To the Graduate Council:

I am submitting herewith a thesis written by Gholam-Hassan Ghaicepour entitled "The Hardware and Software Design and Implementation of a Robot System." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

Walter L. Green, Major Professor

We have read this thesis
and recommend its acceptance:

Accepted for the Council:

Vice Provost
and Dean of The Graduate School

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Master's degree at The University of Tennessee, Knoxville, I agree that the Library shall make it available to borrowers under rules of the Library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of the source is made.

Permission for extensive quotation from or reproduction of this thesis may be granted by my major professor, or in his absence, by the Head of Interlibrary Services when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Signature M. Khaicpo

Date Sept 11, 1986

THE HARDWARE AND SOFTWARE DESIGN AND IMPLEMENTATION OF A ROBOT SYSTEM

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Gholam-Hassan Ghaicepour

December 1986

ACKNOWLEDGMENTS

The author wishes to express his sincere appreciation to his major professor, Dr. Walter L. Green, for his encouragement, patience, and helpful advice throughout the course of his research.

Special thanks go to Dr. J. M. Bailey and Dr. J. M. Googe, the members of his committee, for review of the thesis.

A special thanks is due to Mr. Rick Igou for his valuable help with the computer material.

Finally, the author wishes to express his appreciation to his parents and his brother for their support and encouragement throughout his academic career.

ABSTRACT

A robot arm system with three axis of freedom is designed and constructed. The three degrees of motion consists of 330 degrees of rotation around the base axis and two arms of approximately 18 inches long with 90 degrees of rotation at their joints. The robot arm has the ability to carry a load of up to six pounds at the end of the second arm.

Analog feedback control theory is applied to control the motion of the two arms. A microprocessor based computer (SBE 200), coupled with digital control, is utilized to control the motion of the base axis. The FORTH programming language and Intel 8085 assembly language are used to develop the supporting software programs.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The concept of an industrial robot was patented in 1954 by George C. Devol [1]. The principal objective was to construct a controlled mechanical arm which would not only be able to do several tasks automatically, but also have the capabilty of being reprogrammed so that it would perform a new task without expensive retooling.

Robotics is a young, fast-growing interdisciplinary. It covers wide aspects of research activities: Preception, action control, task level strategies, mechanisms, and so on. These different disciplines, originally from other areas, are now converging into a new field that is called robotics.

Analog and digital closed-loop feedback control systems are applied to robot systems extensively. Design of fine mechanisms and fine control with sensor feedback are important to facilitate good action.

In recent years the advances made in microcomputer and microprocessors have infused new life to the growth of digital control systems. With inceased computer control and advances in sensory preception, tactile and visual sensors, robots are " getting smarter " and their application to new and more challenging task is increasing [2-3].

The main objective of this thesis is to design and construct a robot arm system that will be used as a research tool at the Robotics and Integrated Manufacturing Laboratory of the Electrical Engineering Department of the University of Tennessee, Knoxville . At the present time the robot arm has

three degrees of freedom (base, shoulder, and elbow joints). The shoulder and the elbow joints are controlled by using an analog closed-loop feedback control system. They are capable of rotating 90 degrees at their joints. Digital control is applied to control the base joint which can be moved 330 degrees around the axis. An SBE 200 microcomputer, which is a high-performance 16/32-bit computer system featuring the ModulasTen [1]M68K10 single-board computer with a 10 MHZ 68000 microprocessor, is utilized. The operating system is polyFORTH/32 [2] which is supplied on a Winchester disk with the SBE 200 system. Two Data Translation, Inc. Intelligent Analog Peripheral (IAP) boards (DT3752,DT3762) are used to provide all the D/A, A/D convertors and subroutines necessary to implement the digital control loop.

Chapter 2 gives an introduction to actuator systems used in typical industrial robots. Hydraulic, electric, and pneumatic systems are discussed. Control loops for robots using DC torque motors as drive systems are presented in chapter 3. Two control concepts, one using current amplifiers and the other using voltage amplifiers are described in chapter 3. Chapter 4 presents the mechanical design and the constructed robot system. The computer system used to control the base axis is discussed in this chapter. Analog and digital control alogrithms applied to the joints are explained in detail. The supporting MULTIBUS communication programs are given in chapter 4. Suggestions for further development of the system are discussed briefly in chapter 5.

---

[1]ModulasTen, M68K10 are trademark of SBE, Inc.

[2]polyFORTII, FORTII are trademark of FORTII, Inc.

CHAPTER 2

DRIVES AND ACTUATOR SYSTEMS

To convert electrical command signals to mechanical motions, axes of a robot manipulator requires actuation mechanisms. The actuation mechanism may operate by hydraulic, pneumatic, or electrical means. Implimentation may employ either servomechanisms with feedback or nonservo open-loop regulation. The electric systems can be either AC sevomotors, DC motors, or stepping motors. DC motors and hydraulic systems are widely used as actuators.

The selection of an actuator system is depentent on the application and the enviroment in which the robot system operates. Factors influencing the choice are as follows:

1. In factories, air and oil pressure are readily available. For economic reasons pneumatic on hydraulic devices may be favored.

2. Electrical actuators are excellent where high accuracy, repeatability, and quiet operation are needed.

3. Pneumatic actuators are generally cheap, clean, and safe and can produce moderately high forces, but the compressibility of air reduces the " stiffness " of pneumatic systems, and this can be a disadvantage in some circumstances.

4. Hydraulic actuators are very stiff in action and produce high forces, but they are basically unclean and relatively expensive.

2.1 Hydraulic Systems

Hydraulic systems are used because of their ability to deliver substantial power while being relatively small in size. Hydraulic systems use of an incompressible fluid (an oil) which is forced under presure into a cylinder [5-7]. Hydraulic systems as shown in Figure 2.1 generally comprise the following components:

1. A servovalve for each axis of motion.
2. A hydraulic motor.
3. A sump.
4. A hydraulic power supply.

2.1.1 Servovalve

The electrohydraulic servovalve controls the flow of the high-pressure oil to the hydraulic motor. The servovalve receives a voltage-actuating signal and uses it to drive either an electric motor or a solenoid device, which moves the valve spool. The magnitude of the input voltage (or current) defines the flow rate of oil through the valve. The flow rate of oil through the valve is proportional to the velocity of the hydraulic motor. The time constant of a servovalve, in a high-power system, is on the order of 5 ms, and is usually negligible compared with the other lags in the system.

2.1.2 The Hydraulic Motor

The Hydraulic motor is either a hydraulic cylinder for linear motion or a rotary-type motor for angular motion.

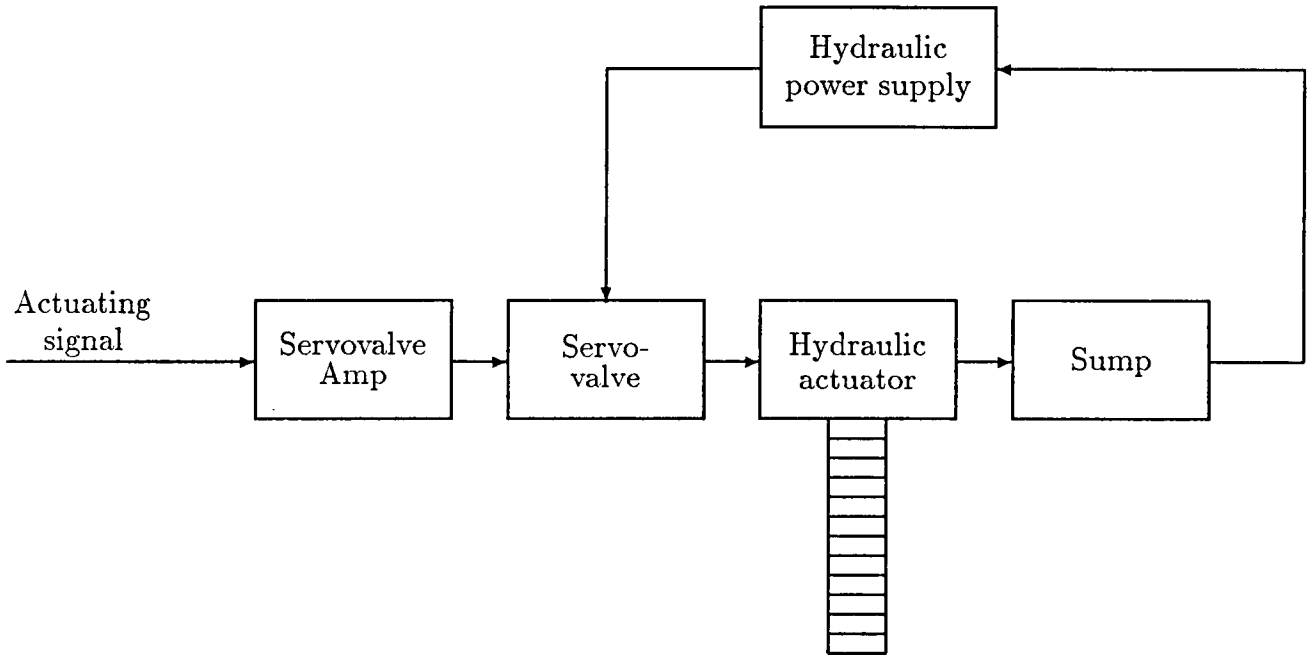


Figure 2.1: Hydraulic drive system used in industrial robots.

The hydraulic cylinder, due to the large quantity of high-pressure oil which it contains, is limited to a relatively small motion. The rotary hydraulic motor is usually used in larger power servosystems. It operates at high speeds and is geared down to the shaft which drives the robot joint.

2.1.3 The Sump

The used oil is returned to the sump,or tank, through a special return line. The oil is fed back to the hydraulic power supply and forms its source of fluid.

2.1.4 Hydraulic power supply

The hydraulic power supply is a source of high-pressure oil for the hydraulic motor and the servovalve. The main components of the hydraulic power supply are as follows:

1. A pump for supplying the high-pressure oil. The frequently used types are the gear pump and radial or axial displacement pumps.

2. An electric motor, usually a three-phase induction motor, for driving the pump.

3. A fine filter for protecting the servosystem from any dirt or chips.

4. A coarse filter, located at the input of the pump, for protecting the latter against contamination that has entered into the oil supply.

5. A check valve for eliminating a reverse flow from the accumulator into the pump.

6. A pressure-regulating valve for controlling the supply pressure to the servosystem.

7. An accumulator for storing hydraulic energy and for smoothing the pulsating flow. Accumulators can provide a large amount of energy over a short interval of time and are used where the load is characterized by an average demand which is far below the required peak. The accumulator supplies the peak requirements and is subsequently recharged by the pump. Another function of the accumulator is to smooth the pulsations caused by the pump, and the variations caused by the sudden motions of the valve. The accumulator functions like a capacitor in an electric circuit.

2.2 Electric Systems

Electric systems are used because of their high accuracy and repeatability. The electric motors are either three-phase AC motors, DC servo motors, or stepping motors. Regardless of the type, the electric motor provides rotary motion that is geared down to provide the proper and direction of motion for the robot. Electric motors are found on medium to small size robots.

2.2.1 AC Servomotors

The following material is presented as an introduction to AC servomotors , since at the present time, few robot manufacturers have adapted AC servomotors [8].

The standard sevomotor is a two-phase reversible AC induction motor having two winding in the stator (i.e., a fixed or reference phase winding

spaced 90 degrees apart electrically from a variable or control phase winding) and a squirrel cage rotor. In operation, the fixed phase is excited at rated voltage, and a second voltage, 90 degrees out of phase, is applied to the control phase. A rotating field results in the stator and generates a rotor torque proportional to the magnitude of control phase voltage. A high torque-to-inertia ratio and a straight-line speed-torque are inherent characteristics of this type of motor.

Classes of AC servomotors include the following:

1. *Standard Low-Inertia Servomotors* for precision servos where rapid reversal and high velocity are necessary.

2. *Viscous Damped Servomotors* that provide greater damping than is available through the technique of reducing the no-load speed. This is normally accomplished by means of a low-inertia drag cup coupling a fixed magnetic field. These motors are normally used in simple instrument servos where high damping and/or low time constants are required, and they can be provided with almost any degree of damping desired.

3. *Inertial Damped Servomotors* introduce the equivalent of viscous friction into a loop for stabilization without causing velocity lag. Damping is provided on an acceleration or deceleration basis by the viscous coupling of a low-inertia drag cup to a freely rotating permanent magnet flywheel. These motors make possible system cutoff frequencies up to 28 HZ.

2.2.2 DC Motors

DC motors allow precise control of either the speed or the torque by manipulating the applied voltage or current to the motor. They are well suited for driving the axes of small-to medium-size robots.

Unlike AC motors with their characteristic rotating stator fields, DC motors have a fixed stationary magnetic field. The DC motor fixed field magnets may be permanent magnets or electromagnets. The DC motor rotor is called *armature*, and it consists of two or more wirewound electromagnetic poles. The armature magnetic field is created by passing DC current through the armature coil. Current flows to the armature through brushes and a segmented slip ring called a *commutor.*

There is no frequency dependent upper limit on DC armature rpm as there is in an AC motor with its synchronous stator field speed. Armature speed in a DC motor is governed by the amount of torque which can be generated by the armature coil/field coil interaction. In a seperately excited DC motor Torque depends on magnetic field strength which depends almost linearly on *armature current:*

$$T_m = K_t I_a \tag{2.1}$$

where

$$\begin{aligned} T_m &: \textit{Motor torque delivered } (in - lb) \\ K_t &: \textit{Motor torque constant } (in - lb/amp) \\ I_a &: \textit{Armature current } (amp). \end{aligned}$$

The DC motor torque constant is determined by the particular motor design, and is specified for each motor by the manufacturer.

Armature current is supplied to the armature by applying a DC voltage to the motor terminals. At zero speed, the armature current I_a is

$$I_a = \frac{V}{R_a} \tag{2.2}$$

where

I_a : *Armature current (amps)*

V : *Voltage applied at motor terminals (DCvolt)*

R_a : *Armature winding resistance (ohms).*

As the armature winding rotats through the fixed magnetic field of the motor field winding (or magnets) a voltage is generated (Faraday's Law of Induction) which opposes the applied terminal voltage. The faster the armature turns, the greater this opposing voltage becomes. It is appropriately called back EMF or counter EMF and can be written as :

$$E_b = K_b N \tag{2.3}$$

where

E_b : *Back EMF (volts)*

K_b : *Back EMF constant (volts/rpm)*

N : *Armature speed (rpm).*

Equation 2.2 is only appropriate for no armature rotation. The more general expression for armature current takes rotation into considration as follows :

$$I_a = \frac{V - E_b}{R_a} \tag{2.4}$$

or

$$I_a = \frac{V - K_b N}{R_a} \tag{2.5}$$

where

V : *Terminal voltage (volts)*

K_b : *Back EMF constant (volts/rpm)*

N : *Motor (armature) speed (rpm)*

R_a : *Armature resistance (ohms).*

The ideal DC motor torque-speed relation can be obtained by combining Equations 2.1 and 2.5 to yield :

$$T_m = K_t I_a = \frac{K_t}{R_a}(V - K_b N) \tag{2.6}$$

This relationship assumes a constant field and a linear torque-current relation which is very close to the actual case for permanent magnet DC motors and servomotors. One additional motor characteristic of concern is the internal friction of the motor. Often this friction is linear with speed and is expressed :

$$T_v = K_v N \tag{2.7}$$

where

T_v : *Internal viscous friction* $(in - lb)$

K_v : *Viscous friction constant* $(in - lb/rpm)$

N : *Motor speed (rpm).*

If viscous friction is significant, it must be subtracted from the ideal torque expression of Equation 2.6 to yield net developed torque.

Shunt motor speed can be controlled by field voltage. Control is possible because field strength affects armature back EMF. Reducing field voltage reduces back EMF generated in the motor armature, so that armature current, torque, and speed increase.

The preceding discussion and analysis of ideal DC motors predicts performance rather well for DC motors with permanent magnet fields. DC motors also have wirewound field coils and are often classified in terms of how the field winding is connected with respect to the armature.

