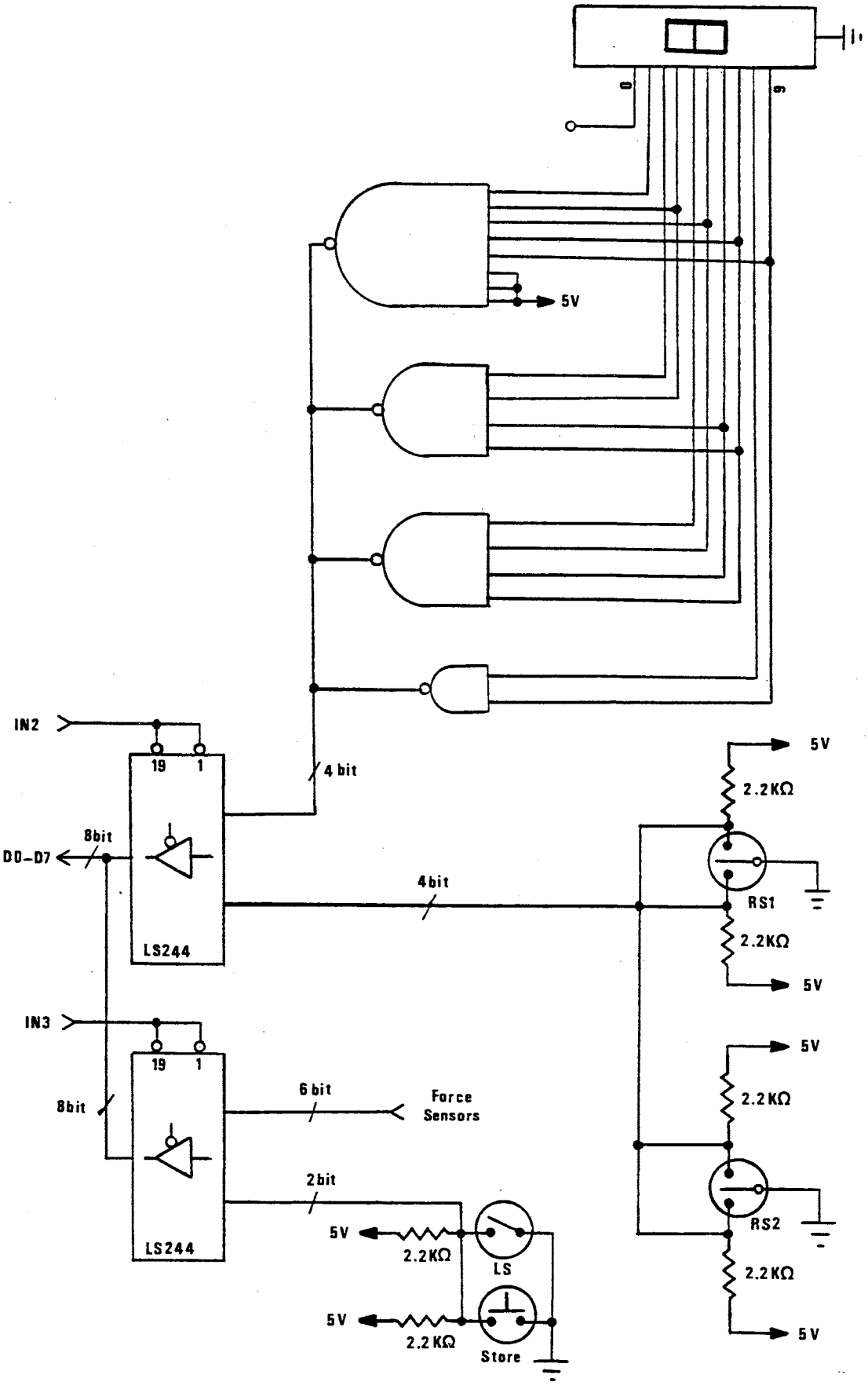


Figure 12. Teach Pendent Interface Schematic

CHAPTER VIII

SUMMARY AND CONCLUSIONS

To understand the dynamics, kinematics, and control of closed loop robots, building a test-bed for the Oklahoma State Crawdad was taken as the main part of the thesis objective. Development of a powerful and efficient operating system completed the primary goals of this project. Such a test-bed was built to provide grounds for future research and study of this class of robots.

The Crawdad robot arm is basically a three degree of freedom planar mechanism with sliding joint added to the output link to allow manipulation of a small object in space. The output link of the planar mechanism is capable of two translational motion components and a rotational motion about an axis, vertical to the plane of the mechanism. However, the rotational component is highly constrained by the position of the output link such that at extreme positions in the workspace, no independent rotation may be achieved due to the nature of the mechanism. Thus, the dexterity of the robot arm may be appreciably improved by integration of a "roll" in the wrist. Furthermore, "pitch" and "yaw" components may be considered for higher flexibility.

Control of the stepper motors to drive the three input links of the Crawdad was the next item to be considered, once the mechanical configuration of the arm was decided upon. Speed and efficiency were the primary criteria for the stepper motor controllers. High speeds of operation were achieved using high current, low voltage stepper motors and appropriate circuitry. Linear tracking power control was implemented to avoid excessive power dissipation in stepper motor windings and the series resistors. However, some undesirable overheating occurred in the power MOSFETs which was the side-effect of this method. Incorporation of chopper amplifiers is recommended to overcome this problem.

The actuators to drive the Z-axis prismatic joint and the gripper had to be mounted on the moving output link of the mechanism. Therefore, weight of these actuators was the primary criterion to be considered. Servo motors were selected for this purpose due to their superior power to weight ratio.

Conventional PID (Proportional/Integral/Differential) control strategy was applied to the Z-axis servo motor, while proportional control alone was decided to be adequate for the gripper servo. The servo actuators operated properly without producing any major complexities.

Software development strategies proved to be one of the more challenging problems in this project. Development of control modules of software in high level languages such as C was found impossible due to inadequate execution speed

of a microcomputer. Therefore, Assembly language was used where high speeds of execution were necessary. Pascal language was selected to structure the software and handle disk access and user interface.

Due to the segmented structure of the TI-Professional microcomputer, complexities arose when trying to link the servo interrupt service routine to other software modules. However, advanced software development tools would considerably simplify this task.

One of the most important features of a robot manipulator is the task-programming technique. Typical point-path robots are taught a task by moving each joint independently to a desired position and saving the vector of joint positions to describe a particular point in the path. Basically, this method has been used to program the Crawdad arm. All joint velocities are brought to rest at each point programmed by the operator. The teach process is different from the general point path robots in one aspect. Complex equations of motion must be carried out to compute the joint angles, given an end effector position of a closed loop robot. Thus, the conventional open loop robot programming techniques are tedious and impractical to be implemented on a closed loop manipulator. The teach mode of operation of the Crawdad arm is similar to that of some continuous-path robots available in market (painting robots). The user grabs the end effector and maneuvers it to a desired position. Strain gages are mounted on the input links to sense the direction of the force applied by

the operator. The base motors are then driven such that the arm responds by moving the output link in the direction of the applied force. The Z-axis and the gripper are controlled conventionally with various speed selections for coarse or fine positioning of the end effector.

Once a desired position and orientation of the end effector has been achieved, all joint positions are stored. When the same task is played back in automatic mode the joint motions are linearized with respect to each other using a special algorithm. Thus, the path from any point to the next is generated by this technique, independent of the intermediate arm positions manipulated by the operator. This ensures smooth motion of the arm between points. The operator may set the speed of the robot and acceleration of the input joints in automatic mode, corresponding to different load conditions.

