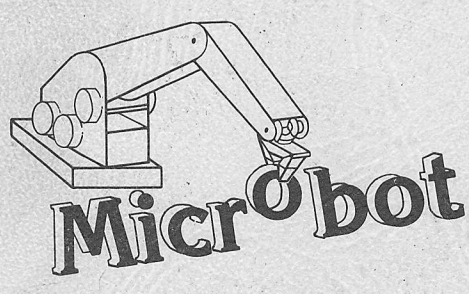
# MiniMover-5

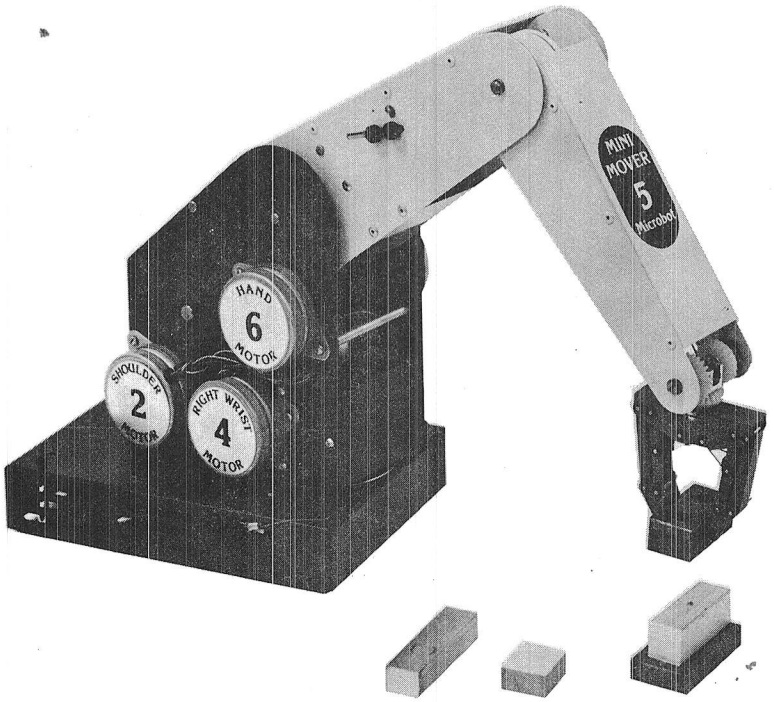
# USER REFERENCE AND APPLICATIONS MANUAL




1259 El Camino Real, Suite 200
Menlo Park, CA 94025
(415) 326-6997

# Operation of the 5-AXIS TABLE-TOP MANIPULATOR

## Models MIM-5 and 5-P




**Microbot, Inc.**
1259 El Camino Real, Suite 200,
Menlo Park, CA 94025

CONTENTS

ILLUSTRATIONS

TABLES

# I INTRODUCTION

The MiniMover-5 manipulator is a computer-controlled, 5-jointed mechanical arm which has been designed to provide an unusual combination of dexterity and low cost. A complete system using the MiniMover-5 has been developed for Radio Shack's TRS-80™* microcomputer (Level II required): The arm need only be plugged into the computer and the ARMBASIC cassette loaded. With only slight modification, operation from other computers is possible from a parallel port.

The MiniMover-5 manipulator can be used for many educational, practical, and enjoyable applications which are limited only by the imagination of the user. Among these applications are:

* Education in the areas of coordinate transformations, matrix algebra, manipulation algorithms, etc.
* Artificial Intelligence and Robotics which use computer vision, and/or involve planning strategies, language development, etc.
* Industrial Automation for light assembly, kitting of parts, palletizing, operation of equipment and tools, etc.
* Playing Games such as Chess, Tic-Tac-Toe, etc.
* Direct Art with felt-tip pens, paints, etc.
* Computer Construction with Lincoln Logs, blocks, Legos, etc.

Previously, work with computer-controlled manipulators was limited to laboratories at a few research-oriented universities in the USA and overseas. Principal advances in this field have come from artificial intelligence laboratories, particularly at Stanford University and The Massachusetts Institute of Technology. Work has concentrated on

---

* TRS-80 is a registered trade mark of Tandy Corporation, Fort Worth, Texas. TRS-80 is used in this manual only to refer to the operation of the Microbot arm from the Tandy computer. Microbot, Inc. is not affiliated with Tandy Corporation.

languages for manipulation, planning manipulation strategies, coordinate transformations, sensors, and manipulator kinematics. Much of this work is deeply mathematical and has been performed on highly sophisticated computing systems.

Industry has taken some of these results and applied them to factory situations to the extent that thousands of robot arms are used to manufacture products we use daily. These industrial robots are used to spot weld automobile bodies, feed punch-presses, spray paint articles, empty injection molding machines, and perform many other factory jobs [19]. Principal manufacturers of these machines include Unimation and Cincinnati Milacron in the USA and ASEA in Sweden. The price range, from $35,000 to $120,000, hinders robot experimentation by individuals and schools as do the carefully guarded, proprietary, hardware and software details of these machines.