Figure 2.2 shows the circuit diagram of a DC *shunt motor*. The field winding is wired in parallel with the armature winding. Figure 2.3 shows a DC *series motor* wherein the field is wired in series with the armature. Figure 2.4 shows the circuit for a combination shunt-series field connection known as a *compound motor*.

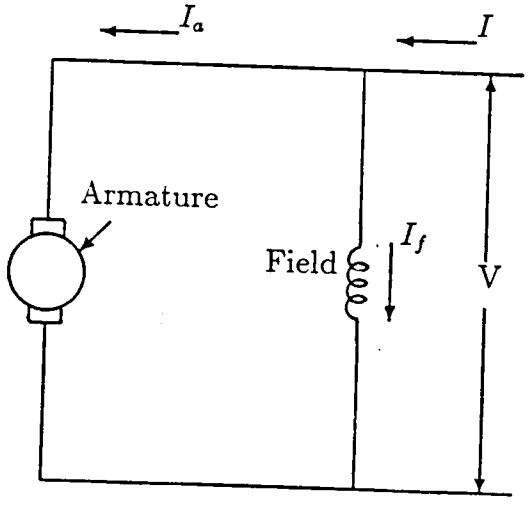


Figure 2.2: Shunt-Wound DC motor.

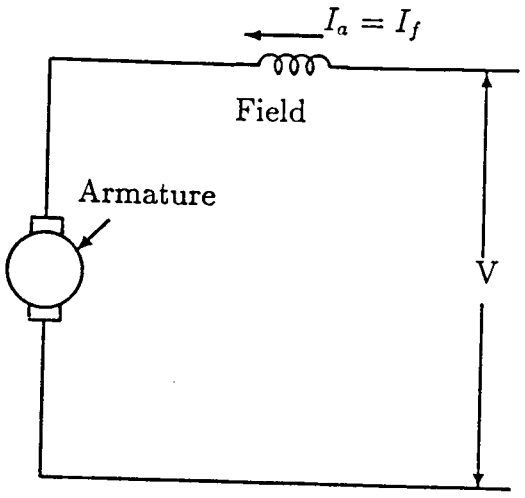


Figure 2.3: Series-Wound DC motor.

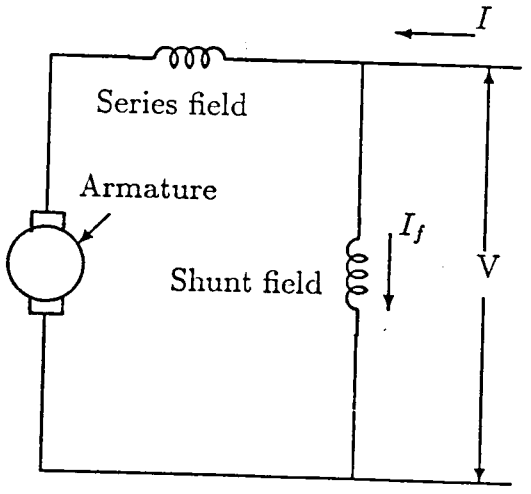


Figure 2.4: Compound-Wound DC motor.

In a shunt motor, the field current, hence field strength, depends only on the resistance of the field winding. In this sense, the shunt motor is like a permanent magnet motor except that field current increases and decreases with " applied terminal voltage ." Figure 2.5 shows the characteristic torque-speed curve and current-speed curve for a DC shunt motor.

The speed of a shunt motor can be controlled by varying terminal voltage, thus armature current in much the same way as one controls a permanent magnet DC motor. The motor speed is controlled by varying full power to the motor. The advantage of DC shunt motors is that sensitivity of motor speed to changes in torque is relatively small. However, if the armature is stalled, armature current draw can be very high.

In the DC series motor Figure 2.3, field current and armature current are equal. Maximum current flows through the armature and field at zero speed (back EMF=0), hence high torque is developed on startup. Primary applications of DC series motors are when high starting torque is required. Since both field current and armature current vary with armature speed, there is a pronounced effect of speed variation on torque. A series-wound motor is more sensitive to changes in load than are the other types of wound field motors. Figure 2.6 shows the typical torque-speed and torque-current curves of series-conected motors.

An important advantage of series-wound motors is that armature current is limited by the field resistance at stall torque. A serious disadvantage is that the motor will run away (i.e., overspeed) if the load is removed. Also, the direction of rotation of the series motor can not be reversed by simply reversing applied voltage. A separate set of field coils connected to create a magnetic field opposite that of the series field must be switched into the circuit.

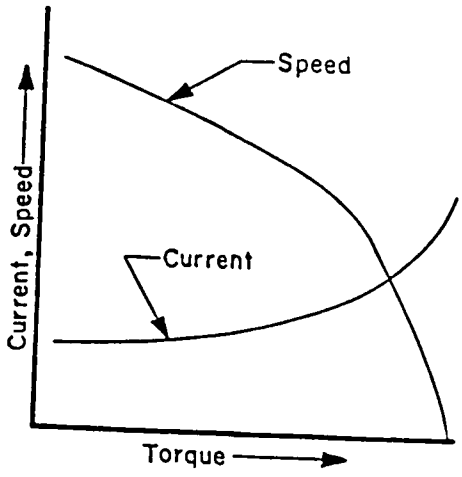


Figure 2.5: Shunt motor performance characteristics(typical).

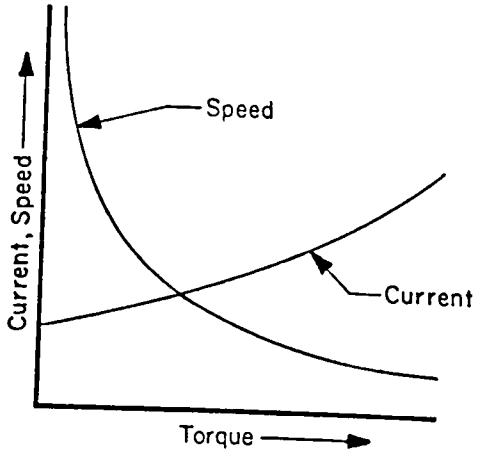


Figure 2.6: Series motor performance characteristics(typical).

The torque-speed characteristics of a compound motor can be tailored by design of relative shunt field/series field strength. Figure 2.7 shows a typical compound motor curve. The torque-speed curve can be shifted by changing the relative direction of the series and shunt fields.

Compound motors offer high starting torque and can be tailored for relatively constant speed over a wide range of torque output. The main disadvantage is that the current through both field winding must be reversed relative to the armature current in order to reverse motor direction [9-10].

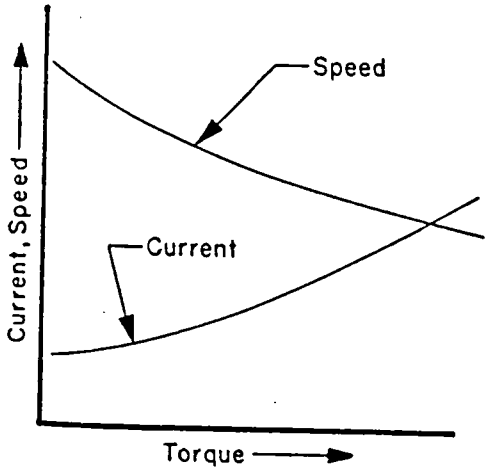


Figure 2.7: Compound motor performance characteristics (typical).

2.2.3 Stepping Motors

Stepping motor is a device that will transform digital electric input to a mechanical motion. The rotation of shaft will be in equal increments called the " steps " if the motor is controlled properly. A stepping motor rotates through a fixed angle called the " step angle " and is expressed in degrees. The smaller the step angle, the higher the resolution of positioning can be. The step number S, which is the number of steps per revolution has the following relation with the step angle θ_s

$$S = \frac{360}{\theta_s} \quad . \tag{2.8}$$

The stepping motors can be classified into several types according to their rotor structure and principle of operation, and they can be listed as follows:

1. Variable-Reluctance (VR) type.
2. Permanent Magnet (PM) type.
3. Hybrid type.
4. Outer-rotor type.
5. Liner type.
6. Monofilar and Bifilar winding type.

Variable-Reluctance and Permanent-Magnet stepping motors are the two most common types in use [11-13].

Figure 2.8 shows the block diagram for a drive system used for a three-phase motor (each set of winding is called a ' phase '). Upon receiving a step command pulse, the logic sequencer determines the phase to be excited (or

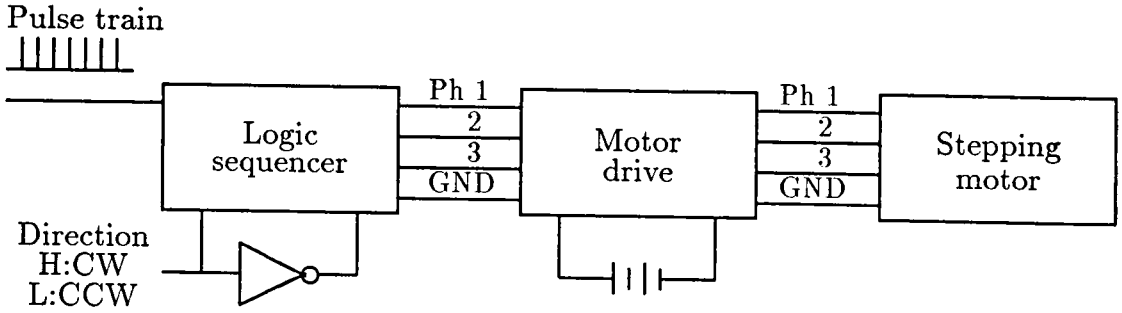


Figure 2.8: Block diagram for motor drive system.

energized) and the phase to be de-energized, and it sends signal to the motor driver which is the stage which controls current supplied to the motor. The logic sequencer is usually assembled with TTL or CMOS intergrated circuit chips. When the potential of an output channel from the logic sequencer is on level H (=high), the power driver works to excite the corresponding phase of the winding. Similarly, if the output is at level L (=low), the phase of the same number is not excited, or it is turned off.

As shown in Figure 2.9 the motor runs clockwise (=CW) in the excitation sequence of $1 \rightarrow 2 \rightarrow 3 \rightarrow 1 \rightarrow \ldots$, the direction of rotation will be counter-clockwise (=CCW) on the reversed sequence of $1 \rightarrow 3 \rightarrow 2 \rightarrow 1 \rightarrow \ldots$. In general, there is no established rule to define which direction is clockwise or counter-clockwise, for a motor rotating clockwise when seen from one end appears to be rotating counter-clockwise if seen from the other end. The direction of rotation is usually defined by agreement between parties concerned.

The excitation used in Figure 2.9 is referred to as the single-phase or one-phase excitation, which means that only one phase of the three (or four in a four-phase motor) is energized at any time. This form of excitation is just been used to explain the fundamental principles of the stepping motor.

2.3 Pneumatic Systems

Pneumatic systems operate by passing compressed air into and out of a cylinder with a piston that provides linear motion. Air is a fluid with very low viscosity and will, therefore, move through the tubing and cylinder very fast, and cycle times of under a second are realizable. However, gases are compressible; consequently, with a given amount of air in the cylinder

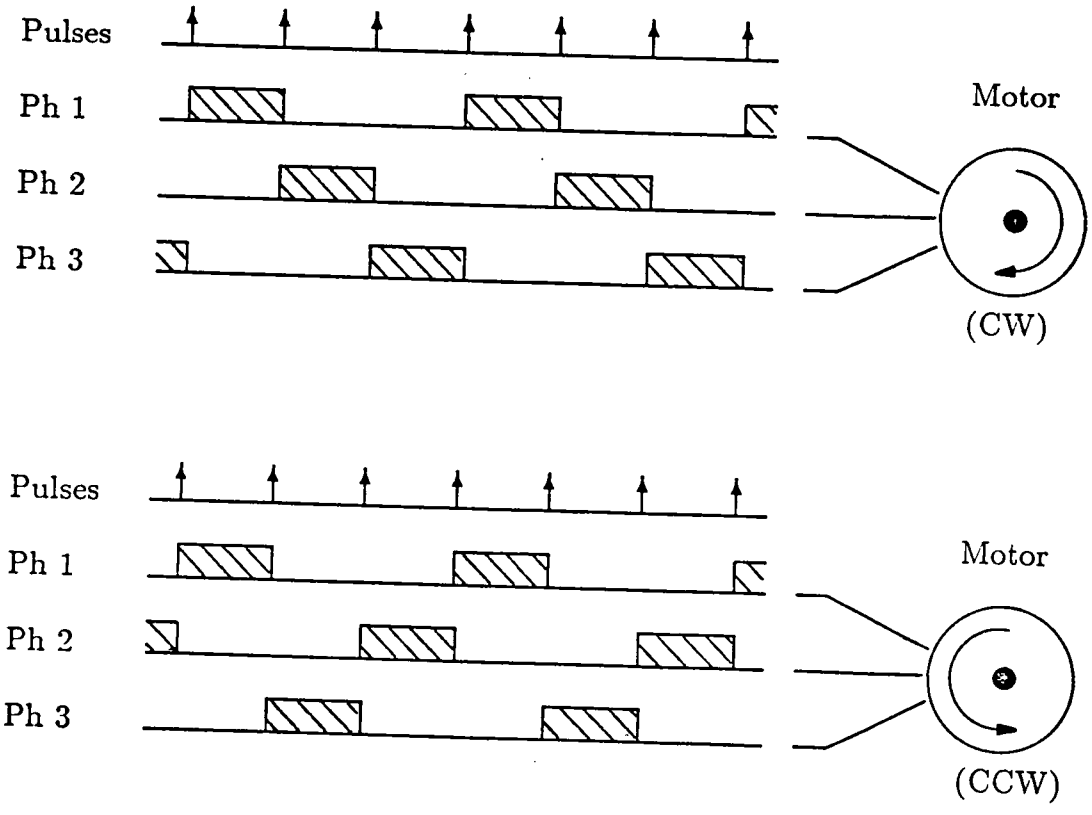


Figure 2.9: Input pulse series and excitation sequence.

position of the piston will be very dependent on the load being imposed on it. The only way of being sure of positionl accuracy is with the use of mechanical stops, with the air providing substantially greater force against the stop than the load to limit the motion of the cylinder. Where rotary motion is required, a lever system, or a rack and pinion, is used. The flow of the air is controlled by solenoid-operated valves. Pneumatic systems require a compressor, which typically is like an internal combusion engine with inlet and outlet valves and reciprocating piston driven by an electric motor. Therefore, the pneumatic systems are usually limited to non-servo robots that depend on stops and gauges to determine their positions. Some designers have developed servo feedback systems that ovecome this drawback in air robots, but they are not in use nearly as much as electric or hydraulic robots [14].

2.4 Advantages and Disadvantages of the Actuator Systems

From the material presented in this chapter the advantages and disadvantages of each actuating system can be summarized as follows:

Electric actuators

Advantages :

1. Electric actuators are fast and accurate.
2. It is possible to apply sophisticated control techniques.
3. Easily available and relatively inexpensive.
4. Simple to use.

Disadvantages :

1. Require gear trains or the like for transmission of power.
2. Gear backlash limits precision.
3. Electric arcing might be a problem.
4. Power limit.

Hydraulic actuators

Advantages :

1. Large lift capacity.
2. Moderate speed.
3. Oil is incompressible; hence, once positioned, the joint can be held motionless.
4. Offers accurate control.

Disadvantages :

1. Hydraulic systems are expensive.
2. They pollute the workspace with fluids and noise.
3. Not suitable for really high speed cycling.

Pneumatic actuators

Advantages :

1. Relatively inexpensive.
2. High speed.
3. They do not pollute the workspace with fluids.
4. Can be used in laboratory work.

Disadvantages :

1. The compressibility of air limits their accuracy.
2. Noise pollution still exists.
3. Leakage of air is a major concern.
4. Additional air filtering system and drying system are needed.
5. Increased maintenance and construction requirements.