The only problems encountered in the teach mode were related to the use of commercial grade electronic components in force sensor interface. Temperature instability of the resistors and the potentiometers used in the Wheatstone bridge required adjustments repeatedly. The use of quality products with low coefficient of temperature is highly recommended.

Possibilities for future research in development areas are integration of a vision system or incorporation of tactile sensors on the gripping surfaces. Problems such as obstacle avoidance may be studied in detail using a vision system. Also, integration of pattern recognition with the

robot arm is another challenging field for further research.

A SELECTED BIBLIOGRAPHY

1. Sumpter, B., "Simulation Algorithm of Oklahoma Crawdad Robot," Applied Mechanisms Conference, Kansas City, 1985.

2. "Stepper Motors and Controls," Bodine Electric Company, Catalog ST-1, 1981.

3. McAllister, K., "D.C. Servo Motor Control Using the ICH8510," Intersil Application Bulletin A026.

4. Paul, R. P., "Robot Manipulators," The MIT Press, 1981.

5. "Robotics Reference and Applications Manual," Microbot Inc., 1981.

6. "Motorola Power Data Book," Motorola Inc., 1982.

7. "Power Op Amps Data Book," Apex Microtechnology Corp., 1985.

8. DALLY, J. W. and RILEY, W. F., "Experimental Stress Analysis," McGraw-Hill Book Company, 1978.

9. "Professional Computer Technical Reference Manual," Texas Instruments, 1983.

10. "Component Data Catalog," Intel Corporation, 1982.

APPENDIXES

APPENDIX A

PASCAL MAIN PROGRAM LISTING

APPENDIX A

PASCAL MAIN PROGRAM LISTING

```
PROGRAM ARM(INPUT,OUTPUT);
TYPE
   LIST= ARRAY[1..5] OF INTEGER;
CONST
   SIZE=100;
VAR
   FILENAME:LSTRING(30);
   INCDEC:INTEGER;
   F: FILE OF CHAR;
   TABLE: ARRAY [1..5,0..SIZE] OF INTEGER;
   DEL:LIST;
   TEMP: LIST;
   ENDTAB:WORD;
   OPTION:INTEGER;C,C1,KEY,CHC:CHAR;
   CURRENT,NEWPOINT,MAX,I,J:INTEGER;
   SLOPE,AMAX,VMAX:REAL;
   ATABLE:WORD;
   POINT:INTEGER;
VALUE
   AMAX:=50.0;SLOPE:=150.0;VMAX:=5.0;
   ATABLE:=#3000;
(*.............. PROCEDURE DECLARATIONS HERE .............*)

PROCEDURE RSET1(A:WORD);EXTERNAL;
PROCEDURE RSET2;EXTERNAL;
PROCEDURE DRIVE(D1,D2,D3,M1,M2:INTEGER);EXTERNAL;
PROCEDURE TEACH(VAR MT1,MT2,MT3,SRV1,SRV2:INTEGER);EXTERNAL;

PROCEDURE MANUAL;
BEGIN
   TEACH(DEL[1],DEL[2],DEL[3],DEL[4],DEL[5]);
   FOR I:=1 TO 5 DO
     [TEMP[I]:=TEMP[I]+DEL[I];
   CURRENT:=CURRENT+1;
   FOR J:=CURRENT TO MAX DO
     FOR I:=1 TO 5 DO
       TABLE[I,J+1]:=TABLE[I,J];
   FOR I:=1 TO 5 DO
     TABLE[I,CURRENT]:=TEMP[I];
   MAX:=MAX+1;
END;
```

```
PROCEDURE DRV;
BEGIN
    DRIVE(DEL[1],DEL[2],DEL[3],DEL[4],DEL[5]);
END;

FUNCTION GET_CHR: CHAR;
BEGIN
   REPEAT
      GET(F)
   UNTIL F^ <> CHR(0);
   GET_CHR:=F^
END;

PROCEDURE GO_SEQ;
BEGIN
   INCDEC:=1;
   IF CURRENT = NEWPOINT THEN RETURN;
   IF CURRENT > NEWPOINT THEN INCDEC:=-1;
    FOR J:=CURRENT TO NEWPOINT DO
       BEGIN
          FOR I:=1 TO 5 DO
             DEL[I]:=TABLE[I,J+INCDEC]-TABLE[I,J];
          DRV
       END;
    CURRENT:=NEWPOINT;
    FOR I:=1 TO 5 DO
       [DEL[I]:=0; TEMP[I]:=TABLE[I,CURRENT]]
END;

PROCEDURE GO_LIN;
BEGIN
    FOR I:=1 TO 5 DO
       DEL[I]:=TABLE[I,NEWPOINT]-TABLE[I,CURRENT];
    DRV;
    POINT:=CURRENT;
    CURRENT:=NEWPOINT;
    FOR I:=1 TO 5 DO
       [DEL[I]:=0; TEMP[I]:=TABLE[I,CURRENT]]
END;

PROCEDURE REP_SEQ;
BEGIN
   REPEAT
      WHILE CURRENT < MAX DO
        BEGIN
          FOR I:=1 TO 5 DO
            DEL[I]:=TABLE[I,CURRENT+1]-TABLE[I,CURRENT];
            DRV;
          CURRENT:=CURRENT+1;
          GET(F);
          KEY:=F^;
          IF KEY=' ' THEN
```

```
            WRITE('E TO EXIT, ANY OTHER KEY TO CONTINUE');
           READLN(KEY);
           IF KEY='E' THEN BREAK
         END;
       FOR I:=1 TO 5 DO
         DEL[I]:=TABLE[I,1]-TABLE[I,MAX];
       DRV;
       CURRENT:=1;
   UNTIL FALSE;
 FOR I:=1 TO 5 DO
       [DEL[I]:=0; TEMP[I]:=TABLE[I,CURRENT]]
END;

PROCEDURE DEL_PT;
BEGIN
    WRITE('ENTER THE POINT TO BE DELETED: ');
    READLN(POINT);
    IF (POINT > MAX) OR (POINT<= 0) THEN
       [WRITELN('THE POINT DOES NOT EXIST');RETURN;];
    MAX:=MAX-1;
    IF POINT=CURRENT THEN
       [WRITELN('CANNOT DELETE CURRENT POINT! '); RETURN ];
    FOR J:=POINT TO MAX DO
        FOR I:=1 TO 5 DO
           TABLE[I,J]:=TABLE[I,J+1];
    IF POINT < CURRENT THEN
       [CURRENT:=CURRENT-1;
        FOR I:=1 TO 5 DO
            TEMP[I]:=TABLE[I,CURRENT]
       ]
END;

PROCEDURE LOAD;
VAR
   IN_FILE: FILE OF INTEGER;
BEGIN
   MAX:= 0;
   READLN;
   WRITE('ENTER THE FILENAME: ');
   READLN(FILENAME);
   ASSIGN(IN_FILE,FILENAME);
   RESET(IN_FILE);
   WHILE (MAX <SIZE) AND NOT EOF(IN_FILE) DO
      BEGIN
         MAX:=MAX +1;
         FOR I:=1 TO 5 DO
           [TABLE[I,MAX]:= IN_FILE^;
            GET(IN_FILE)]
      END
END;

PROCEDURE SAVE;
VAR
```

```
   OUT_FILE: FILE OF INTEGER;
   C: CHAR;
BEGIN
   WRITELN('CURRENT FILE NAME IS: ',FILENAME);
   READLN;
   WRITE('ENTER NEW FILE NAME: ');
   IF NOT EOLN THEN
       READLN(FILENAME);
       ASSIGN(OUT_FILE,FILENAME);
       REWRITE(OUT_FILE);
       FOR J:=1 TO MAX DO
          BEGIN
             FOR I:=1 TO 5 DO
                [OUT_FILE^:= TABLE[I,J];
                 PUT(OUT_FILE)]
          END
END;

PROCEDURE MENU;
BEGIN
   WRITELN;WRITELN;
   WRITELN('.............. MAIN MENU ..............');
   WRITELN;
   WRITELN(' 0. GENERATE NEW ACC./DECC. TABLE ' );
   WRITELN(' 1. MANUAL / TEACH ');
   WRITELN(' 2. GO TO A POINT / SEQUENTIALY ');
   WRITELN(' 3. GO TO A POINT / LINEARLY ' );
   WRITELN(' 4. REPEAT A SEQUENCE ');
   WRITELN(' 5. DELETE A POINT ');
   WRITELN(' 6. LOAD A FILE FROM DISK ');
   WRITELN(' 7. SAVE CURRENT FILE ');
   WRITELN(' 8. EXIT ');
   WRITELN;
   WRITE(' ENTER OPTION NUMBER : ');
   READLN(OPTION);
END;

PROCEDURE CALCULATE;FORWARD;

PROCEDURE USER;
BEGIN
  RSET1(0);
  WRITE('DO YOU WANT TO MODIFY MOTOR PARAMETERS? Y/N : ');
  READLN(CHC);
  IF CHC<>'Y' THEN [CALCULATE;RETURN];
  WRITELN;
  WRITELN('MAXIMUM VELOCITY IS: ',VMAX:-4:1,'KSTEPS/SEC');
  WRITE('ENTER THE NEW VALUE: ');
  READLN(VMAX);
  WRITELN('MAXIMUM ACCEL. IS: ',AMAX:-4:1,'KSTEPS/SEC^2');
  WRITE('ENTER THE NEW VALUE: ');
  READLN(AMAX);
  WRITELN('SLOPE  IS: ',SLOPE:-4:1,'KSTEPS/SEC^3');
```

```
  WRITE('ENTER THE NEW VALUE: ');
  READLN(SLOPE);
  CALCULATE
END;

PROCEDURE CALCULATE;
VAR
   V,A:REAL;SS:INTEGER;
   DEL:ADS OF INTEGER;
   R,S,OFFSET:WORD;
BEGIN
   WRITELN;
   WRITELN('PLEASE WAIT ...');
   SS:=1;
   OFFSET:=0;
   DEL.R:=OFFSET;
   DEL.S:=ATABLE;
   DEL^:=MAXINT;
   WHILE ( (A<AMAX) AND (V<VMAX) ) DO
      BEGIN
         V:=EXP( 1/3*LN(4E-6*SLOPE*SQR(SS)) );
         A:=SQRT(SLOPE*V);
         OFFSET:=OFFSET+2;
         DEL.R:=OFFSET;
         DEL^:=ROUND(625/V);
         SS:=SS+1;
      END;
   WHILE V<VMAX DO
      BEGIN
         V:=SQRT(2E-3*SS*AMAX);
         OFFSET:=OFFSET+2;
         DEL.R:=OFFSET;
         DEL^:=ROUND(625/V);
         SS:=SS+1
      END;
   ENDTAB:=WRD(SS*2);
   RSET1(ENDTAB)
END;

(*.......................... MAIN ..........................*)

BEGIN
    RSET1(0);
    WRITELN('TURN ON THE ROBOT POWER AND PRESS RETURN');
       WHILE NOT EOLN DO
          RETURN;
    READLN;
    RSET2;
    USER;
    MAX:=0;
    CURRENT:=0;
    ASSIGN(F,'USER');
    RESET(F);
```

```
    C:='N';
       REPEAT
         MENU;
           CASE OPTION OF
             0: USER;
             1: MANUAL;
             2: [WRITELN('ENTER THE POINT: ');READ(NEWPOINT);
                 GO_SEQ];
             3: [WRITELN('ENTER THE POINT: ');READ(NEWPOINT);
                 GO_LIN];
             4: REP_SEQ;
             5: DEL_PT;
             6: LOAD;
             7: SAVE;
             8: [WRITE('QUIT PROGRAM ? ');READLN(C);]
             OTHERWISE
               WRITELN('TRY AGAIN ');
           END;
       UNTIL C='Y'
END.
```