For two years, Microbot, Inc. has been developing a low-cost manipulating system so that both amateur computer enthusiasts and others could participate in this exciting field. Their objective has been to provide a complete system including the arm, computer interface, software drivers, coordinate conversions, and the control language so that the user would not have to undertake a major development effort to put the unit into operation.

This manual provides a brief history and background and throughly discusses the construction and operation of the MiniMover-5 manipulator. Other sections deal exclusively with the software and provide numerous examples. Additionally, the hardware and software aspects of using the MiniMover-5 with computers other than the TRS-80 is provided. Finally, this manual provides many generally useful basic and advanced robotic principles, involving both hardware and software, which can be extended outside the realm of manipulators to mobile systems, etc.

# II GETTING STARTED

This section gives step-by-step instructions (1) for connecting the MiniMover-5 to a power supply and to a TRS-80 computer (Level II required), (2) for loading the software from cassette, and (3) verifying that the entire system is functioning properly. The material presented in the remainder of this manual can be read at a later time.

Those interested in operating the MiniMover-5 from other computers need only read parts A and B of this Section. Those interested in writing their own software are referred to Section X and Section XI.

## A. Mechanical Check Out

Carefully unpack the MiniMover-5 manipulator and place it on a convenient working surface near the computer. Visually inspect it and remove any pieces of packing material which might have become lodged in the gearbox or housing. Operate all of the joints by manually turning each of the six large plastic drive gears which are accessible at the rear of the arm. The drive gears should turn with the application of a moderate force and each of the joints should move. Now, grab the body and arm segments and move each one manually with the exception of the the hand and wrist. There should be little play and the arm segments should offer only moderate resistance (1 - 2 lbs) to movement.

## B. Connecting The Arm

Attach the flat ribbon cable at the back of the base of the arm to the card-edge expansion connector on the back of the TRS-80 computer as shown in Figure 1. The cable should exit downwards at the back of the computer. Alternatively, if a TRS-80 expansion interface is being used, then the arm cable should be plugged into its card-edge expansion connector, and again should exit downwards.

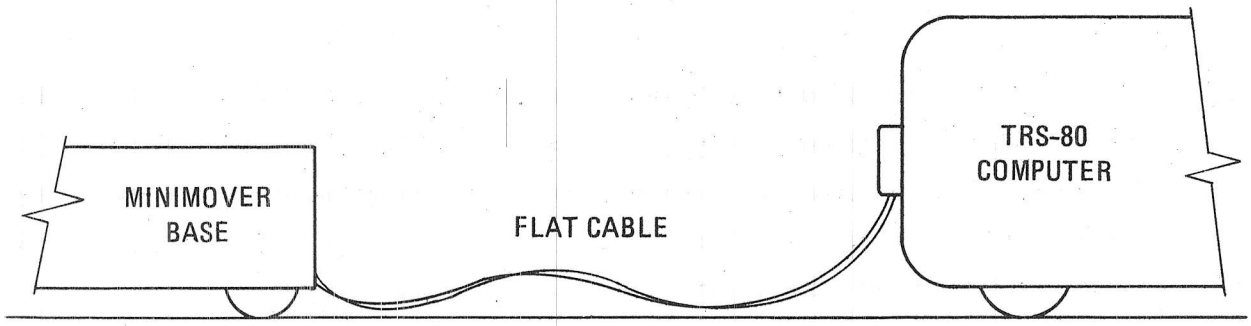



Figure 1 Proper Connection to the TRS-80

Connect the power cord at the back of the base of the arm to a 12-volt DC power supply which is capable of supplying at least 4 amperes. The power cord terminals will fit a variety of screw terminals and binding posts.

```
**************************************************************
*                                                            *
*                        IMPORTANT                           *
*                                                            *
*  TERMINAL WITH RED TAG MUST BE CONNECTED TO POSITIVE VOLTAGE  *
*                                                            *
**************************************************************
```

The power supply need not be regulated, but it should have a ripple (peak-to-peak) of 1 volt or less. A few companies supplying such power supplies are given in Table 1. Battery eliminator supplies (providing 13.8 volts) are not recommended as the motors will become quite hot and might be damaged.

Table 1

Sources of Power Supplies for MiniMover-5

| Company | Supply | Comments |
|---|---|---|
| Clifford Industries<br>P. O. Box 436<br>Camarillo, CA 93101 | Vista V<br>Vista 6CB | All Low cost,Unregulated & Enclosed with binding posts, cord and switch. |
| Standard Power<br>1400 S. Village Way<br>Santa Ana, CA 92705 | 60B12 | Open frame, Unregulated. |
| Elpac Power Systems<br>3131 S. Standard Ave.<br>Santa Ana, CA 92705 | SOLV30-12 | Open frame, Regulated. |
| ACDC Electronics<br>401 Jones Road<br>Oceanside, CA 92054 | OEM12N5.8<br>EC12NG | Open frame, Regulated.<br>Open frame, Regulated. |
| ERA Transpac Div.<br>311 E. Park Street<br>Moonachie, NJ 07074 | OE512/12<br>SI2B5P8 | Open frame, Regulated.<br>Open frame, Regulated. |
| Acopian Corp.<br>Easton, PA 18042 | B12GT650 | Enclosed, Regulated. |

C. Interface Check Out

Once the MiniMover-5 has been connected to the Radio Shack TRS-80 Computer (Level-II required), the following procedure will verify proper operation of the interface card*.