CHAPTER 3

CONTROL LOOPS FOR ROBOT SYSTEMS

In robot systems each axis of motion is separately actuated by a control loop which contains the drive system. Control loops for robot systems fall in two categories: open loop and closed loop control systems. In open-loop control systems there is no link or feedback from the output to the input of the system, and therefore the input can not be influenced by the output. To obtain more accurate control, the controlled signal should be fed back and compared with the reference input, and an actuating signal proportional to the difference of the input and the output must be sent through the system to correct the error. A system like this which has one or more feedback paths is called a closed-loop system. In closed-loop systems used in a robot arm, the resultant motion is sensed by a feedback device, such as an incremental encoder.

Robots that are controlled by an open-loop control system will use a stepping motor as their drive system. The structure of a 6-axis open-loop robot system using stepping motor is shown in Figure 3.1. Since stepping motors provide the simplest means of converting input electrical pulses into proportional angular movement, and as a result represent a relatively inexpensive solution to the control problem. There is no feedback from the shaft position, positioning and accuracy is solely a function of the motor's ability to step through the exact number of pulses provided at its input.

If the inertia of the load is a constant, then acceleration and deceleration

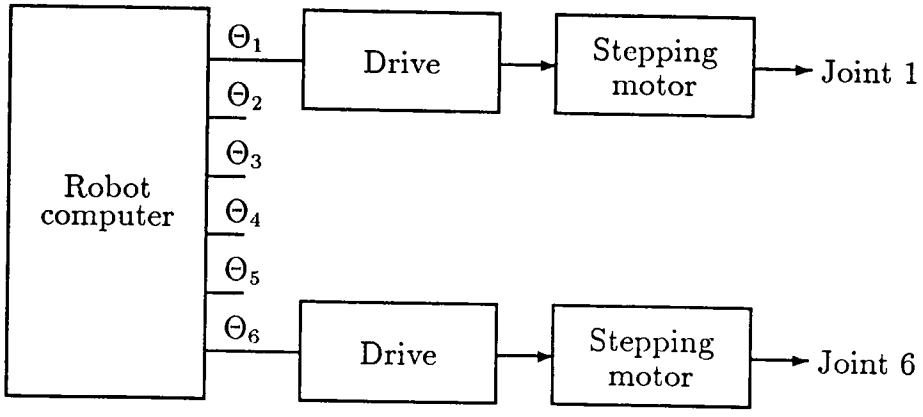


Figure 3.1: Structure of an articulated 6-axis open-loop robot system.

algorithms can be developed that allow step motors to be used at high speed without loss of positional accuracy. In robotic systems, however, motors must develop torques that depend not only on the gripper load but also on the position of the robot arm. If the inertia changes, the physical performance of the motor system may be faster or slower than that used in the design, and the controller may " lose " steps. Therefore, stepping motors are not recommended for use as drives for industrial robots.

A closed-loop (servo) control for a single axis of motion is shown in Figure 3.2. The position and motion of the robot system is controlled from sources both internal and external to the robot arm. Internal control sources include the feedback devices which mechanically measure the angle of the joints and convert the angular measurements to proportional electrical signals. In addition, other feedback devices provide rate-of-change data to be used in controlling the velocity and acceleration/deceleration rates at the tool-center-point. The difference between the actual and the desired values is the error. The error signal is used to drive the motor. The control strategy is designed to eliminate this error or reduce it to a minimum, as in the case of closed-loop negative feedback systems.

The feedback device in the system is an incremental shaft encoder which is mounted on the shaft and supplies a pulsating ouput. The incremental encoder is the most popular feedback device used in robot systems. An encoder is an electromechanical device used to monitor the motion or position of an operating mechanism, and to translate that information into a useful output. Two basic types of optical encoder are available: linear and rotary. Rotary encoders are either incremental or absolute. Functionally, the incremental encoder generates a serial pulsetrain output by reading the number of increments traversed on the code disc track as the encoder shaft rotates.

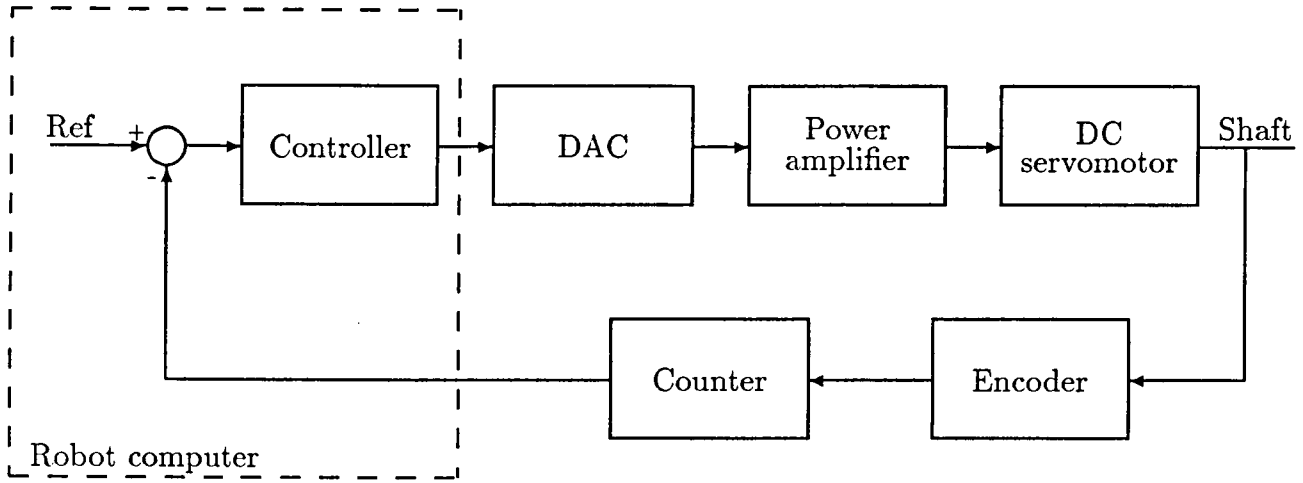


Figure 3.2: Block diagram of a digital closed-loop control system.

In order to know the exact amount of travel, it is necessary to count the output pulses with an external digital counter. The counter, then, maintains a memory of the position information of the encoder.

In the closed-loop system of Figure 3.2 each control loop is closed through the robot computer itself. The encoder pulses are accumulated in a counter, which is sampled by the computer at constant time intervals. In other system configurations, a microprocessor is used as the comparator in each control loop, and a minicomputer coordinates the reference commands to the individual microprocessors.

In closed-loop control system the motion of a robot arm driven by a DC servomotor is controlled basically in two different methods. One method is to control the torque of the robot arm by changing the current applied to the motor. The other approach is to control the motor speed by manipulating the motor voltage [15-18].

The first method treats the torque produced by the motor as an input to the robot joint. In the second method, the robot arm is treated as a load disturbance acting on the motor's shaft.

The selection of the control method is dependent on the application and the environment in which the robot arm operates. The allowable positioning error, accuracy, repeatability, and response time also have to be taken into consideration when an optimum performance is required. When the arm is free to move along some coordinate, the specification of velocity is appropriate. When the robot's end effector might be in contact with another object in such a way as to prevent motion along a coordinate, then specification of torque is appropriate. The design of the control loop and the choice of the drive element require a knowledge of the nature of the application and the loading torques.

3.1 Control Loops Using Current Amplifier

In torque-control loop method, in order to control the motion of the robot arm, an appropriate torque is applied to overcome gravity, dynamic torques due to the moment of inertia and all frictions involved. Since the motor armature current is proportional to the loading torque, a current amplifier should be used in torque-control loops. Block diagram of the system using current amplifier is shown in Fiqure 3.3.

An important advantage of the torque-control approach is that we can maintain a desired torque or force. This is useful in some robotics applications, such as screwing or assembly of mating parts. Another advantage occurs when the robot arm encounters resistance (e.g., the gripper touches a rigid obstacle); it maintains a constant torque and does not try to draw additional power from the electrical source.

The problem with this type of system is the need to have an accurate estimate of the moment of inertia at each joint of the robot arm in order to obtain the desired trajectory. If the actual value of the inertia is smaller than expected, then the torque applied is larger than required. This torque is translated to higher acceleration and consequently higher velocity. This can have disastrous consequences; for example, a part can be struck and broken since the velocity is not zero at the desired target position. On the other hand, if the inertia is larger than expected there is a loss of time, since the arm decelerates a long distance before the target point and " creeps " toward it very slowly.

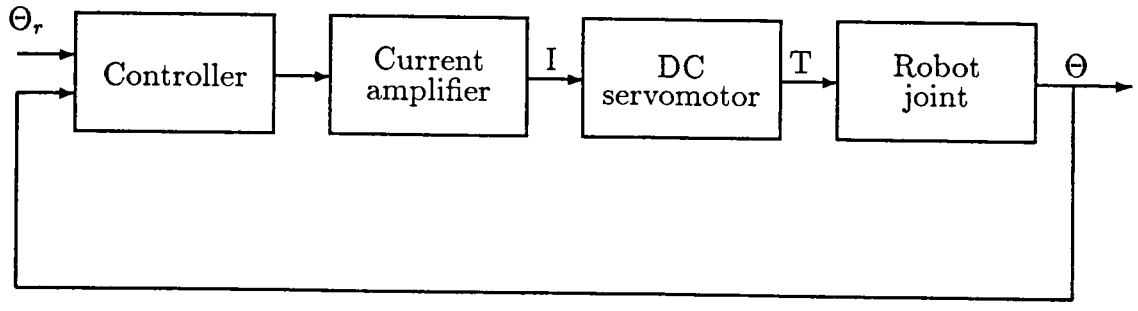


Figure 3.3: Control loop utilizing a current amplifier.

3.2 Control Loop Using Voltage Amplifier

The alternative approach is to control the velocity of the robot arm by manipulation of the DC motor voltage, utilizing a voltage amplifier in the motor's drive unit as shown in Figure 3.4. A similar approach is also usually used in hydraulically driven robots. The main advantage of this approach is that variations in the moment of inertia affect only the time constant of the response but do not result in any disastrous consequences and do not affect the time required to reach the target position. The arm always approaches the target smoothly and at very low speed. The problem with this approach is that the torque is not controlled, and the motor will draw from the voltage amplifier whatever current is required to overcome the disturbance torque. This can lead to burning of the amplifier's fuse when the robot arm encounters a rigid obstacle. Another disadvantage is that this system is not suitable for certain assembly tasks, such as press fitting and screwing, which require a constant torque or force.

Most modern robots use sampled-data control systems. The sampled-data technique can be used with both the velocity-control and the torque-control approach. The control loop is closed either through the robot computer (see Figure 3.2) or in microprocessors, as shown in Figure 3.5.

The hierarchical structure shown in Figure 3.5 is typical of many modern robot systems . At the top of the system hierarchy is the robot supervisory computer, and at the lower level are microprocessors, one for each axis of motion. The supervisory computer performs the following functions :

1. Trajectory planning and interpolation in world coordinates.

2. Coordinate system transformation from world to joint coordinates and

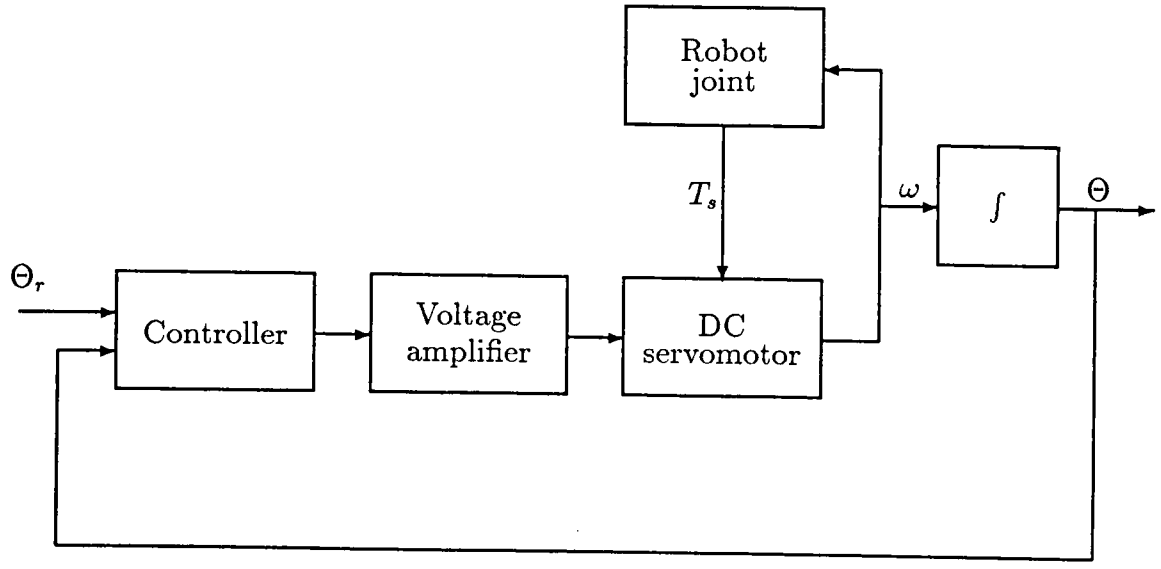


Figure 3.4: Control loop utilizing a voltage amplifier.

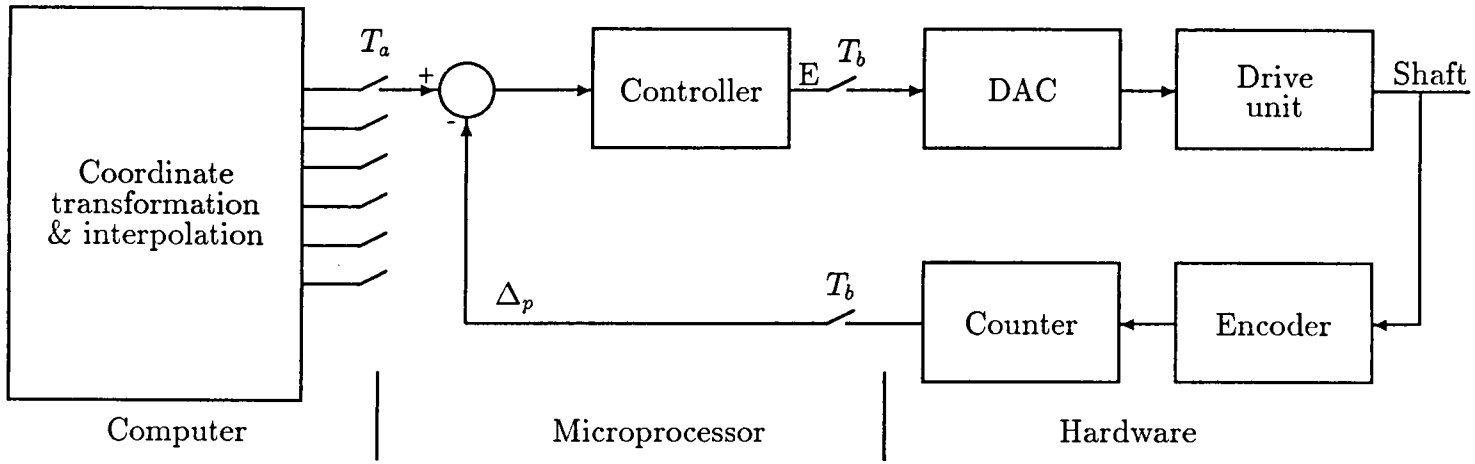


Figure 3.5: Block diagram of a sampled-data closed-loop control system.

sending incremental position commands, to the microprocessor of each joint every T_a milliseconds.

3. Receiving a signal from each microprocessor that the corresponding axis has completed its motion and the next end-point coordinate transformation should be performed.

4. In robot systems which include programming with a high-level language, the supervisory computer also contains the language compiler, and the task programming is executed with the aid of this computer.