APPENDIX B

ASSEMBLY LANGUAGE PROCEDURES LISTING

APPENDIX B

ASSEMBLY LANGUAGE PROGRAM LISTING

```
PORT1     EQU     0208H              ;MOTOR1
PORT2     EQU     0209H              ;MOTOR2
PORT3     EQU     020AH              ;MOTOR3
FORCE     EQU     0203H              ;FORCE SENSORS /STORE SWITCH

PANEL     EQU     0202H              ;SPEED AND ROCKER SWITCHES

DAC1      EQU     020CH              ;10 BIT D/A
DAC2      EQU     020BH              ;8 BIT D/A
NCODER    EQU     0200H              ;SHAFT-ENCODER
ADC       EQU     0204H              ;8 BIT A/D
ZERO      EQU     0206H              ;CLEAR SHAFT ENCODER /RESET A/D
IVECT1    EQU     40H                ;VECTOR FOR INTERRUPT 0
LVECT1    EQU     IVECT1*4
HVECT1    EQU     LVECT1+2
IVECT2    EQU     41H                ;VECTOR FOR INTERRUPT 1
LVECT2    EQU     IVECT2*4
HVECT2    EQU     LVECT2+2
TIMER0    EQU     210H
TIMER1    EQU     211H
TCNTRL    EQU     213H
SCAN      EQU     5000D              ;SCAN SERVOS EVERY 5 ms

TABLE     SEGMENT AT 3000H           ;ACC / DEC TABLE
          DW 8000H DUP(?)            ;WILL BE STORED HERE
TABLE     ENDS

DATA1     SEGMENT
          TAB     DB 9,3,1,2,        ;HALF STEPPING PHASE TABLE
                  DB 6,4,0CH,8
          DIR     DW 3 DUP(1)        ;CW IS THE DEFAULT DIRECTION
          PNTR    DW 3 DUP(2)        ;PHASE TABLE POINTER IS 2
          SUM     DW 3 DUP(?)        ;DUMMY VARIABLE
          MAX     DW ?               ;MAXIMUM # OF STEPS
          ACD     DW ?               ;2 MEANS ACC. /-2 MEANS DEC.
          TEMP    DW ?               ;DUMMY VARIABLE
          HAFMAX  DW ?               ;MAX STEPS DIVIDED BY TWO
          GSWICH  DB ?               ;GRIPPER FORCE SWITCH FLAG
          SSWICH  DB ?                ;SERVO PANEL SWITCHES
          DELAY   DW 40000,20000,    ;DELAYS BETWEEN EACH STEP
                  DW 13333,10000     ;FOR TEACH MODE
                  DW 8000,6667,5714
                  DW 5000,4444,4000
          DELTA   DW 5,8,11,13,16    ;DUMMY TABLE FOR SERVO
```

```
                DW 19,22,25,31,35 ;SPEED
DSIRD1  DW ?                    ;DESIRED POSITION FOR Z-AXIS
DSIRD2  DB ?                    ;DESIRED POSITION FOR GRIPPER
POSTN1  DW ?                    ;PRESENT POSITION OF Z-AXIS
POSTN2  DB ?                    ;PRESENT POSITION OF GRIPPER
ENDTAB  DW ?
MOT     DW 3 DUP(?)             ;POSITION OF STEPPER MOTORS
INTG    DW 8 DUP(0)             ;ARRAY USED FOR INTEGRATION
CNTR    DW 4                    ;POINTER USED FOR INTEGRATOR
OLDE    DB 0                    ;PREVIOUS ERROR OF GRIPPER
DATA1   ENDS

        PUBLIC TEACH
        PUBLIC DRIVE
        PUBLIC RSET1
        PUBLIC RSET2

COD1    SEGMENT
        ASSUME CS:COD1,DS:DATA1
TEACH   PROC   FAR
        PUSH   BP               ;SAVE PASCAL BP
        PUSH   DS               ;PASCAL DS IS ON TOP OF STACK
        MOV    BP,SP            ;POINT TO PASCAL FRAME
        MOV    AX,DATA1
        MOV    DS,AX
        CLI                     ;DISABLE INTERRUPTS
        MOV    DX,TCNTRL        ;MODE 0 FOR TIMER 0
        MOV    AL,30H
        OUT    DX,AL
        SUB    AX,AX            ;AX <--- 0
        MOV    ES,AX            ;ES <--- 0
        MOV    AX,OFFSET ISRTN1 ;PATCH IN NEW INTERRUPT VECTOR
        MOV    ES:[LVECT1],AX
        MOV    AX,COD1
        MOV    ES:[HVECT1],AX
        MOV    DX,TIMER0        ;SET UP FOR INTERRUPT 1
        MOV    AL,0FFH
        OUT    DX,AL            ;TIMER0 <--- 0FFFFH
        OUT    DX,AL
LOOP:   MOV    DX,PANEL
        IN     AL,DX            ;READ PANEL
        MOV    CL,AL            ;SAVE IN CL
        AND    AX,0FH           ;SPEED IN AX (MASK OFF REST)
        MOV    SI,AX
        SHL    SI,1             ;SPEED*2 IN SI
        MOV    BX,[DELTA+SI]    ;LOOK UP PROPER PARAMETER
        MOV    DX,ADC
        IN     AL,DX            ;READ POSITION OF GRIPPER
        SAL    CL,1             ;CHECK GRIPER OPEN SWITCH
```

```
        JC      LL1
        SUB     AL,BL           ;START OPENING THE GRIPPER
        MOV     [DSIRD2],AL
LL1:    SAL     CL,1            ;CHECK GRIPPER CLOSE SWITCH
        JC      LL2
        ADD     AL,BL           ;START CLOSING THE GRIPPER
        SUB     AH,AH
        CMP     AX,0A6H         ;DON'T LET THE GRIPPER
        JLE     FF2             ;OPEN TOO MUCH
        MOV     AL,0A6H         ;MAXIMUM ALLOWABLE POSITION

        SUB     AH,AH           ;OF GRIPPER
FF2:    MOV     [DSIRD2],AL
LL2:    SHL     BX,1            ;MULTIPY THE PARAMETER BY 8
        SHL     BX,1
        SHL     BX,1
        MOV     DX,NCODER       ;READ THE POSITION OF Z-AXIS
        IN      AX,DX
        SAL     CL,1            ;CHECK Z-AXIS DOWN SWITCH
        JC      LL3
        SUB     AX,BX           ;REDUCE THE DESIRED POSITION
        MOV     [DSIRD1],AX
LL3:    SAL     CL,1            ;CHECK Z-AXIS UP SWITCH
        JC      LL4
        ADD     AX,BX           ;INCREASE THE DESIRED POSITION
        MOV     [DSIRD1],AX
LL4:    MOV     DX,FORCE        ;READ FORCE
        IN      AL,DX
        SHR     AL,1            ;CHECK THE STORE PUSH BUTTON
        JC      DOWNA           ;RETURN TO PASCAL IF PRESSED
        MOV     DX,TCNTRL       ;MODE 0 FOR TIMER 0