At the lower level in the system hierarchy is the axial microprocessor. With this structure, the controller block in Figure 3.3 and Figure 3.4 is included in the microprocessor which controls the corresponding axis. It computes the error signal and sends it through the digital-to-analog converter (DAC) to the drive unit.

In many robot systems, there are two servo loops for each joint control. The inner loop is of an analog type and is closed in the drive unit. It consists of the power amplifier, i.e., the voltage or current amplifier; the joint drive (e.g., a DC servomotor), and a DC tachometer as a velocity feedback device. The outer loop is of a sampled-data type. Its typical feedback device is an incremental encoder interfaced with the microprocessor through a counter which is incremented by the pulse received from the encoder. The microprocessor samples the contents of the counter at fixed time intervals T_b. The number tranferred from the counter to the computer, Δ_p, is equal to the incremented positional displacement . The control program compares the reference from the supervisory computer with the contents of the counter to determine the position error. When the velocity control is applied, this error signal is fed every T_b milliseconds to a DAC, which, in turn, supplies a

voltage proportional to the required axis velocity. When the torque control approach is applied, the error signal is converted in the microprocessor to a corresponding torque signal, and subsequently the latter is used to drive the motor.

The main functions of the axial microprocessor include:

1. Every T_a milliseconds receive a position and/or velocity command from the supervisory computer.

2. Every T_b milliseconds read the counter which stores the position value from the joint encoder.

3. Check if the actual position has reached the required end-point position. If reached, send an appropriate signal to the supervisory computer.

4. Based upon the instantaneous desired and actual positions, calculate the axial position error and use it in a specified control algorithm which depends on the control approach.

5. Send the alogorithm result to the DAC.

The DAC converts the binary-word input to a proportional voltage which is used to drive the axis of motion through the drive unit.

CHAPTER 4

DESIGN AND IMPLEMENTATION

Chapters 1 through 3 presented the main objectives of this thesis and provided an introduction to the background theory necessary to design and implement the control strategy.

In this chapter, the first section contains a brief summary of the mechanical design of the robot arm. The main components of the digital control system which will be used to control the movement of the base joint of the robot arm are presented in section 4.2. In section 4.3 the control alogrithms are discussed in detail.

4.1 Mechanical Design of the Robot Arm

This part of the project was done as a term project to fulfill the requirements for completion of Mechanical Engineering 4624. Two teams of four mechanical engineering students, under supervision of Dr. S. E. Becker of the Mechanical Engineering Department, were given the task to design the structure of the robot arm. The design developed had to meet the following specifications:

1. Fairly simple construction as facilitated by tools available to the Electrical Engineering Department.

2. Rotation around the base joint of at least 300 degrees with a minimum angular velocity of 1/2 revolution per second.

3. Two arms of approximately 18 inches, each capable of moving at an angular velocity of 1/8 revolution per second and 90 degrees of rotation at their joints.

4. Ability to carry a load of up to 6 pounds at the end of the second arm.

5. Mechanical stops with electrical switches to prevent rotation.

6. Use of provided shaft encoder at base and potentiometers at joints to allow positional feedback.

7. Light weight and low bulk to allow for ease of mobility.

8. As low a cost as posible, preferably under a budget of $500.

From the two designs presented at the end of the quarter, it was decided to construct the design given by the team of Messrs. Mark Collette, Rick Dorris, Jeff Jones, and Randy Oliver. Figure 4.1 shows the assembled structure of the design. The design had the following features:

1. Square aluminum tubing was used for the arms. Three inch square was selected for the upper arm (shoulder) and two inch square for the lower arm (forearm).

2. One inch square steel tubing was used to construct and aluminum sheet metal was used to cover the base.

3. Bevel gear with a drive shaft was used for the forearm.

4. A " Lazy Susan " bearing plate was used for rotation of the base. Using the bearing plate would allow direct coupling to the rotational axis (base).

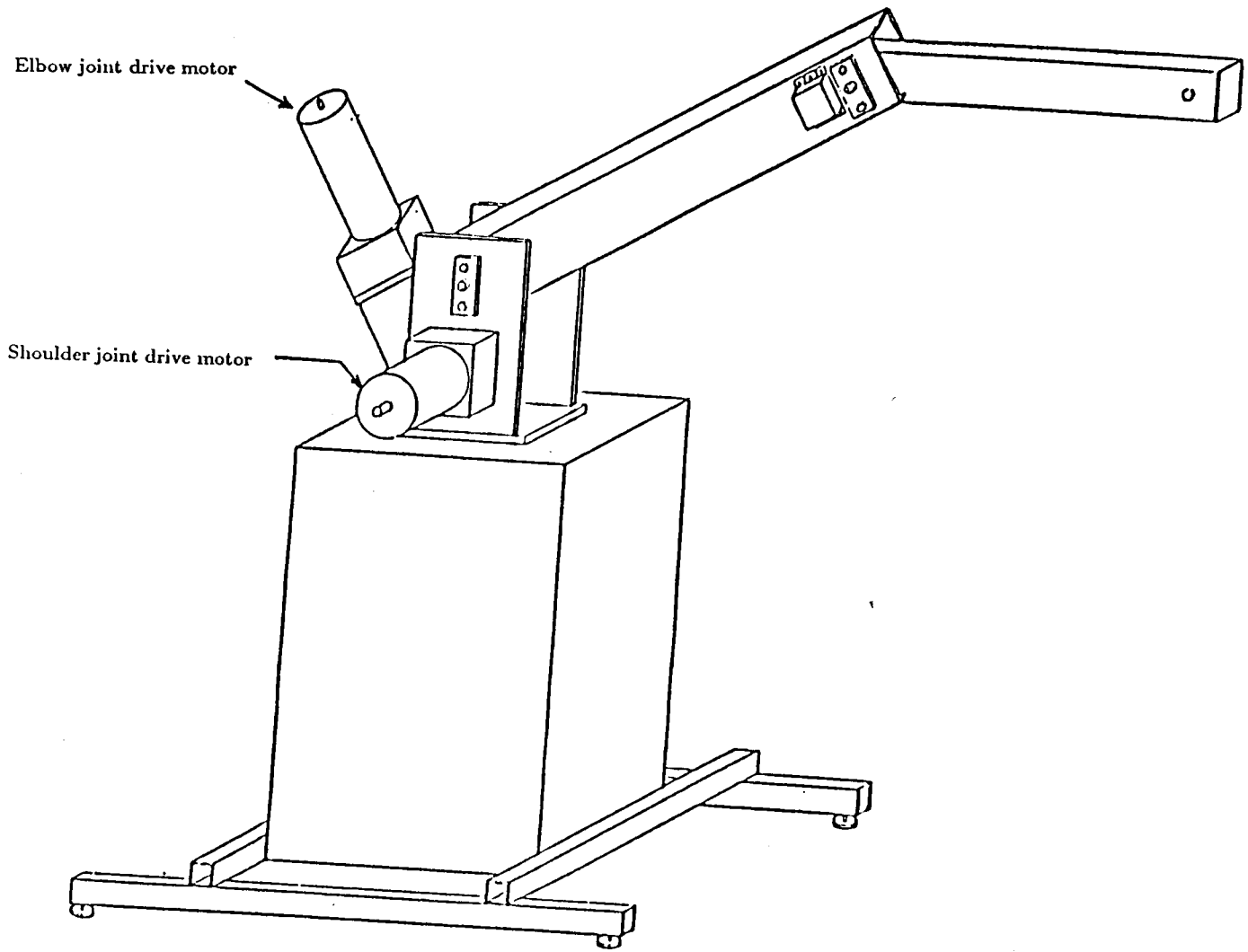


Figure 4.1: The assembled structure of the robot system

4.2 The Digital System

This section contains a brief introduction to the main components of the digital control system used in the base joint of the robot arm. The presented material is extracted from the reference manuals provided by the manufactures.

4.2.1 SBE 200 Computer System

The SBE 200 is a high-performance 16/32-bit microcomputer system featuring the ModulasTen M68K10 single-board computer with a 10 MHz 68000 microprocessor. The SBE 200 also features Multibus expansion capability.

Generally, the SBE 200 includes a 10 Megabyte 5-1/4" Winchester disk drive, a removable backup device (tape or floppy disk drive), and a high-speed Winchester/floppy disk controller.

The Winchester disk supplied with the SBE 200 contains the operating system. The operating system for this particular system is polyFORTH/32, and will boot from the Winchester disk [24].

4.2.1.1 ModulasTen^{TM} M68K10^{TM}

The M68K10 is a high-performance MULTIBUS [1] compatible single-board computer featuring a 10 MHz 68000 microprocessor. It is designed to function as a stand-alone microcomputer, a single CPU/controller in a MULTIBUS system, or a single CPU element in a multiprocessor configuration. The M68K10 allows users to take advantage of the wide choice of I/O and memory

[1]MULTIBUS, iSBX ,and MULTIMODULE are trademak of Intel corporation.

products compatible with the MULTIBUS and iSBX standards, and to enjoy the superior performance and ease of programming provided by the 68000 family of microprocessors.

The 68000's 32-bit internal architecture, speed, and well-designed instruction set account for its high popularity among OEMs who are building systems that must perform sophisticated tasks. For systems which must respond to real-time events, or systems in which execution speed is critical, the 68000's superior assembly language instruction set is highly advantageous.

Memory

Dual-Ported Dynamic RAM. The M68K10 can be configured with up to 265K bytes of fast on-board dynamic RAM using 64K by 1 bit memory components. A simple jumper change allows the M68K10 to accommodate 256K by 1-bit memory components, increasing the on-board dynamic RAM capacity to one Megabyte.

One half of the M68K10's dynamic RAM is resident on-board; the remainder is provided the MXM10 memory expansion module. With 64K-byte memory components, the M68K10 can supply 128K bytes of dynamic RAM without an MXM10. The MXM10 expansion module is required only if your application requires more than 128K bytes of memory.

The 68000 operates at full speed when accessing on-board dynamic RAM; an on-board memory read or write operation requires only 400 nanoseconds. The M68K10 automatically performs the periodic refresh operation required to maintain the data stored in dynamic RAM. The refresh controller interleaves the refresh operations with normal data accesses.

Part or all of the M68K10's on-board dynamic RAM may be accessed by other MULTIBUS masters just as if the M68K10 were a memory board.

Jumpers allow this dual-ported RAM to addressed on any 64K byte boundary within the MULTIBUS's 16 Megabyte address space.

EPROM and Static RAM. Two pairs of 28-pin sockets on the M68K10 accept a wide variety of EPROMs and RAMs. The first pair of sockets is used for up to 64K bytes of EPROM-resident code, which must include a small power-up bootstrap program. PROBUG^{TM2} , the interactive debugger/monitor often used with the M68K10, contains the required bootstrap.

The second pair of sockets can be used either for additional EPROMs or for static RAMs. A total of 128K bytes of EPROM can be accommodated; alternatively, 64K bytes of EPROM can be combined with 16K bytes of static RAM. CMOS static RAM can be powered by external batteries, and can be write- protected in case the main power supply is turned off. On-board jumpers allow selection of memory component size and the battery back-up option.

Input/Output

The input/output (I/O) capabilities of the M68K10 include two RS-232C compatible serial channels, twenty-four bits of parallel I/O, three 16-bit counter/timers, and eight software-controllable status LEDs. Two independent iSBX connectors allow one to add extra I/O to a system without adding MULTIBUS I/O boards.

The two RS-232C serial channels are implemented with a Multi-protocol serial controller (MPSC). In addition to standard asynchronous formats, the MPSC also supports IBM Bisync and SDLC formats. A dual baud rate generator may be used to select one of 16 standard rates for each channel.

---

[2]PROBUG is trademark of SBE, inc.

Alternatively, one or both channels may be set for a programmable baud rate generated by one of the counter/timers.

Both channels A and B include all the handshake signals required by terminals, printers, and modems. A 50-pin connector at the top of the board is wired so that two 25-pin D-type connectors may be directly connected via flat-cable.

The 24 parallel I/O lines are divided into three 8-bit ports: port A, port B, and port C. All 8 bits of port A or B are programmed as a group for either input or output; also port A may be programmed as a bi- directional I/O bus. The 8 bits of port C may be programmed in groups of 4 bits as either input or output; or some of them may be programmed as handshake lines for ports A and B.

The triple counter/timer can be configured to generate a programmable baud rate, time-of-day clock interrupt, external timing signals, etc. Eight programmable LEDs provide a simple operator interface for indicating board or system status.

Each of the M68K10's two iSBX connectors supports an industry-standard I/O expansion interface. Numerous manufacturers provide I/O modules comparible with this interface. These modules may be used to expand and serial I/O, or to add I/O functions such as an IEEE-488 interface, Bubble Memory, a SASI [3] (Shugart Associates System Interface) disk interface, time-of-day clock/calendar, or analog I/O.

MUTIBUS Interface

The M68K10's MULTIBUS interface supports operation in a multi-master environment. Either daisy-chained serial or parallel bus arbitration may be

[3]SASI is a trademark of Shugart Associates.

selected. All 24 address line of the 68000 are brought to the MULTIBUS. 128K bytes of the 68000's total address space is used for on-board I/O (64K bytes) and static RAM/EPROM (64K bytes). Additionally, the M68K10 can have up to 1 Megabyte of dynamic RAM. This leaves at least 14.75 Megabytes of accessible MULTIBUS memory. Since on-board I/O does not occupy any of the MULTIBUS's 64K byte I/O range, the full 64K bytes is available.

The 68000 stores the two bytes within a 16-bit word in the opposite order form the MULTIBUS standard. To compensate for this, the M68K10 automatically swaps bytes when doing word accesses to or from MULTIBUS memory. On byte accesses, it also reverses the meaning of the low-order address bit so that bytes as well as words are accessed correctly. The byte swap logic also functions for slave accesses, delivering data in the order that MULTUBUS requires.

Interrupts

The M68K10 supports seven levels of automatically vectored interrupts. Six of these levels may be driven by either on-board sources or the interrupt request pins of the MULTIBUS; or externally via the J2 connector. The remaining seventh level is non-maskable and may be driven by an external switch.

On-Board Addressing

The M68K10's on-board dynamic RAM begins at address $000000, so MULTIBUS memory cannot begin until after the location at which on-board RAM ends. For example, with 256K bytes of on-board dynamic RAM, the first memory location accessible on the MULTIBUS is $40000.

MULTIBUS memory extends from the end of on-board between memory to location $FBFFFF. The 28-pin sockets occupy the range between $FC0000 and $FCFFFF. On-board I/O occupies the range between $FD0000 and $FDFFFF, and MULTIBUS I/O begins at $FE0000. Address $FF0000 up is unused unless 32K-byte EPROM are installed [25].

4.2.1.2 SBEX-SASI Multimodule Interface Board

The SBEX-SASI Mutimodule Interface board allows connection of a wide variety of controllers for floppy disks, Winchester hard disks, SMD hard disks and tape drives. The board provides a general-purpose interface between a host system containing a Multimodule interface connector, and controllers that are compatible with the industry-standard SASI (Shugart Associates System Interface) bus.
The SBEX-SASI Multimodule Interface board includes these features:

1. Eliminates Multibus system bus latency and increase system throughput.

2. Requires only a single width Multimodule connector on the host system board.

3. Requires only +5V power from processor board.

4. Supports DMA or programmed data transfer.

5. Automatic handshake and selection logic reduces software overhead and increases data transfer rates [26].

4.2.1.3 ModulasTen^{TM} PROBUG^{TM}

PROBUG is a professional interactive debugger for testing software written for the 68000 MPU. Designed to facilitate debugging on the SBE ModulasTen line of single-board computers, PROBUG can be used on other 68000-based computers as well. Offering a wide variety of convenient, easy-to-learn commands and a comprehensive set of error messages, PROBUG helps programmers to identify the most common, as well as the most elusive, programming errors.

PROBUG provides commands to:

- Print, inspect, and alter memory
- Print and alter registers
- Copy and fill memory
- Search memory for a pattern
- Assemble and disassemble instructions
- Download programs from, and write programs to, a host computer or other external device
- Boot up disk operating system

In addition, there are PROBUG commands that allow you to control execution of your program. With these program execution commands, you can:

- Breakpoint through your program with optional iteration count
- Trace through RAM or ROM
- Halt program execution on memory change [27].

4.2.2 Data Translation's IAP Boards

The DTFIRM firmware package is designed to provide users of Data Translation's MIDAX [4] intelligent I/O subsystem with a powerful integrated software environment. DTFIRM contains those parts of the software which are common in most applications, thus easing target system software development [28].

DTFRIM is contained in 8K byte of 2732 type EPROM, and is physically located on the DT3752 board. It can be conceptually divided into the following parts:

1. System Subroutine Library

   The major portion of DTFRIM is devoted to a large library of subroutines which provide a multiplicity of useful functions. This is the primary interface between the MIDAX hardware and the user's software. All I/O functions, as well as many other types of constructs, are made available in the form of easily accessible subroutines.

2. Multitasking Executive

   This provides the ability to write programs as groups of tasks that run concurrently. A large number of applications can be broken up into independent parts that conceptually run in parallel, and the multitasker provides a clean way of implementing these applications. A major feature of the executive is that it is based on message passing, the state of the art in computation theories.

[4] Refers to any of the packaged I/O systems sold by Data Translation under that name which use the IAP as the processing component.

3. Communication servers

An additional feature of DTFRIM is upward compatibility into the multiprocessing environment. MIDAX was designed to be an intelligent subsytem, and though it is sufficiently powerful for many applications, often it is desirable to use multiple MIDAX systems, or for MIDAX to operate as a slave to a larger machine. The major requirement in such parallel processing configurations is the ability to communicate quickly and effectively with the other processors. DTFRIM provides two communications methods, one for MULTIBUS use, and the other for an asynchronous serial line. Both methods allow transmission of data, programs, and commands.

4. Debugger

In order to facilitate software development, DTFIRM contains a powerful debugger which provides the user with the capability to stop program execution at specified points, interactively examine and modify memory, registers and I/O ports, and then proceed. The debugger can also be entered (called) like any normal DTFRIM subruotine, providing a convenient "fatal error catcher" subroutine.

5. Software drivers

All I/O functions as well as many processor functions are supported by high-level drivers. These drivers understand all details required to make the hardware operate correctly. The programmer, thus, need not have intricate knowledge of the MIDAX sytem to utilize the hardware effectively.

4.2.2.1 DT3750 Series

The first and most important board is the Intelligent Analog Peripheral (IAP), which is a member of the DT3750 series. The IAP is numberd DT3752, DT3754, or DT3755, depending on the analog input options selected .

The IAP contains the 8085 microprocessor which is the heart of the MIDAX system. Thus the IAP is also the controlling element of the system; the other boards contain subsystem which are under the 8085's direct control. In addition to the processor, the IAP also contains the following:

1. 8259 Interrupt controller– this subsystem controls 8 interrupt lines for the 8085. Since the processor already has 4 lines built in, there are a total of 12 interrupt lines available on the IAP board. Most of these lines are brought out to wire-wrap posts, where they may be connected to a number of TTL signals.

2. 8251 USART– this subsystem (Universal Synchronous/ Asynchronous Receiver-Transmitter) provides MIDAX with a single serial I/O channel. This channel is most often used in the asynchronous mode for driving terminals, printers, or talking to other computers. The I/O channel is brought out to an edge connector on the DT3752 board, connector J2.

3. 8253 Programmable Interval Timer– this timer subsystem contains three programmable timers, which provides MIDAX with the following signals: a baud rate clock for the 8251 serial line described above; a time base for the A/D converter subsystem; and a real-time clock for the DTFRIM executive program. None of these timers are directly avail-

able to the MIDAX programer; however, there are methods for programming them. The timer values may be changed by the subroutines CBAUD, SETADR, and the variable RTCNT respectively.

4. A/D Converter Subsystem– this subsystem interfaces to data translation's wide range of A/D converter modules, providing the ability to perform analog input conversions efficiently. The subsystem contains a direct memory access (DMA) device that allows conversions to take place with minimal processor intervention. The IAP can support 16 single-ended (SE) or 8 differential (DI) inputs by itself, but can be expanded to 64SE/32DI using the DT3772 [29].

4.2.2.2 DT3762 Series

The DT3762 Multifunction Expander Board is present in all MIDAX configurations, and greatly expands the capabilities of the system. The subsystems available on the DT3762 are as follows:

1. D/A Converter Subsystem – this is a four-channel analog output converter, which can be operated directly under the control of DTFRM. The channels on this board are numbered from 0 to 3.

2. 2651 Programmable Communication Interface – this is a second serial I/O channel, implemented using different hardware than the one on the IAP. When using DTFIRM, the hardware differences are completely transparent, allowing either of the two channels to be used with equal facility.

3. 8255A Programmable Peripheral Interface – this subsystem provides the MIDAX user with 48 lines of TTL level digital I/O. These lines

can be programmed in groups of 8 for either input or output. Note that the lines must be programmed in software and that wire wrap jumper must be set to correspond with the software as well.

4. 9513 Counter/Timer subsystem – this powerful subsystem contains five user-programmable 16-bit timers, plus a crystal-controlled timebase. These timers can perform many interesting timing, counting, and signal generation functions with TTL signals. DTFIRM provides a simple human-oriented interface to the subsystem [30].

4.2.3 The Dynamics Research Corporation (DRC) Model 39 Shaft Encoder

The following information is obtained from a brochure for the Model 29 provided by the manufacturer. According to the manufacturer the Model 39 which is used in our system has the same specifications.

This DRC's new model shaft encoder employs optical techniques to convert rotary motion into electrical waveforms that are easily transformed into uniform and precisely spaced digital pulses. This high-accuracy, high-speed digitizer can reliably encode shaft position or rate, as well as provide direction sensing.

In terms of high accuracy, high speed, low noise, low wear, low friction, low inertia, and lack of ambiguity, for example, no other type of encoder can match the optical type. And this DRC's model utilizes this concept. Photosensors view large areas of light (not just a thin line) and produce high, reliable signal outputs, with derated lamps run at low excitation levels for long life. And because lights and sensors operate as push-pull pairs, this model is not affected by normal changes in excitation voltage, lamp intensity,

ambient temperatures, mechanical misalignments, or bearing wear. The use of optical techniques in the Model 29 also eliminates the problems of sliding contacts (friction, bounce, and wear). The input shaft has low inertia and low friction, and peak rotational speed up to 5000 rmp can be handled. There is a choice of three output configuration... sine waves or square wave directly from the encoder, or pulses using the LM-1 series external logic electronics. The specifications of the encoder can be summarized as follows:

- 360 cycle per shaft revolution.
- 4 pulse counts per cycle.
- Has zero reference.
- Power supply = 5 Volt.
- Has a shaft seal.
- Side connector location.

The pin assignments of the encoder are:

- A : Output pulses for CW rotation.
- C : Output pulses for CCW rotation.
- E : Zero reference (ZR) channel.
- F : Zero reference ($\overline{ZR}$) channel.
- I : +5 VDC.
- J : GND.

4.3 Control Alogrithms

4.3.1 Control Alogrithm for Elbow and Shoulder Joints

The elbow and shoulder joints of the robot arm are controlled using analog control method. The systems used for both joints are identical, except some components have different values. Schematic diagram of the control system which consists of velocity and positional servocontrol loops is shown in Figure 4.2 [19].

Initially, the arm is at complete rest, and it will start moving after an input command signal is given. Since the arm is a long way from the desired setpoint immediately after the input command signal is given, the voltage corresponding to the position error signal is large enough to activate one of the relays, depending on the desired direction of the movement. The relay closes the path which applies a constant signal of ± 15 volts through a potentiometer to the input of the servoamplifier that drives the motor. The potentiometer would allow one to change the speed at which the robot arm will travel to the desired setpoint. Activating the relay takes the position error signal out of the closed-loop and puts the arm under the velocity control loop.

At the end of each movement, we want to stop the arm as close as possible to the desired setpoint, and therefore are interested in the position control loop. When the arm is close to the desired setpoint the voltage corresponding to the position error signal is not large enough to keep the relay activated; therefore, the 15 volts signal is removed and the error signal is fed to the servo amplifier to drive the motor. The arm will move until the position error is zero.

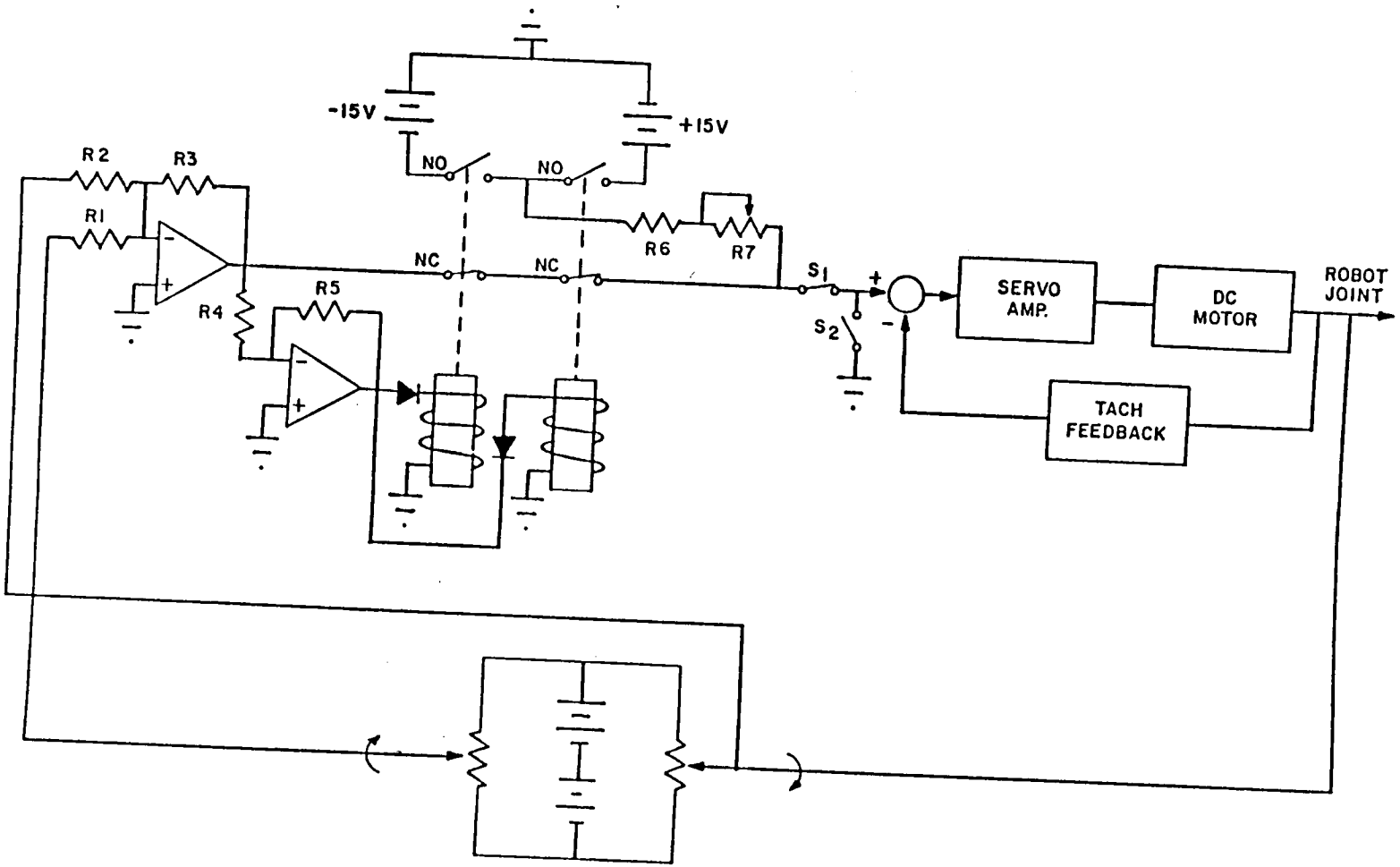


Figure 4.2: Positional and velocity loop used in elbow and shoulder joints

Since changing the potentiometer that is used to give the input command signals from minimum to its maximum value will cause the arm to rotate more than 90 degrees at its joint, mechanical stops are used to stop the arm from rotating more than 90 degrees. It is also desired to stop the motor's shaft from rotating when the arm reaches these mechanical stops. This is achieved by using the circuit diagram shown in Figure 4.3. SW_1 and SW_2 are two submini lever switches which are mounted on the arm. When the arm reaches one of the mechanical stops, it will close one of the switches and will move the switch to position B.

Two voltage comparators (LM393) are used to compare the given input command with the two reference voltages. An input signal command equal to the one of these reference voltages will move the arm to the limits of the 90 degree movement. The range of the input command signals are ± 1.75 volts and ± 1.81 volts for upper and lower arm respectively. The circuits used to set the reference voltages are shown in Figure 4.4. When an input command greater than these values is given, the output of one of the comparators (depending on the polarity of the input command) will change its value from -5 volts to +5 volts. A germanium diode is used to limit the transition range of the output voltage from $\approx +0.48$ volts to +5 volts.

In order to stop the motor from rotating, it is required to activate S_2 and deactivate S_1 (see Figure 4.2). S_1 and S_2 are two bilateral CMOS switches. The S_2 will be activated when two conditions are met: (a) One of the mechanical switches is closed (b) One of the comparators has a output of +5 volts. S_2 should remain activated as long as the input command signal is greater than the reference voltage. At this point, the only way to move the arm is to bring the input command signal within the two reference voltages. If this is done, the arm will move in the direction away from the mechani-

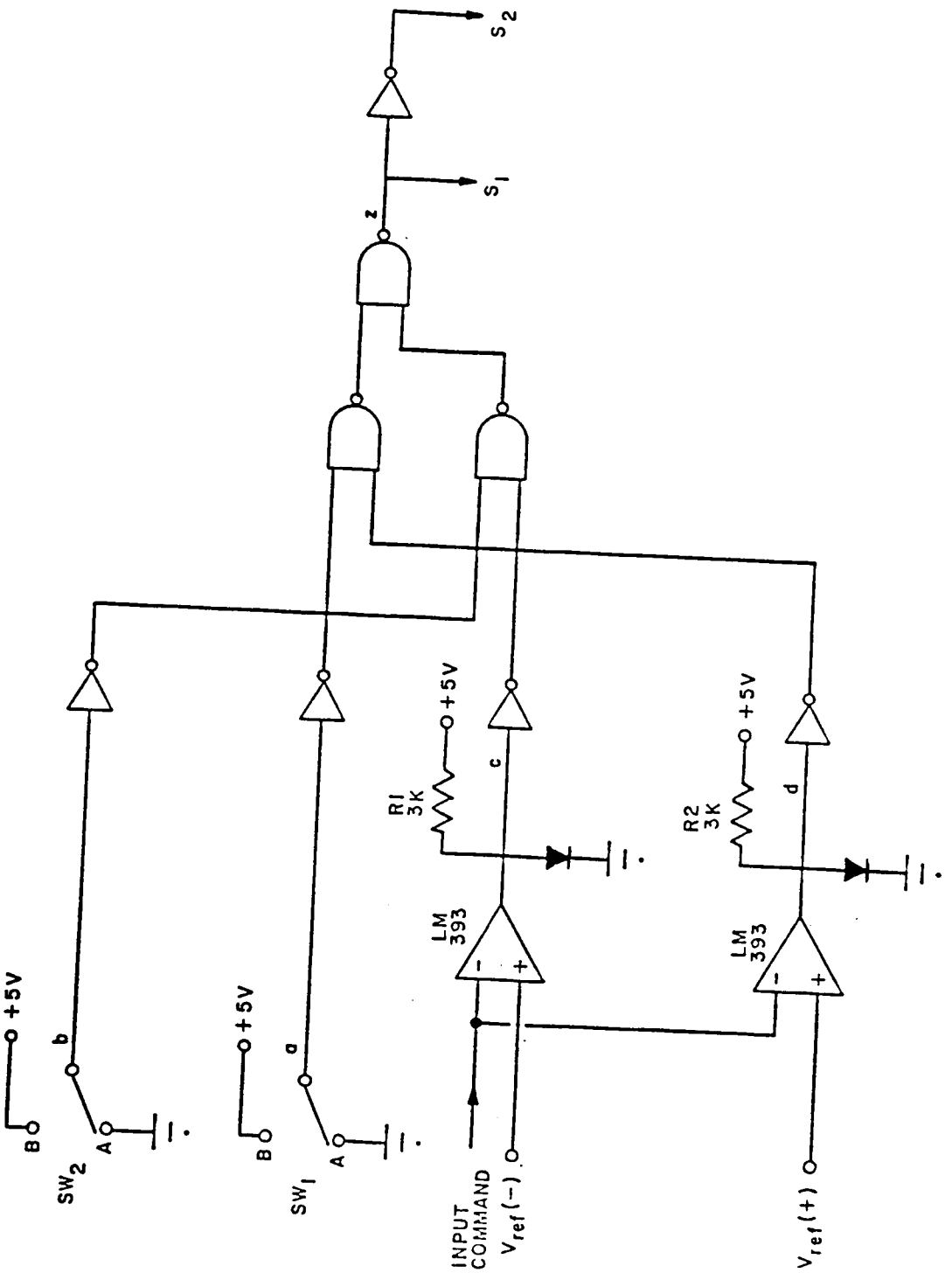


Figure 4.3: Circuit used to activate the CMOS switches.