        MOV     AL,30H          ;(DISABLE TIMER0)
        OUT     DX,AL
        POP     AX              ;PASCAL DS IN AX
        PUSH    AX
        MOV     ES,AX           ;PASCAL DS IN ES
        MOV     BX,[BP+8]       ;ADDRESS OF DSIRED2 IN BX
        MOV     AL,[DSIRD2]     ;AL <--- DSIRED2
        SUB     AH,AH           ;AH <--- 0
        MOV     ES:[BX],AX      ;DEL[5]=DSIRD2
        MOV     BX,[BP+10]
        MOV     AX,[DSIRD1]
        MOV     ES:[BX],AX      ;DEL[4]=DSIRD1
        MOV     BX,[BP+12]
        MOV     AX,[MOT+4]
        MOV     ES:[BX],AX      ;DEL[3]=MOT3
        MOV     BX,[BP+14]
        MOV     AX,[MOT+2]
        MOV     ES:[BX],AX      ;DEL[2]=MOT2
        MOV     BX,[BP+16]
        MOV     AX,[MOT]
        MOV     ES:[BX],AX      ;DEL[1]=MOT1
        STI
```

```
        POP     DS
        POP     BP              ;RETURN TO PASCAL
        RET     10
DOWNA:  SHR     AL,1            ;GSWITCH IN CARRY (CONTINUE)
        JC      OFF
        MOV     GSWICH,0        ;GSWICH=0 MEANS ON
        JMP     ON
OFF:    MOV     GSWICH,0FFH     ;GSWICH=FF MEANS OFF
ON:     STI                     ;ALLOW INTERRUPTS HERE
        STI
        CLI
        JMP     LOOP            ;CONTINUE

ISRTN1  PROC    NEAR            ;THIS ISR SCANS FORCE SENSORS
        PUSH    AX              ;AND STEPS THE MOTORS
        PUSH    CX              ;AS NECESSARY
        PUSH    DX
        PUSH    DI
        MOV     AX,[DELAY+SI]   ;LOOK UP APROPRIATE DELAY
        MOV     DX,TIMER0
        OUT     DX,AL           ;LOAD THE TIMER WITH DELAY
        MOV     AL,AH
        OUT     DX,AL
        MOV     DX,FORCE        ;READ THE FORCE SENSORS
        IN      AL,DX
        SHR     AL,1            ;STORE BIT IS SHIFTED OUT
        SHR     AL,1            ;GRIPPER BIT IS SHIFTED OUT
        MOV     CL,AL           ;CL <--- FORCE SENSORS
F1CW:   SHR     CL,1            ;FORCE-1-CW IN CARRY
        JC      F1CCW
        MOV     DI,[DIR]
        CMP     DI,1            ;DIR=CW ?
        JNE     DOWN1
        INC     [MOT]
        MOV     DI,[PNTR]       ;USE DI FOR PHASE POINTER
        INC     DI
        AND     DI,07           ;INCREMENT DI IN MODULE 8
        MOV     [PNTR],DI
        MOV     AL,[TAB+DI]     ;LOOK UP THE PHASE VALUE
        MOV     DX,PORT1        ;FROM TABLE
        OUT     DX,AL           ;STEP THE MOTOR
        JMP     F2CW
DOWN1:  MOV     [DIR],1
        JMP     F2CW
F1CCW:  SHR     CL,1            ;F1CCW IN CARRY
        JC      F2CW
        MOV     DI,[DIR]
        CMP     DI,-1           ;DIR=CCW ?
        JNE     DOWN2
        DEC     [MOT]
        MOV     DI,[PNTR]
        DEC     DI
```

```
        AND     DI,07
        MOV     [PNTR],DI
        MOV     AL,[TAB+DI]
        MOV     DX,PORT1
        OUT     DX,AL
        JMP     F2CW
DOWN2:  MOV     [DIR],-1
F2CW:   SHR     CL,1                    ;F2CW IN CARRY
        JC      F2CCW
        MOV     DI,[DIR+2]
        CMP     DI,1                    ;DIR=CW ?
        JNE     DOWN3
        INC     [MOT+2]
        MOV     DI,[PNTR+2]
        INC     DI
        AND     DI,07
        MOV     [PNTR+2],DI
        MOV     AL,[TAB+DI]
        MOV     DX,PORT2
        OUT     DX,AL
        JMP     F3CW
DOWN3:  MOV     [DIR+2],1
        JMP     F3CW
F2CCW:  SHR     CL,1                    ;F2CCW IN CARRY
        JC      F3CW
        MOV     DI,[DIR+2]
        CMP     DI,-1                   ;DIR=CW ?
        JNE     DOWN4
        DEC     [MOT+2]
        MOV     DI,[PNTR+2]
        DEC     DI
        AND     DI,07
        MOV     [PNTR+2],DI
        MOV     AL,[TAB+DI]
        MOV     DX,PORT2
        OUT     DX,AL
        JMP     F3CW
DOWN4:  MOV     [DIR+2],-1
F3CW:   SHR     CL,1                    ;F3CW IN CARRY
        JC      F3CCW
        MOV     DI,[DIR+4]
        CMP     DI,1                    ;DIR=CW ?
        JNE     DOWN5
        INC     [MOT+4]
        MOV     DI,[PNTR+4]
        INC     DI
        AND     DI,07
        MOV     [PNTR+4],DI
        MOV     AL,[TAB+DI]
        MOV     DX,PORT3
        OUT     DX,AL
        JMP     OUTC
```

```
DOWN5:  MOV     [DIR+4],1
        JMP     OUTC
F3CCW:  SHR     CL,1                    ;F3CCW IN CARRY
        JC      OUTC
        MOV     DI,[DIR+4]
        CMP     DI,-1                   ;DIR=CCW ?
        JNE     DOWN6
        DEC     [MOT+4]
        MOV     DI,[PNTR+4]
        DEC     DI
        AND     DI,07
        MOV     [PNTR+4],DI
        MOV     AL,[TAB+DI]
        MOV     DX,PORT3
        OUT     DX,AL
        JMP     OUTC
DOWN6:  MOV     [DIR+4],-1
OUTC:   POP     DI                      ;RESTORE THE REGISTERS
        POP     DX
        POP     CX
        MOV     AL,60H                  ;END OF INTERRUPT COMMAND
        OUT     18H,AL
        POP     AX
        IRET
ISRTN1  ENDP
TEACH   ENDP


DRIVE   PROC    FAR
        PUSH    DS
        PUSH    BP
        MOV     AX,DATA1
        MOV     DS,AX
        MOV     BP,SP
        MOV     AX,[BP+16]      ;GET THE ARGUMENTS FROM PASCAL
        MOV     [MOT],AX
        MOV     AX,[BP+14]
        MOV     [MOT+2],AX
        MOV     AX,[BP+12]
        MOV     [MOT+4],AX
        MOV     AX,[BP+10]
        MOV     [DSIRD1],AX