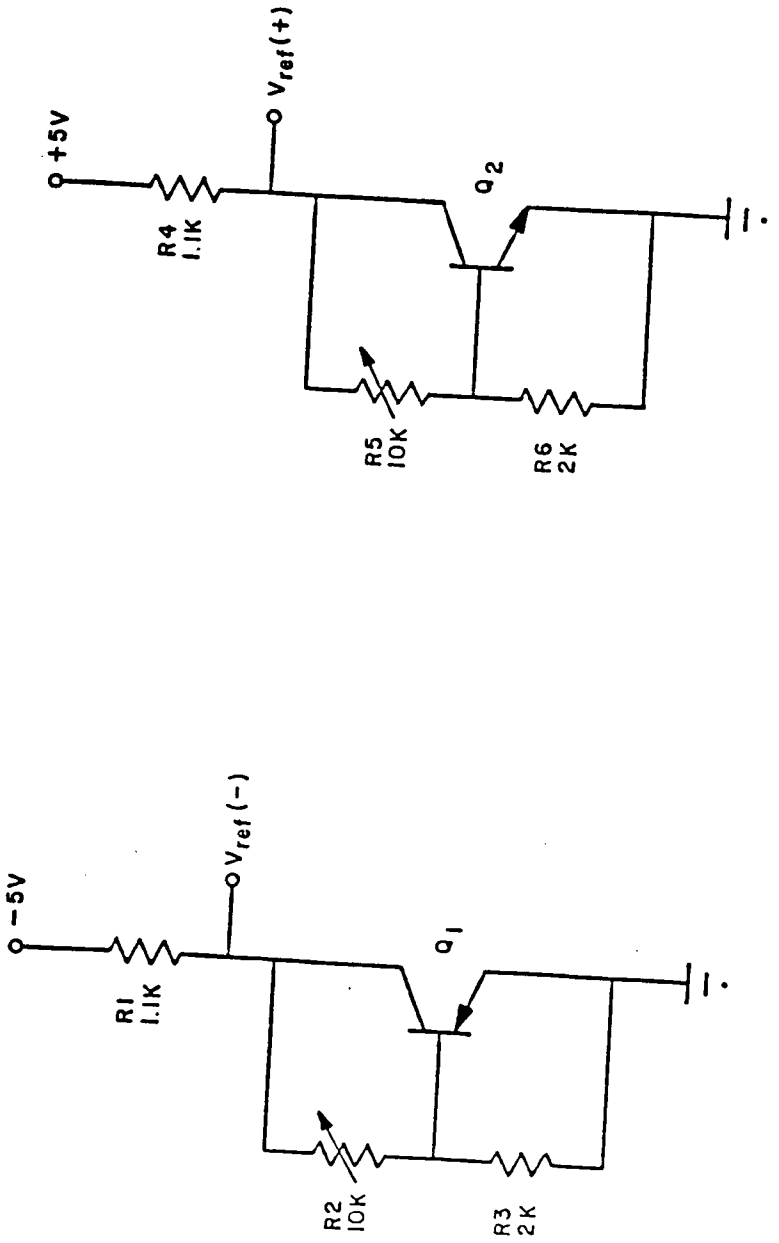


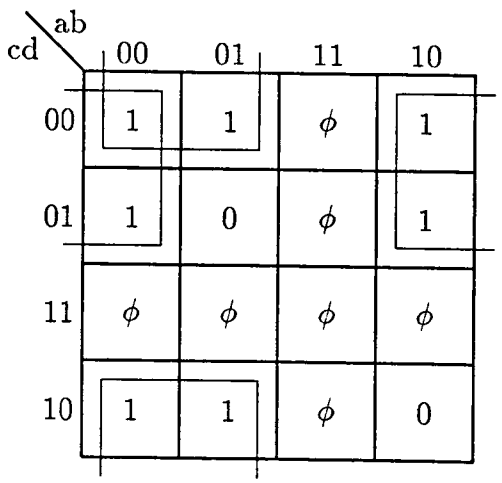
Figure 4.4: Circuits used to set the reference voltages

cal stop. The truth table and the Karnaugh map shown below satisfies the conditions for the correct operation. The equation for the output (Z) is

$$Z = \overline{a}\overline{d} + \overline{b}\overline{c}. \tag{4.1}$$

The two NAND implementation of this equation is used in Figure 4.3.

a	b	c	d	Z
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	ϕ
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	ϕ
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	ϕ
1	1	0	0	ϕ
1	1	0	1	ϕ
1	1	1	1	ϕ



4.3.2 Control Alogrithm For The Base Joint

The block diagram of the control system used for the base axis is shown in Figure 4.5. The supporting software program for this system was devel-

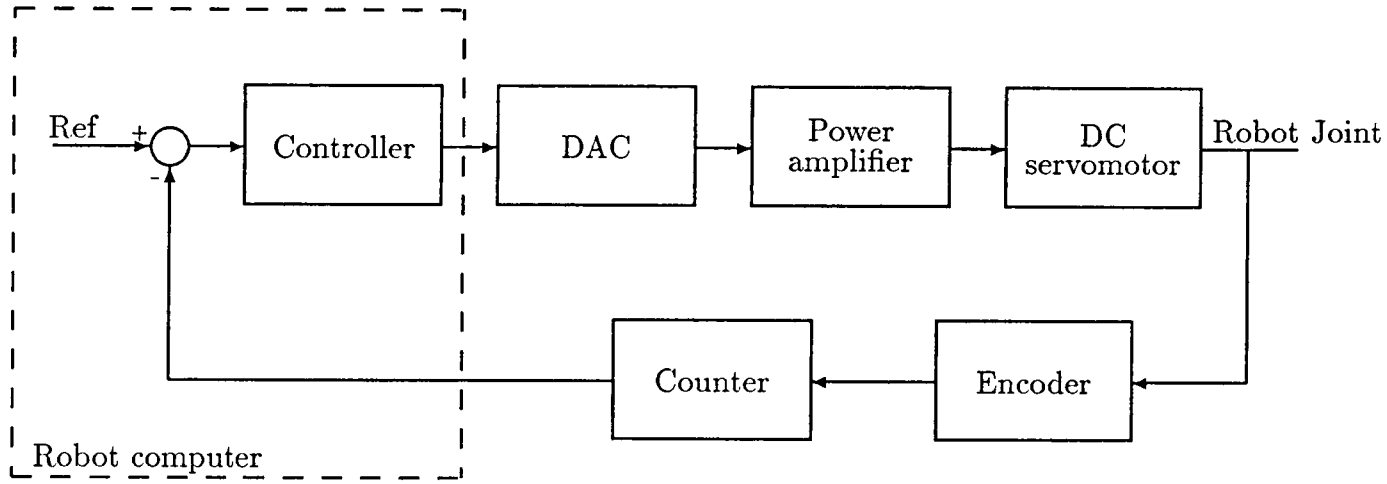


Figure 4.5: Block diagram of the control system used for the base joint.

oped in two portions: (1) develop the control alogrithm routines in assembly language using the Intel 8085 microprocessor and supporting subroutines provided with Data Translation's IAP boards (2) define FORTH words that will be used by the user to give the input commands and words that would make possible to transmit data from SBE 200 system to the IAP board (and vice versa) over the MULTIBUS.

The Main Program

The program consists of four modes. They are the acceleration mode, the maximum velocity mode, the deceleration mode, and the position control mode [20-23]. When moving long distances, the control system utilizes all four modes, but during short distances it may skip the maximum velocity, deceleration, or acceleration mode. Due to the limitation imposed by the resolution of the encoder and the use of the " Lazy Susan " bearing plate, the minimum detectable rotation of the shaft is 0.25 degrees. The resolution of the encoder used in the system is 1440 counts per revolution. Experimentally, it was decided that the travel distances less than 3 degrees would be considered short distances. For such distances, the arm will move with a predetermined velocity while the position of the arm is monitored. As soon as the arm is within two encoder counts, the digital-analog converter (DAC) will output a zero volt to the drive system to stop the arm. However, if the required travel distance is more than 3 degrees, the arm will start to accelerate linearly. While accelerating, the position of the arm continuously is monitored to decide whether to start to declerate or continue the acceleration until the maximum velocity is reached.

The deceleration routine will start under two conditions: (a) the arm still accelerating but it has reached to the midway of the required travel distance

(b) the arm is moving with the maximum velocity and is 10 degrees away from the setpoint. At the end of each movement, the position of the joints (shoulder, elbow, base) will be read and saved in memory location. FORTH words will use this information to display the location of the arm. Figure 4.6 shows the flowchart of the main program used for the base joint.

In order to develop the program, assembly language (Intel 8085) and the system library was utilized. This library provides many useful functions which give the user powerful control over the action of the hardware. To isolate the user from the unnecessary detail of finding out the real memory addresses of the subroutines so that they may be linked into a program, each system subroutine is assigned an " index number ." This number will always be small enough to fit in a single byte, and will uniquely identify a system subroutine, just as an address normally would.

There are two ways in which one can take advantage of the index number to achieve modularity. The first, which is referred to as the system call, involves using the 8085 RST (restart) instructions, requiring 2 bytes to do a call; this is a savings of one byte over the normal call, which is an advantage in addition to the gained modularity. The second method involves the use of a normal call into the "system dispatch table," which takes 3 bytes, but is considerably faster than using the RST instruction. This method is referred to as the dispatch call. Both methods have some time penalty compared to just calling the actual memory address of the system subroutines. Whether such delays are acceptable is a question to be considered by the software implementer; clearly the advantage of independence from DTDIRM's changing addresses is a good one.

System calls are performed by the folloing instructions in 8085 assembly language code:

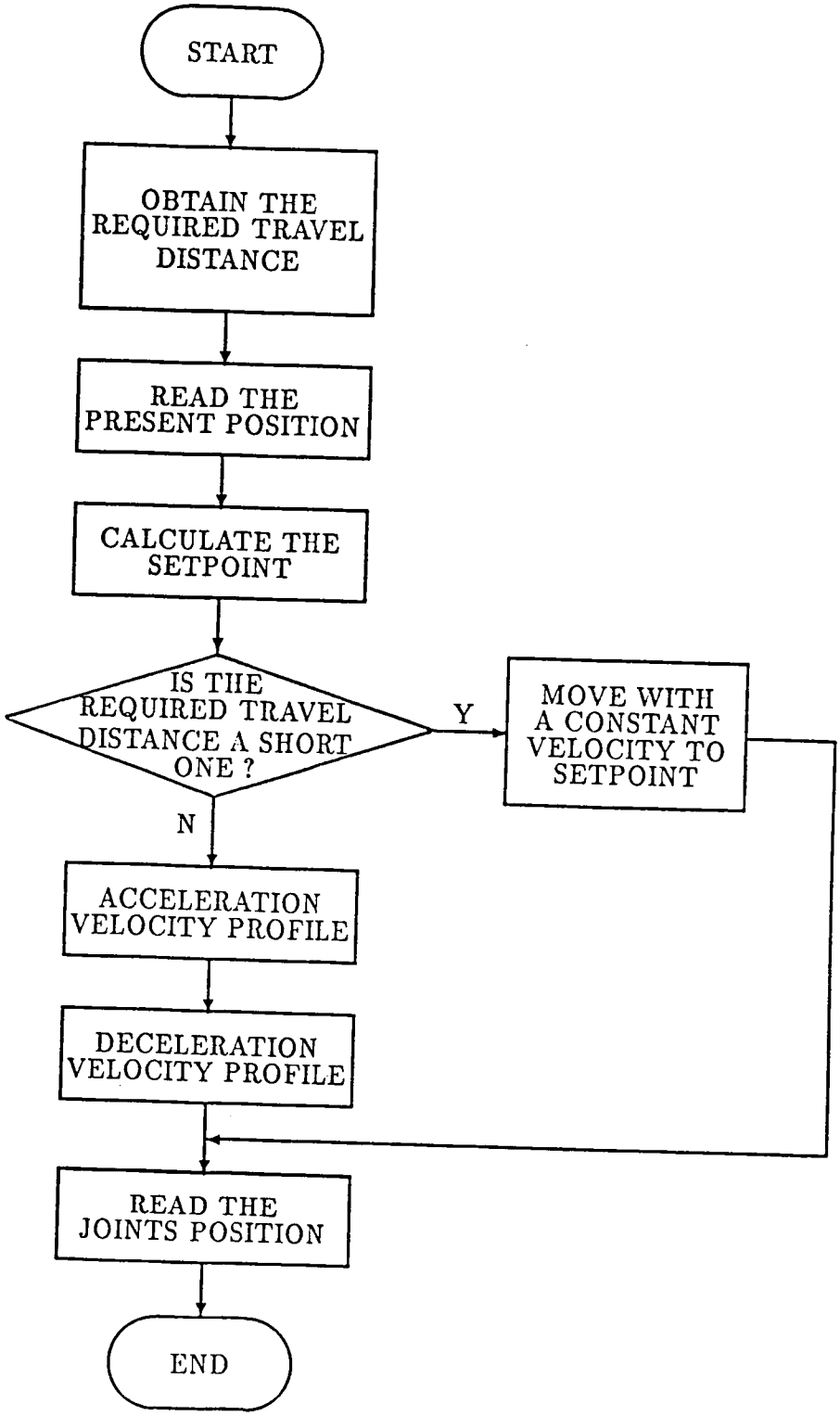


Figure 4.6: The flowchart for the main program

```
RST     2           ; this instruction causes a software trap.
DB      < index >   ; this data byte tells the system what routine
                      is desired.
```

RST 2 is one of the 8085 restart instructions, while DB means " Define Byte ," and merely deposits the specified byte in the given location. This sequence can be automatically generated with macros by the assembler, making it reasonably convenient to use. In fact, the DTFIRM package use system calls internally by means of such a macro that also translates from the mnemonic subroutine names to the index numbers.

Dispatch calls are extremely simple to describe. DTFIRM has a dispatch table which starts at location 40 hexadecimal. The table contains a series of JMP instructions, which each takes 3 bytes of code. To call a system subroutine, use

```
CALL    0040H + (3 * < index >)
```

where $< index >$ is the index number of the subroutine wanted. It is, of course a simple matter to JMP to system subroutines in the same way. Note that the only overhead involved with a dispatch call is the extra time required to execute the JMP instruction in the dispatch table, which is typically 4-5 microseconds.

When describing the properties of a subroutine, several things are of interest to the programmer:

1. **Name** – the mnemonic name of the subroutine. This is given in upper case, underline.

2. **Index Number** – the data byte used in the system call instructions; this number is given in hexadecimal notation.

3. **Input and output** – these are by far the most important aspects of the

subroutine, as they define the conditions under which the subroutine is expected to operate correctly.

4. **Registers preserved** – the second most important thing about a subroutine is which registers that it uses and which it leaves lone. Typically, registers BC, DE, and HL are preserved unless they are listed as outputs, and PSW is never preserved; then this is the default. However, registers may be listed as either "used" or "unused" to explicitly describe their state.

5. **Description** – finally, a brief but complete English statement describing the action of the subroutine is included for each. In some cases this is only a sentence, while in others it can be many paragraphs.

As an example, here is a subroutine description:

DA

Index:	32
Inputs:	channel number in B.
	input data in HL.
Outputs:	illegal channel in carry (true = 0).
Used:	PSW.
Description:	this routine performs a single D/A conversion
	on the channel specified.

For the description of the other subroutines used in the control routines see the user manuals supplied by the Data Translation, Inc. [10-12].

Development of the FORTH words

FORTH words are defined to allow the user to give the system different commands to follow. Also, by using these words inter-computer (SBE 200

and Data Translation IAP boards) communication over the MULTIBUS will be accomplished.

The strategy of the communication between the master (SBE 200) and the slave (IAP boards), can be summarized as follows:

1. Transmit data from the master to the slave (write).

2. Transmit data from the slave to the master (read).

3. Transmit commands from the master to the slave, and reply back from the slave to the master (call).

The basis behind the MULTIBUS scheme for the IAP board is the hardware implementation of the IAP, which is designed to facilitate efficient, powerful modes of communication. There are two major hardware features that aid MULTIBUS communications: (a) the dual port RAM, which allows two-way data flow with no software effort, and (b) an interlocking bidirectional interrupt structure which facilitates control flow and signalling.

The interlocking interrupts depend on a privileged memory location in the dual port RAM; this is location 4000, the lowest RAM address on the IAP board. When this location is written from the MULTIBUS side, the on-board CPU gets an RST 7.5 interrupt; when written from the IAP side the MULTIBUS gets an interrupt. This is the fastest and most effective means of signalling available for MULTIBUS communications.

The only thing that is needed for effective MULTIBUS communications is the ability to transmit commands from the host (master) computer to the MIDAX slave, and to get the appropriate response. An interface which has the advantages of simplicity, effectivness, and generality, is the subroutine call. Thus, the server essentially gives a multibus computer the ability to

call an arbitrary subroutine on the MIDAX, with arbitrary arguments, and to return a reply when finished.

Here is a typical scenario for such communications: The user first installs an interrupt handler on the RST 7.5 interrupt (number 1 to the interrupt system), and enables it. The external host then loads data into the dual port RAM; at some time thereafter, location 4000 (the lowest available location in the dual port RAM) is written, which causes the interrupt to occur. The user's interrupt handler runs, processing the data as required, and then at some point, the MULTIBUS server is called. The server loads the registers from a block of dual port RAM, calls a subroutine address also in RAM, and returns. The returned register values are rewritten in RAM, and the MULTIBUS server returns to the interrupt handler. The handler then writes something back into location 4000, which causes a MULTIBUS interrupt, informing the host processor that the command that the command was processed and the reply is now in RAM.

Figure 4.7 shows the program that makes the MULTIBUS communication for the DTFIRM boards possible. This program makes the boards able to receive commands from the SBE 200 system and send data to the SBE 200 system over the MULTIBUS. When the program is loaded, it will wait indefinitely until a number (0–7) corresponding to a requested routine in the jump table (JMPTBL) is written to the memory location 4000. After the execution of the requested routine is completed, the content of the memory location 4000 is incremented by one, and routine will go back to wait state. The incrementation of the location 4000 is used by the FORTH words to detect if and when the given command is executed. In conjunction with this program, a code using the M68000 instructions and the FORTH assembler was developed to make possible for the SBE 200 system to communicate over

```
        LHLD    4020        ; 4002 will hold stackbase so interrupt
        SHLD    4002        ; can send message to the main routine.
        MVI     A,MULINT    ; Install interrupt vector for
        LXI     INTCOD      ; MULTIBUS communication.
        CALL    INTSET
        CALL    INTEN       ; Enable interrupt. DTFIRM automatically
                            ; enables whole interrupt system.
LOOP    LXI     B,00
        CALL    RCV         ; Wait for MULTIBUS interrupt.
                            ; This is also the return point for
                            ; all routines.
        LXI     D,JMPTBL    ; Figurcout which routine is request
                            ; and take the JMP.
        LXI     H,4000
        MOW     A,M
        ANI     7
        MOV     B,A
        ADD     B
        ADD     B
        MVI     B,0
        MOV     C,A
        XCHG
        DAD     B
        PCHL
```

Figure 4.7: IAP's MULTIBUS communication routine.

```
JMPTBL  JMP          ; Starting address of the first routine(N=0).
        JMP          ;
        JMP          ;
        JMP          ;
        JMP          ;
        JMP          ;
        JMP          ;
        JMP          ; Starting address of the last routine(N=7).

NOTYET: LXI  H,4000  ; JMP to NOTYET if code is not
                     ; implemented yet.
        MVI  A,04    ; 04 is complete code.
        MOV  M,A
        JMP  LOOP

        SHLD 4004
        LHLD 4002
        CALL NSEND
        LHLD 4004
        POP  PSW
        EI
        RET
```

Figure 4.7: continued.

the MULTIBUS with the DTFIRM. This code is in one of the FORTH available blocks and is given in Figure 4.8. When this block is loaded to the system, all the routines developed for the IAP boards (8085 instruction codes) can be loaded to the DTFIRM memory locations by using FORTH store operator (C!). To load a routine to the DTFIRM load a FORTH block which has the following informations:

1. A FORTH word that stores the starting address of the routine to the memory location 80000 and 80001. The memory location 80000 of the SBE 200 system is same as memory location 4000 for the DTFIRM system.

2. FORTH store operation storing the instruction codes for the routine byte by byte to the SBE 200 memory locations corresponding to the DTFIRM memory locations.

The listing of all the defined FORTH words and developed routines is given in the appendix.

4.4 Operating Instructions

In order to move the robot arm around its base axis, follow the following steps:

1. Load block 1024.

2. Execute the word " READY . "

3. Execute the word " 0 DAC ."

```
HEX

VARIABLE INTSTK VARIABLE SPBUFF

ASSEMBLER BEGIN

( SAVE REG'S ) D0 INTSTK AB MOV

( ACKNOWLEGE INTERRUPT ) 80000 AB D0 B. MOV

( INDECATE ) D0 80001 AB B. ADD

( RESTORE REG'S ) INTSTK AB D0 MOV

( END OF INTERRUPT ) RET 64 !
```

Figure 4.8: Code to handle MULTIBUS communications.

4. Turn on the power for the servopower amplifier that drives the motor of the base joint. Then, turn on the switch that connects the DAC to the input of the servo amplifier. This switch is mounted on the chassis of the up-down counter.

5. If it is desired to send the robot to the home position, execute the word " HOME . "

6. Initialize the counter by executing the word " INITIAL ."

7. Execute the word " INT ."

8. At this point, the robot is ready to follow the given commands. To move the robot to a new position, indicate the amount of movement (in degrees) by executing the word " XXX DEGREE "(-XXX for CW and XXX for CCW).

9. Execute the word " GO ."

10. Repeat steps (8) (9) for additional movements.

CHAPTER 5

SUGGESTIONS FOR FURTHER DEVELOPMENT

Extending from the work presented here, areas could be investigated further. The following are a few suggestions that, if implemented, will develop the system to a more complete robot system.

1. Adding a gripper to the arm.

2. Controlling all the joints using analog control algorithms.

3. Extending the digital control algorithm, and putting all the joints under the computer control.

4. Applying various adaptive control methods to design a self-tuning controller which would automatically adjust to changes in the parameters of the controlled robot system.

LIST OF REFERENCES

LIST OF REFERENCES

[1] Malon, Robert, "The Robot Book," Push Pin Press, New York, 1978.

[2] Dwivedi, S. N., "Robotics and Factories of the Future", Springer-Verlag., 1984.

[3] Garoogian, Andrew, "Robotics 1960-1983," CompuBibs, 1984.

[4] Kafrissen, E., Stephans, M., "Industrial Robots and Robotics," Reston Publishing Company, Inc., 1984.

[5] Rehg, J., "Introduction to Robotics," Prentice-Hall, Inc., 1985.

[6] Snyder, W. E., "Industrial Robots," Prentice-Hall, Inc., 1985.

[7] Heath, L., "Fundamentals of Robotics, Theory and Applications," Reston Publishing Company, Inc., 1985.

[8] Cunningham, J. E., Horn, D. T., "Handbook of Remote Control & Automation Techniques," TAB Book Inc., 1984.

[9] Electro-Craft Cop., "DC Motors, Speed Controls, and Servo Systems," 1980.

[10] Kuo, B. C., "Automatic Control Systems," Prentice-Hall, Inc., 1980.

[11] Kuo, B. C., "Theory and Applications of Step Motors," West Publishing Co., 1973.

[12] Kenjo, Takashi, "Stepping Motors and Their Microprocessor Control," Clarendon Press, Oxford, 1974.

[13] Gilder, J. H., "Focus on Stepping Motors," Electronic Design, October 25, 1977.

[14] Rusinoff, S. E., "Automation in Practice," American Technical, 1957.

[15] Hurteau, R., De Santis, R. M.,"Real-Time Computer Control of a DC Servomotor," IEEE J. Decision and Control, Vol.2, 1982.

[16] Shah, P. S., DiMeo, F. N., "Complete Control of Permanent Magnet DC Motor Using a Microprocessor," Motorcon Proceedings, March 1982.

[17] Juds, Scotte, "Principles and Techniques of Digital Velocity and Position Servocontrol Using Microprocessor Technology," Motorcon Proceeding, March 1982.

[18] Koren, Yoram, "Torque and Speed Control of DC-Servomotors for Robots," Annals of the CIRP, Vol.33, No.1, 1984.

[19] Yee, E. A., "Control System Analysis and Design of a Surface Finish Inspection Machine," A Master's Thesis, The University of Tennessee, Knoxville, 1968.

[20] Ohel, Isaac, "Emulator Helps Speed Motor Control System Design," Control Engineering, May 1983.

[21] Guaen, Kim, "Designing a DC Servo Position Control Using a Microcomputer," Control Engineering, July 1983.

[22] Koren, Y., "Design of Computer Control for Manufacturing Systems," Journal of Engineering for Industry," Vol.101, August 1979.

[23] Raibert, M. H., Craig, J. J., "Hybrid Position/Force Control of Manipulators," Journal of Dynamic Systems, Measurement, and Control, Vol.102, June 1981.

[24] SBE 200 Computer System Reference Manual, SBE, Inc., 1984.

[25] ModulasTen^{TM}M68K10^{TM} 68000 MULTIBUS Single Board Computer Reference Manual, SBE, Inc., 1984.

[26] ModulasTen^{TM}PROBUG^{TM} 68000 Debuger Version 2.1 Reference Manual, SBE, Inc., 1984.

[27] SBEX-SASI^{TM} Multimodule Interface Board Hardware Reference Manual, SBE, Inc., 1984.

[28] User Manual for DTFIRM, Data Tanslation, Inc., October 1981.

[29] User Manual for DT3752, DT3754, Data Translation, Inc., October 1981.

[30] User Manual for DT3756, Data Translation, Inc., October 1981.

APPENDIX

```
1000 LIST

                THIS ROUTINE WILL INITIALIZE THE COUNTER


1026 LOAD
HEX : INITIAL      72 80001 C!     04 80000 C! ;

 D7  83204  C!  35  83205  C!  06  83206  C!  03  83207  C!
 D7  83208  C!  37  83209  C!  06  8320A  C!  04  8320B  C!
 D7  8320C  C!  37  8320D  C!  06  8320E  C!  05  8320F  C!
 D7  83210  C!  37  83211  C!  01  83212  C!  00  83213  C!
 19  83214  C!  D7  83215  C!  3B  83216  C!  01  83217  C!
 80  83218  C!  19  83219  C!  D7  8321A  C!  3B  8321B  C!
 01  8321C  C!  80  8321D  C!  18  8321E  C!  D7  8321F  C!
 3B  83220  C!  01  83221  C!  00  83222  C!  18  83223  C!
 D7  83224  C!  3B  83225  C!  06  83226  C!  00  83227  C!
 21  83228  C!  00  83229  C!  00  8322A  C!  D7  8322B  C!
 32  8322C  C!  C9  8322D  C!

DECIMAL
```

1001 LIST

THIS ROUTINE WILL SEND THE ROBOT TO HOME POSITION

1026 LOAD

HEX : HHHH 72 80001 C! 40 80000 C! ;

06	83240	C!	01	83241	C!	21	83242	C!	00	83243	C!
04	83244	C!	D7	83245	C!	32	83246	C!	06	83247	C!
02	83248	C!	21	83249	C!	0C	8324A	C!	00	8324B	C!
D7	8324C	C!	32	8324D	C!	32	8324E	C!	00	8324F	C!
21	83250	C!	30	83251	C!	FF	83252	C!	D7	83253	C!
32	83254	C!	06	83255	C!	02	83256	C!	D7	83257	C!
2F	83258	C!	11	83259	C!	00	8325A	C!	01	8325B	C!
D7	8325C	C!	55	8325D	C!	D2	8325E	C!	55	8325F	C!
72	83260	C!	CD	83261	C!	04	83262	C!	72	83263	C!
21	83264	C!	00	83265	C!	40	83266	C!	3E	83267	C!
03	83268	C!	77	83269	C!	C3	8326A	C!	1F	8326B	C!
76	8326C	C!									

DECIMAL

1002 LIST

THIS ROUTINE WILL READ THE ENCODER

1026 LOAD

HEX : POSITION 72 80001 C! 84 80000 C! ;

```
01  83284 C!  80  83285 C!  18  83286 C!  D7  83287 C!
3B  83288 C!  06  83289 C!  00  8328A C!  D7  8328B C!
38  8328C C!  69  8328D C!  06  8328E C!  01  8328F C!
D7  83290 C!  38  83291 C!  61  83292 C!  01  83293 C!
00  83294 C!  18  83295 C!  D7  83296 C!  3B  83297 C!
22  83298 C!  10  83299 C!  71  8329A C!  C9  8329B C!
```

DECIMAL

```
1003 LIST

        THIS ROUTINE WILL TAKE 2'S COMPLEMENT OF NEG #S
                INCLUDING THE SIGN BIT
        EXAMPLES: 8001 = 7FFF,0001 = 0001


1026 LOAD
HEX : SIGN2'S      73 80001 C!      04 80000 C! ;

 7C  83304  C!  17  83305  C!  D0  83306  C!  D7  83307  C!
 61  83308  C!  C9  83309  C!

DECIMAL
```

```
1004 LIST

    THIS ROUTINE WILL PUT OUT VOLTS TO THE DAC THAT
                DRIVES THE DC MOTOR


1026 LOAD

: VOLT        2047000 × 9995 / ;
: BYTES       DUP 256 / SWAP DUP 256 / 256 × - ;
HEX
: DAC       VOLT BYTES 80008 C! 80009 C! 00 80005 C! 00 80002 C!
        73 80001 C! 20 80000 C! ;

D7 83320 C! 32 83321 C! C9 83322 C!

DECIMAL
```

1005 LIST

THIS ROUTINE FINDS THE RIGHT OUT PUT VOLTAGE

1026 LOAD

HEX : VOUT 73 80001 C! 84 80000 C! ;

```
E5 83384 C! 2A 83385 C! 04 83386 C! 71 83387 C!
7C 83388 C! E6 83389 C! 80 8338A C! 17 8338B C!
D2 8338C C! 91 8338D C! 73 8338E C! E1 8338F C!
C9 83390 C! E1 83391 C! D7 83392 C! 61 83393 C!
C9 83394 C!
```

DECIMAL

1006 LIST

THIS ROUTINE WILL CHECK THE MSB, IF 0 TAKE 2'S OF THE NUMBER, IF IT IS 1 CHANGE IT TO 0 AND PASS THE NUMBER

EXAMPLES: 0020 = FFE0, 8020 = 0020

1026 LOAD

```
HEX : GETSIGN      74 80001 C!      04 80000 C! ;

7C  83404  C!  F5  83405  C!  E6  83406  C!  80  83407  C!
17  83408  C!  D2  83409  C!  11  8340A  C!  74  8340B  C!
F1  8340C  C!  E6  8340D  C!  7F  8340E  C!  67  8340F  C!
C9  83410  C!  D7  83411  C!  61  83412  C!  F1  83413  C!
C9  83414  C!

DECIMAL
```

1007 LIST

ACCELERATION ROUTINE

1026 LOAD

HEX : SPEEDUP 74 80001 C! 84 80000 C! ;

2A	83484	C!	0A	83485	C!	71	83486	C!	E5	83487	C!
CD	83488	C!	84	83489	C!	72	8348A	C!	CD	8348B	C!
04	8348C	C!	74	8348D	C!	EB	8348E	C!	E1	8348F	C!
D7	83490	C!	54	83491	C!	E5	83492	C!	2A	83493	C!
04	83494	C!	71	83495	C!	CD	83496	C!	04	83497	C!
73	83498	C!	EB	83499	C!	21	8349A	C!	02	8349B	C!
00	8349C	C!	EB	8349D	C!	D7	8349E	C!	53	8349F	C!
22	834A0	C!	06	834A1	C!	71	834A2	C!	E1	834A3	C!
EB	834A4	C!	2A	834A5	C!	06	834A6	C!	71	834A7	C!
D7	834A8	C!	55	834A9	C!	D0	834AA	C!	21	834AB	C!
00	834AC	C!	00	834AD	C!	11	834AE	C!	06	834AF	C!