        MOV     AX,[BP+8]
        MOV     [DSIRD2],AL
        CLI                             ;NO INTERRUPTS
        MOV     AL,30H                  ;MODE 0 FOR TIMER 0
        MOV     DX,TCNTRL
        OUT     DX,AL
        SUB     AX,AX
        MOV     ES,AX
        MOV     AX,OFFSET ISRTN2 ;NEW INTERUPT VECTOR
        MOV     ES:[LVECT1],AX
```

```
        MOV     AX,COD1
        MOV     ES:[HVECT1],AX
        SUB     AX,AX           ;MAX <--- 0
        MOV     MAX,AX
        INC     AX              ;DIR(I) <--- 1
        MOV     [DIR],AX
        MOV     [DIR+2],AX
        MOV     [DIR+4],AX
        MOV     ACD,2           ;ACD=2
ONE:    MOV     AX,[MOT]
        OR      AX,AX           ;SET FLAGS
        JGE     POS1
        NEG     AX              ;GET THE ABS OF THE ARGUMENT
        NEG     [DIR]           ;DIR <--- CCW
        MOV     [MOT],AX
POS1:   CMP     AX,MAX          ;FIND THE MAX
        JLE     TWO
        MOV     MAX,AX
TWO:    MOV     AX,[MOT+2]
        OR      AX,AX
        JGE     POS2
        NEG     AX
        NEG     [DIR+2]
        MOV     [MOT+2],AX
POS2:   CMP     AX,MAX
        JLE     THREE
        MOV     MAX,AX
THREE:  MOV     AX,[MOT+4]
        OR      AX,AX
        JGE     POS3
        NEG     AX
        NEG     [DIR+4]
        MOV     [MOT+4],AX
POS3:   CMP     AX,MAX
        JLE     OUT
        MOV     MAX,AX          ;SET UP FOR THE ALGORITHM
OUT:    MOV     CX,MAX          ;COUNTER(CX)=MAX
        MOV     BX,CX           ;KEEP MAX IN BX
        MOV     AX,BX
        SHR     AX,1            ;AX=MAX/2
        MOV     HAFMAX,AX
        MOV     [SUM],AX        ;SUM[I]=MAX/2
        MOV     [SUM+2],AX
        MOV     [SUM+4],AX
        MOV     SI,0            ;SET UP FOR ACC.
        MOV     AL,0FFH         ;SET UP FOR FIRST INT.
        MOV     DX,TIMER0
        OUT     DX,AL
        OUT     DX,AL
LOOP1:  STI                     ;ENABLE INTERRUPT
        CMP     SI,ENDTAB       ;MAXIMUM VELOCITY YET ?
        JGE     OUTA
```

```
            CMP     CX,HAFMAX       ;HALF DISTANCE YET ?
            JG      LOOP1
            JMP     OUTB
OUTA:       MOV     ACD,0           ;ACD=0 (CONST. VEL.)
            MOV     TEMP,BX
            SUB     TEMP,CX         ;TEMP=MAX-COUNTER
;TEMP KEEPS THE NUMBER OF STEPS NECCESSARY TO DECC. TO 0 !!
LOOP2:      STI
            CMP     CX,TEMP         ;TIME TO DECC. YET ?
            JG      LOOP2
OUTB:       MOV     ACD,-2          ;START DECC.
LOOP3:      STI
            CMP     CX,0            ;DONE ?
            JG      LOOP3
            MOV     AL,34H          ;ENABLE IR0,IR1&IR3
            OUT     19H,AL
            MOV     DX,TCNTRL
            MOV     AL,30H
            OUT     DX,AL           ;MODE 0 (DISABLE TIMER0)
            POP     BP
            POP     DS
            RET     10              ;RETURN TO PASCAL

ISRTN2      PROC    NEAR
            PUSH    AX
            PUSH    DI
            PUSH    DX
            ADD     SI,ACD
            MOV     AX,TABLE
            MOV     ES,AX
            MOV     AX,ES:[SI]
            MOV     DX,TIMER0
            OUT     DX,AL           ;LOOK UP DELAY VALUE
            MOV     AL,AH           ;AND LOAD TIMER1
            OUT     DX,AL
            MOV     AX,[MOT]        ;START THE ALGORITHM
            SUB     [SUM],AX        ;SUM=SUM-MOT
            JGE     DELAY1
            ADD     [SUM],BX        ;SUM=SUM+MAX
            MOV     DI,[PNTR]       ;STEP THE MOTOR
            ADD     DI,[DIR]
            AND     DI,07H
            MOV     [PNTR],DI
            MOV     AL,[TAB+DI]
            MOV     DX,PORT1
            OUT     DX,AL
NEXT1:      MOV     AX,[MOT+2]
            SUB     [SUM+2],AX      ;SUM=SUM-MOT
            JGE     DELAY2
            ADD     [SUM+2],BX      ;SUM=SUM+MAX
            MOV     DI,[PNTR+2]
            ADD     DI,[DIR+2]
```

```
        AND     DI,07H
        MOV     [PNTR+2],DI
        MOV     AL,[TAB+DI]
        MOV     DX,PORT2
        OUT     DX,AL
NEXT2:  MOV     AX,[MOT+4]
        SUB     [SUM+4],AX      ;SUM=SUM-MOT
        JGE     DELAY3
        ADD     [SUM+4],BX      ;SUM=SUM+MAX
        MOV     DI,[PNTR+4]
        ADD     DI,[DIR+4]
        AND     DI,07H
        MOV     [PNTR+4],DI
        MOV     AL,[TAB+DI]
        MOV     DX,PORT3
        OUT     DX,AL
NEXT3:  DEC     CX
        POP     DX
        POP     DI
        MOV     AL,60H          ;END OF INTERRUPT
        OUT     18H,AL
        POP     AX
        IRET
ISRTN2  ENDP

DELAY1: MOV     DI,04           ;DELAY=18X+2Y+19
UP1:    DEC     DI              ;WHERE:
        JNZ     UP1             ;X=CONTENTS OF DI
        NOP                     ;Y= # OF NOPS
        NOP                     ;X=4  & Y=4 -->
        NOP                     ;DELAY=99 CLOCK
        NOP                     ;CYCLES
        JMP     NEXT1
DELAY2: MOV     DI,04           ;DELAY1 & DELAY2
UP2:    DEC     DI              ;AND DELAY3 ARE
        JNZ     UP2             ;IDENTICAL
        NOP
        NOP
        NOP
        NOP
        JMP     NEXT2
DELAY3: MOV     DI,04
UP3:    DEC     DI
        JNZ     UP3
        NOP
        NOP
        NOP
        NOP
        JMP     NEXT3
DRIVE   ENDP
```