DECIMAL

1008 LOAD

1009 LIST

CONTINUATION OF ACCELERATION ROUTINE

HEX

00	834B0	C!	01	834B1	C!	05	834B2	C!	00	834B3	C!
D7	834B4	C!	13	834B5	C!	2A	834B6	C!	08	834B7	C!
71	834B8	C!	19	834B9	C!	22	834BA	C!	08	834BB	C!
71	834BC	C!	06	834BD	C!	00	834BE	C!	CD	834BF	C!
84	834C0	C!	73	834C1	C!	22	834C2	C!	0C	834C3	C!
71	834C4	C!	D7	834C5	C!	32	834C6	C!	11	834C7	C!
20	834C8	C!	01	834C9	C!	CD	834CA	C!	04	834CB	C!
73	834CC	C!	D7	834CD	C!	55	834CE	C!	DA	834CF	C!
84	834D0	C!	74	834D1	C!	C9	834D2	C!			

DECIMAL

1010 LIST

DECELERATION ROUTINE

1026 LOAD

HEX : SLOW 75 80001 C! 04 80000 C! ;

CD	83504	C!	84	83505	C!	72	83506	C!	CD	83507	C!
04	83508	C!	74	83509	C!	E5	8350A	C!	2A	8350B	C!
0A	8350C	C!	71	8350D	C!	EB	8350E	C!	E1	8350F	C!
EB	83510	C!	D7	83511	C!	54	83512	C!	CD	83513	C!
04	83514	C!	73	83515	C!	11	83516	C!	16	83517	C!
00	83518	C!	D7	83519	C!	55	8351A	C!	DA	8351B	C!
42	8351C	C!	75	8351D	C!	2A	8351E	C!	0C	8351F	C!
71	83520	C!	06	83521	C!	00	83522	C!	D7	83523	C!
32	83524	C!	11	83525	C!	10	83526	C!	00	83527	C!
01	83528	C!	15	83529	C!	00	8352A	C!	EB	8352B	C!
CD	8352C	C!	84	8352D	C!	73	8352E	C!	EB	8352F	C!

DECIMAL

1011 LOAD

1011 LIST

CONTINUATION OF THE DECELERATION ROUTINE

HEX

D7	83530	C!	13	83531	C!	D7	83532	C!	54	83533	C!
22	83534	C!	0C	83535	C!	71	83536	C!	CD	83537	C!
04	83538	C!	73	83539	C!	11	8353A	C!	40	8353B	C!
00	8353C	C!	D7	8353D	C!	55	8353E	C!	D2	8353F	C!
04	83540	C!	75	83541	C!	06	83542	C!	00	83543	C!
21	83544	C!	40	83545	C!	00	83546	C!	CD	83547	C!
84	83548	C!	73	83549	C!	D7	8354A	C!	32	8354B	C!
CD	8354C	C!	84	8354D	C!	72	8354E	C!	CD	8354F	C!
04	83550	C!	74	83551	C!	E5	83552	C!	2A	83553	C!
0A	83554	C!	71	83555	C!	EB	83556	C!	E1	83557	C!
EB	83558	C!	D7	83559	C!	54	8355A	C!	CD	8355B	C!
04	8355C	C!	73	8355D	C!	11	8355E	C!	02	8355F	C!
00	83560	C!	D7	83561	C!	55	83562	C!	D2	83563	C!
4C	83564	C!	75	83565	C!	06	83566	C!	00	83567	C!
21	83568	C!	00	83569	C!	00	8356A	C!	D7	8356B	C!
32	8356C	C!	C9	8356D	C!						

DECIMAL

1012 LIST

MAIN ROUTINE

1026 LOAD

1027 LOAD 1028 LOAD 1029 LOAD 1031 LOAD

1032 LOAD 1033 LOAD 1035 LOAD

HEX : RUN 75 80001 C! 84 80000 C! ;

CD	83584	C!	84	83585	C!	72	83586	C!	CD	83587	C!
04	83588	C!	74	83589	C!	EB	8358A	C!	2A	8358B	C!
04	8358C	C!	71	8358D	C!	19	8358E	C!	22	8358F	C!
0A	83590	C!	71	83591	C!	2A	83592	C!	04	83593	C!
71	83594	C!	CD	83595	C!	04	83596	C!	73	83597	C!
EB	83598	C!	21	83599	C!	14	8359A	C!	00	8359S	C!
D7	8359C	C!	55	8359D	C!	DA	8359E	C!	CD	8358F	C!
75	835A0	C!	06	835A1	C!	00	835A2	C!	21	835A3	C!

DECIMAL

1013 LOAD

1013 LIST

CONTINUATION OF MAIN ROUTINE

HEX

40	835A4	C!	00	835A5	C!	CD	835A6	C!	84	835A7	C!
73	835A8	C!	D7	835A9	C!	32	835AA	C!	CD	835AB	C!
84	835AC	C!	72	835AD	C!	CD	835AE	C!	04	835AF	C!
74	835B0	C!	E5	835B1	C!	2A	835B2	C!	0A	835B3	C!
71	835B4	C!	EB	835B5	C!	E1	835B6	C!	EB	835B7	C!
D7	835B8	C!	54	835B9	C!	CD	835BA	C!	04	835BB	C!
73	835BC	C!	11	835BD	C!	02	835BE	C!	00	835BF	C!
D7	835C0	C!	55	835C1	C!	D2	835C2	C!	AB	835C3	C!
75	835C4	C!	06	835C5	C!	00	835C6	C!	21	835C7	C!
00	835C8	C!	00	835C9	C!	D7	835CA	C!	32	835CB	C!
C9	835CC	C!	CD	835CD	C!	84	835CE	C!	74	835CF	C!
2A	835D0	C!	0A	835D1	C!	71	835D2	C!	EB	835D3	C!

DECIMAL

1014 LOAD

1014 LIST

CONTINUATION OF MAIN ROUTINE

HEX

CD	835D4	C!	84	835D5	C!	72	835D6	C!	00	835D7	C!
CD	835D8	C!	04	835D9	C!	74	835DA	C!	D7	835DB	C!
54	835DC	C!	CD	835DD	C!	04	835DE	C!	73	835DF	C!
11	835E0	C!	25	835E1	C!	00	835E2	C!	D7	835E3	C!
55	835E4	C!	D2	835E5	C!	D0	835E6	C!	75	835E7	C!
CD	835E8	C!	04	835E9	C!	75	835EA	C!	CD	835EB	C!
84	835EC	C!	72	835ED	C!	22	835EE	C!	10	835EF	C!
71	835F0	C!	06	835F1	C!	00	835F2	C!	D7	835F3	C!
2F	835F4	C!	22	835F5	C!	12	835F6	C!	71	835F7	C!
06	835F8	C!	03	835F9	C!	D7	835FA	C!	2F	835FB	C!
22	835FC	C!	14	835FD	C!	71	835FE	C!	21	835FF	C!
00	83600	C!	40	83501	C!	3E	83602	C!	01	83603	C!
77	83604	C!	C3	83605	C!	1F	83506	C!	76	83607	C!

DECIMAL

1015 LOAD

1015 LIST

CONTINUTION OF MAIN ROUTINE

HEX

CD	835D4	C!	84	835D5	C!	72	835D6	C!	00	835D7	C!
CD	835D8	C!	04	835D9	C!	74	835DA	C!	D7	835DB	C!
54	835DC	C!	CD	835DD	C!	04	835DE	C!	73	835DF	C!
11	835E0	C!	25	835E1	C!	00	835E2	C!	D7	835E3	C!
55	835E4	C!	D2	835E5	C!	D0	835E6	C!	75	835E7	C!
CD	835E8	C!	04	835E9	C!	75	835EA	C!	CD	835EB	C!
84	835EC	C!	72	835ED	C!	22	835EE	C!	10	835EF	C!
71	835F0	C!	06	835F1	C!	00	835F2	C!	D7	835F3	C!
2F	835F4	C!	22	835F5	C!	12	835F6	C!	71	835F7	C!
06	835F8	C!	03	835F9	C!	D7	835FA	C!	2F	835FB	C!
22	835FC	C!	14	835FD	C!	71	835FE	C!	21	835FF	C!
00	83600	C!	40	83601	C!	3E	83602	C!	01	83603	C!
77	83604	C!	C3	83605	C!	1F	83606	C!	76	83607	C!

DECIMAL

1016 LIST

INTERRUPT ROUTINE

1026 LOAD

HEX : INTERRUPT 76 80001 C! 10 80000 C! ;

2A	83610	C!	20	83611	C!	40	83612	C!	22	83613	C!
02	83614	C!	40	83615	C!	3E	83616	C!	01	83617	C!
21	83618	C!	70	83619	C!	76	8361A	C!	D7	8361B	C!
26	8361C	C!	D7	8361D	C!	27	8361E	C!	01	8361F	C!
00	83620	C!	00	83621	C!	D7	83622	C!	13	83623	C!
11	83624	C!	40	83625	C!	76	83626	C!	21	83627	C!
00	83628	C!	40	83629	C!	7E	8362A	C!	E6	8362B	C!
07	8362C	C!	47	8362D	C!	80	8362E	C!	80	8362F	C!
06	83630	C!	00	83631	C!	4F	83632	C!	EB	83633	C!
09	83634	C!	E9	83635	C!	C3	83640	C!	84	83641	C!
75	83642	C!	C3	83643	C!	84	83644	C!	76	83645	C!
C3	83646	C!	40	83647	C!	72	83648	C!	C3	83649	C!

DECIMAL

1017 LOAD

1017 LIST

CONTINUATION OF INTERRUPT ROUTINE

HEX

60	8364A	C!	76	8364B	C!	C3	8364C	C!	60	8364D	C!
76	8364E	C!	C3	8364F	C!	60	83650	C!	76	83651	C!
C3	83652	C!	60	83653	C!	76	83654	C!	C3	83655	C!
60	83656	C!	76	83657	C!						
21	83660	C!	00	83661	C!	40	83662	C!	3E	83663	C!
04	83664	C!	77	83665	C!	C3	83666	C!	1F	83667	C!
76	83668	C!									
22	83670	C!	04	83671	C!	40	83672	C!	2A	83673	C!
02	83674	C!	40	83675	C!	D7	83676	C!	15	83677	C!
2A	83678	C!	04	83679	C!	40	8367A	C!	F1	8367B	C!
FB	8367C	C!	C9	8367D	C!						

DECIMAL

1018 LIST

THIS ROUTINE WILL READ THE POSITION OF THE ENCODER

1026 LOAD

HEX : READ 76 80001 C! 84 80000 C!;

01	83684	C!	80	83685	C!	18	83686	C!	D7	83687	C!
3B	83688	C!	06	83689	C!	00	8368A	C!	D7	8368B	C!
38	8368C	C!	69	8368D	C!	06	8368E	C!	01	8368F	C!
D7	83690	C!	38	83691	C!	61	83692	C!	01	83693	C!
00	83694	C!	18	83695	C!	D7	83696	C!	3B	83697	C!
22	83698	C!	10	83699	C!	71	8369A	C!	06	8369B	C!
00	8369C	C!	D7	8369D	C!	2F	8369E	C!	22	8369F	C!
12	836A0	C!	71	836A1	C!	06	836A2	C!	01	836A3	C!
D7	836A4	C!	2F	836A5	C!	22	836A6	C!	14	836A7	C!
71	836A8	C!	00	836A9	C!	21	836AA	C!	00	836AB	C!
40	836AC	C!	3E	836AD	C!	02	836AE	C!	77	836AF	C!
C3	836B0	C!	1F	836B1	C!	76	836B2	C!			

DECIMAL

```
1019 LIST

   WORDS TO BE USED FOR OPERATION OF THE BASE JOINT


      : COUNT       4 × HEX ;
HEX : CLEARM        83106 F 0 FILL ;
HEX : DEGREE ( # OF COUNTS IN 7104,7105)
          CLEARM COUNTS DUP 100 / SWAP DUP
          100 / 100 × - 83104 C! DECIMAL ;
HEX : XX 83110 C@ 83111 C 100 × + ;
HEX : ENCODER XX DUP 8000 AND 0 > IF 7FFF AND 4 / DECIMAL
          ." + ". ." DEGREES " ELSE 4 / DECIMAL
          NEGATE . ." DEGREES " THEN ; DECIMAL
    : BASE?       ." BASE IS LOCATED AT " ENCIDER CR ;
```

1020 LIST

WORDS TO BE USED TO FIND THE POSITION OF
THE ELBOW JOINT
(2.44 VOLT = 45 DEG, 2440 = 988 HEX, 45 = 2D HEX)

```
      : ARM DECIMAL 1000 × 2047 / ;
HEX : AAA 83112 C F0 AND 10 / 83113 C 10 × + HEX ;
HEX : POT1 AAA DUP 0800 AND 0 > IF FFFFF000 OR NEGATE
          ARM 2D × 988 / NEGATE . ." DEGREES " ELSE ARM
          2D × 988 / ." + " ." DEGREES " THEN ;
    : ARM ." ARM IS LOCATED AT " POT1 CR ;
```

1021 LIST

WORDS TO BE USED TO FIND THE POSITION OF
THE SHOULDER JOINT

```
    : SHO       DECIMAL 10000 × 2047 / ;
HEX : SSS 83114 C F0 AND 10 / 83115 C 10 × + HEX ;
HEX : POT2 SSS DUP 0800 AND 0 > IF FFFFF000 OR NEGATE SHO
        2D × 384 / ." + " . ." DEGREES " ELSE SHO 2D × 384 / ." - ".
        ." DEGREES " THEN ;
    : SHOL? ." SHOULDER IS LOCATED AT " POT2 CR ;
```

1022 LIST

WORDS TO BE USED TO MOVE THE ARM AROUND THE BASE

AND

SEND THE ROBOT TO THE HOME POSITION

```
HEX : GO 00 80000 C! BEGIN 80000 C 00 > UNTIL CR
        ." JOB IS DONE " CR BASE? ARM? SHOL? ;
HEX : HOME 02 80000 C! BEGIN 80000 C 02 > UNTIL CR
        ." THE ROBOT IS NOW AT HOME POSITION . " CR ;

DECIMAL
```