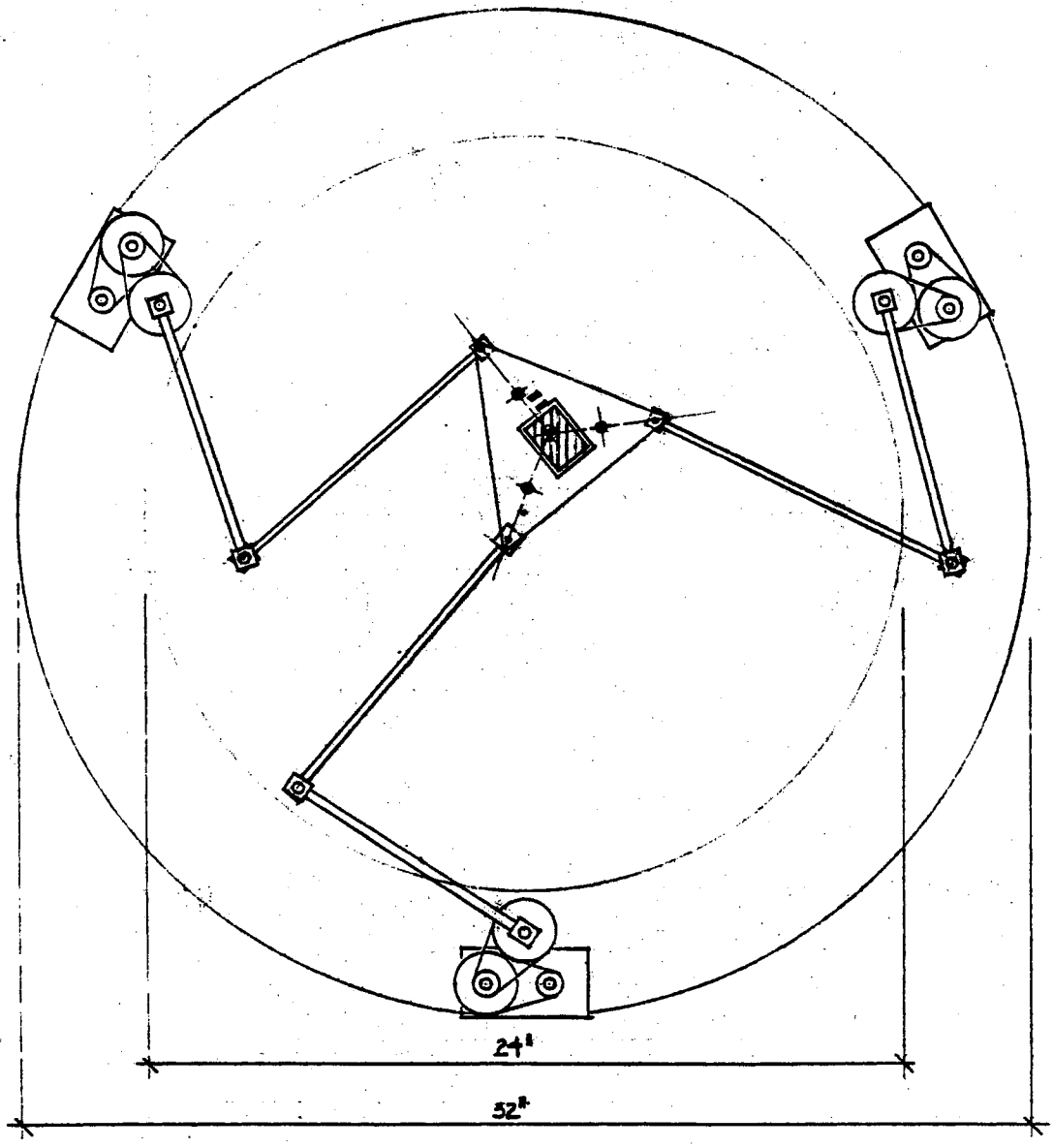
```
RSET1   PROC    FAR             ;THIS PROCEDURE CLEARS D/A's
        PUSH    DS              ;STARTS THE SERVO SCANNING AND
        PUSH    BP              ;RESETS THE TIMERS
                                ;IT ALSO ACCEPTS THE LENGTH OF
        CLI                     ;ACC /DEC TABLE FROM  PASCAL
        SUB     AX,AX           ;PATCH THE ISR VECTOR
        MOV     ES,AX
        MOV     AX,SEG SERVO
        MOV     ES:[HVECT1],AX
        MOV     AX,OFFSET SERVO
        MOV     ES:[LVECT1],AX

        MOV     DX,TCNTRL
        MOV     AL,30H          ;MODE0 FOR TIMER0
        OUT     DX,AL
        MOV     AL,70H          ;MODE0 FOR TIMER1
        OUT     DX,AL

        MOV     AL,34H          ;ENABLE IR0 & IR1
        OUT     19H,AL

        MOV     AX,DATA1
        MOV     DS,AX
        MOV     BP,SP
        MOV     AX,[BP+8]       ;GET THE ARGUEMENT FROM PASCAL
        MOV     ENDTAB,AX       ;STORE THE ARGUEMENT
        MOV     AX,1E6H         ;SEND 0 VOLTS TO  AMPLIFIERS
        MOV     DX,DAC1
        OUT     DX,AX
        MOV     AL,80H
        MOV     DX,DAC2
        OUT     DX,AL
        POP     BP
        POP     DS
        RET     2
RSET1   ENDP

SERVO   PROC    FAR             ;THIS ISR SCANS POSITION OF
        PUSH    AX              ;THE SERVO MOTORS AND COMPARES
        PUSH    BX              ;IT TO DESIRED POSITION OF THE
        PUSH    CX              ;MOTORS TO BRING EACH MOTOR TO
        PUSH    DX              ;THE TARGET POSITON
        PUSH    DS
        PUSH    DI
        MOV     AX,DATA1
        MOV     DS,AX
        MOV     AX,SCAN         ;LOAD TIMER1 WITH PROPER DELAY
        MOV     DX,TIMER1
        OUT     DX,AL
        MOV     AL,AH
        OUT     DX,AL
```

```
          MOV     DX,NCODER
          IN      AX,DX             ;PRESENT POSITION1 IN AX
          MOV     BX,AX             ;PRESENT POSITION1 IN BX
          SUB     AX,[POSTN1]       ;VELOCITY IN AX
          MOV     [POSTN1],BX       ;PRESENT - PREVIOUS

          SUB     BX,[DSIRD1]       ;POSITION ERROR IN BX
          ADD     AX,BX             ;ERROR1 IN AX

;INTEGRATOR FOR MOTOR1
          MOV     DI,CNTR           ;IN THIS PART THE ERRORS OF
          DEC     DI                ;THE FOUR SCANS ARE ADDED
          DEC     DI                ;TOGETHER TO IMPLEMENT AN

          JGE     SUMER             ;INTEGRATOR WITH TIME CONSTANT

          MOV     DI,4              ;APPROXIMATELY 20 mS

SUMER:    MOV     CNTR,DI
          MOV     [INTG+DI],AX
          SUB     AX,AX
          ADD     AX,[INTG]
          ADD     AX,[INTG+2]
          ADD     AX,[INTG+4]

          ADD     AX,500            ;ADJUST FOR BIPOLAR
          JGE     AAA
          SUB     AX,AX             ;CHECK THAT ERROR IS WITHIN
          JMP     BBB               ;RANGE OF THE D/A

AAA:      CMP     AX,3FFH           ;(1000 COUNTS FOR 10 BIT)
          JLE     BBB
          MOV     AX,3FFH
BBB:      MOV     DX,DAC1
          OUT     DX,AX

          MOV     DX,ADC            ;READ THE PRESENT POSITION
          IN      AL,DX
          MOV     [POSTN2],AL
          SUB     AL,[DSIRD2]       ;ERROR2 IN AL
          MOV     AH,AL             ;KEEP ERROR IN AH
;INTEGRATOR FOR MOTOR2
          ADD     AL,[OLDE]         ;JUST ADD PREVIOUS ERROR FOR
          MOV     [OLDE],AH         ;THE GRIPPER INTEGRATOR
          ADD     AL,127            ;CHECK FOR 8 BIT RANGE
          MOV     DX,DAC2
          OUT     DX,AL

          POP     DI
          POP     DS
          POP     DX
          POP     CX
          POP     BX
          MOV     AL,61H            ;END OF INTERRUPT COMMAND
          OUT     18H,AL
```

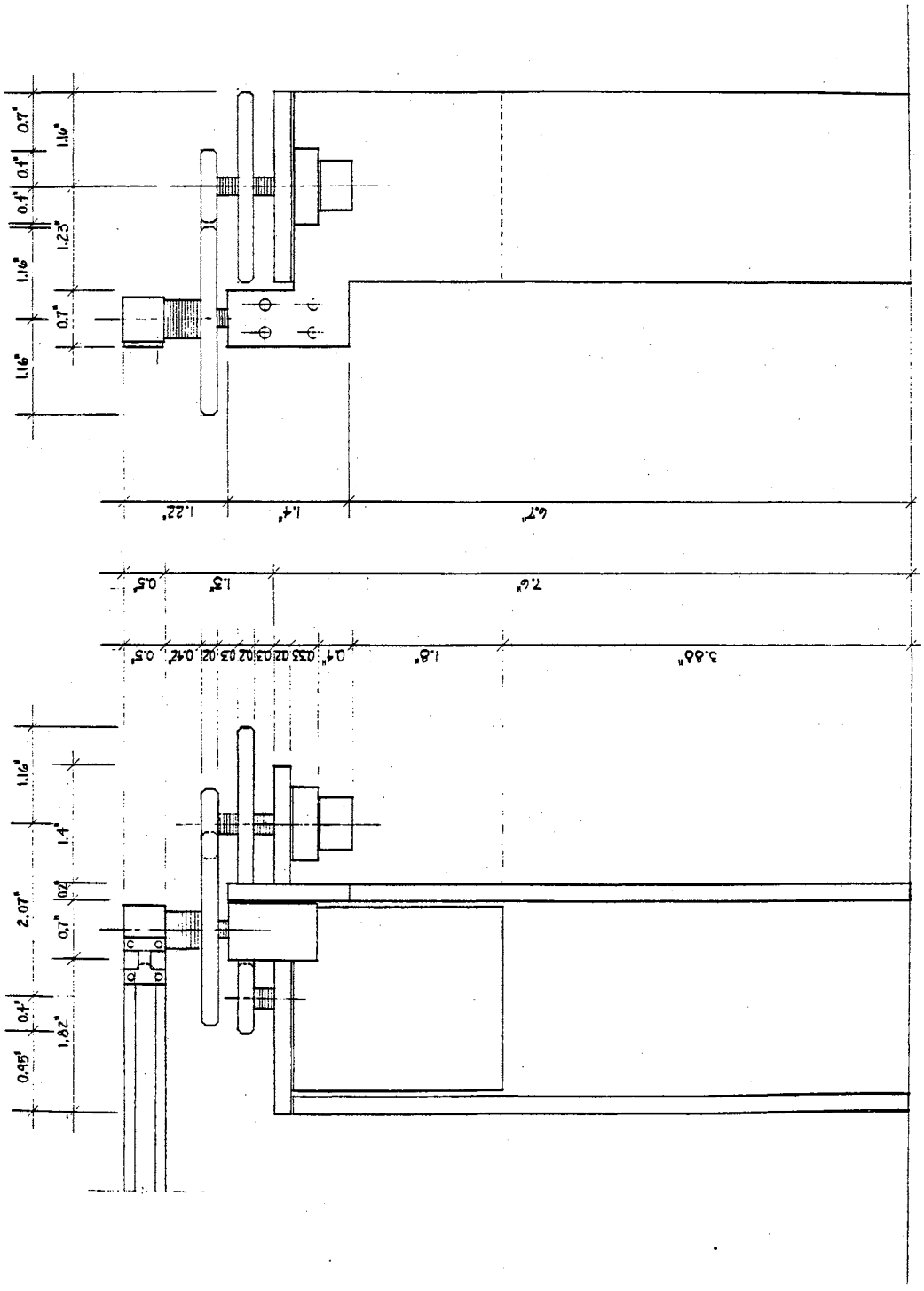
```
        POP     AX
        IRET
SERVO   ENDP

RSET2   PROC    FAR             ;THIS PROCEDURE STARTS SERVO

        PUSH    DS              ;MOTOR SCANNING
        MOV     AX,DATA1
        MOV     DS,AX
        CLI

        MOV     DX,NCODER       ;DESIRED 1 <--- PRESENT1
        IN      AX,DX
        MOV     [DSIRD1],AX

        MOV     DX,ADC          ;DESIRED 2 <--- PRESENT2
        IN      AL,DX
        MOV     [DSIRD2],AL

        MOV     DX,TIMER1
        OUT     DX,AL           ;START SCANING THE SERVOS
        OUT     DX,AL

        POP     DS
        STI
        RET
RSET2   ENDP
COD1    ENDS
        END
```

APPENDIX C

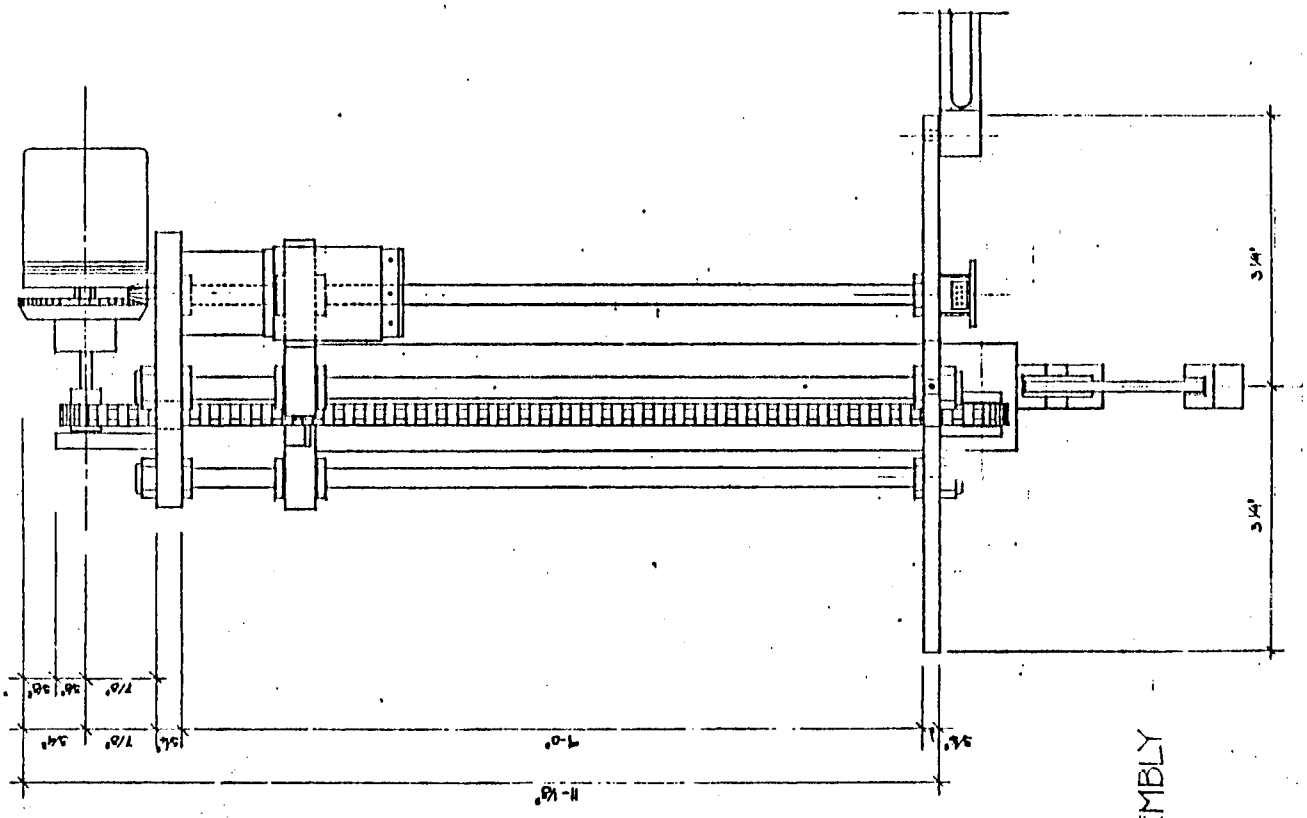
MECHANICAL DRAWINGS



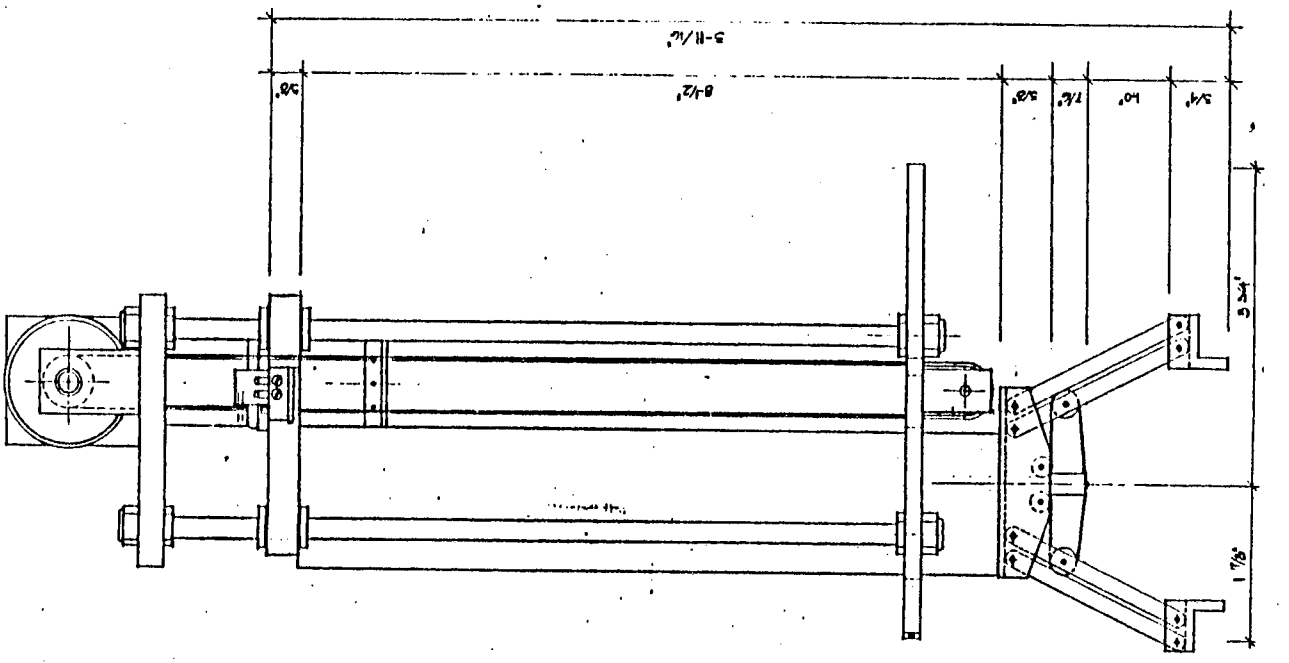
LOBSTER TOP VIEW



BASE SIDE VIEWS



Z-AXIS ASSEMBLY



VITA

Vahid Taban

Candidate for the Degree of

Master of Science

Thesis: DESIGN AND PROGRAMMING OF A LOBSTER ROBOT

Major Field: General Engineering

Biographical:

Personal Data: Born in Tehran, Iran, September 14, 1958, the son of Mr. and Mrs. N. Taban.

Education: Graduated from Hadaf High School, Tehran, Iran, in June, 1975; received the Bachelor of Science degree in Mechanical Engineering from Oklahoma State University in December, 1982; completed requirements for the Master of Science Degree at Oklahoma State University in May, 1986.