(1) Turn on the 12-volt arm power supply.

(2) Turn on power to the TRS-80 and type ENTER in response to the MEMORY SIZE? prompt.

---

* TRS-80 users who have changed the address jumper (see Section VI-B) should substitute the numbers 184, 189, and 191 for the numbers 152, 157, and 159, respectively, in this procedure.

(3) Key in the following line and type ENTER.

```
FOR I=152 TO 157:  OUT I,10: NEXT I
```

(4) Test the six large plastic drive gears - they should all be locked and resist manual turning.

(5) Key in the following line and type ENTER.

```
FOR I=152 TO 157:  OUT I,0:  NEXT I
```

(6) Check that all six large plastic drive gears are again free to turn.

(7) Open the hand by turning the corresponding drive gear manually. Key in PRINT INP(159), and press the ENTER key. 15 should be printed on the screen.

(8) Manually close the hand as in (7) until an audible "click" is heard. (If not heard, check that the grip cable is on the grip switch pulley.) Key in PRINT INP(159) and press the ENTER key. 143 should be printed on the screen.

D. Loading ARMBASIC

Once the proper operation of the interface card has been verified the following steps must be followed in order to load ARMBASIC from cassette.

(1) Attach the cassette recorder according to the directions in the TRS-80 users manual. However, it is strongly suggested that the smaller grey plug not be plugged into the recorder*.

(2) Turn on the power to the computer. In response to the MEMORY SIZE? prompt, press the ENTER key.

(3) Place the ARMBASIC cassette into the recorder and rewind the tape to the beginning.

(4) Type SYSTEM and press the ENTER key.

(5) In response to the *? prompt, type ARMBASIC and press the ENTER key.

---

* The smaller grey plug is used by the TRS-80 to turn the cassette recorder on and off. It has been determined that this can cause a "spike" to be recorded on the tape and thus damage the original recording. This can happen even if the tape is "write protected."

(6) Press the PLAY button on the cassette recorder. In approximately 10 seconds two asterisks should appear in the upper right hand corner of the screen. In another few seconds the following legend should appear*.

ARMBASIC
VERSION 2.X
COPYRIGHT (C) 1980
MICROBOT, INC.

(7) Stop the cassette recorder and rewind** the ARMBASIC cassette.

(8) If ARMBASIC did not load correctly, push the reset button located in the left rear of the TRS-80 keyboard unit, and repeat the above tape loading procedure. The most common loading problem is determining the correct volume setting for the cassette recorder. The volume setting should be changed in small increments between loading attempts; it should not be changed while the cassette is being read. ARMBASIC is recorded three times on the cassette. If, for some reason, the first copy cannot be successfully loaded, then the second or third copies may be attempted.

E. System Check Out

The most complete check out procedure for the MiniMover-5 involves using ARMBASIC to move each of the five joints through its complete range of motion. The step-by-step procedure is as follows:

(1) If the user has changed the address jumper (Section VI-B), type @ARM 2

(2) Type @SET and depress the ENTER key. This places the MiniMover-5 under manual control.

(3) Press the 1 key on the TRS-80 keyboard. This should cause the MiniMover-5 to rotate about its base in a clockwise direction. Depressing the Q key will cause it to rotate in the opposite direction. When the key is released, the motion will stop. Thus the base joint can be moved through its complete range of motion.

(4) Press the 2 key. This will cause the upper arm to rotate up, about the shoulder joint. Pressing the W key causes

---

* The X denotes the particular revision number. The description of ARMBASIC which appears in this manual is valid for any revision of Version 2.

** It is important to rewind the cassette after each use as this will keep the tape clean and prevent it from becoming damaged.

it to move down. The upper arm and hand maintain their orientation in space.

(5) Similarly, the 3 key and the E key will cause the forearm to bend up and down at the elbow, respectively. In this case the hand maintains its orientation.

(6) Pressing the 4 key and the R key will cause the hand to move up and down respectively. This is known as Pitch motion. In this case, two motors will be simultaneously driven.

(7) Pressing the 5 key and the T key will cause the hand to twist. This motion is called Roll. Again, two motors will be simultaneously driven.

(8) Finally, depressing the 6 and Y keys will cause the hand to open and close. The hand should be left in the open position.

(9) Press the 0 (zero) key. This will take the MiniMover-5 out of manual mode and return to ARMBASIC. The ">" prompt character should appear on the screen.

(10) Finally, type @CLOSE followed by ENTER. This will cause the hand to close and the ">" prompt character should again appear. This test causes the grip sensor to be checked. If the switch is shorted the hand will not close; if the switch is open, the motor will continue to try to close the hand, even after it is closed, and ARMBASIC will not display the prompt. If this happens, press the BREAK key on the keyboard.

Once the above procedure has been performed, the user is ready to read the remainder of this manual and to enjoy programming the MiniMover-5 manipulator.

## III BACKGROUND AND HISTORY

### A. Master-Slave Manipulators

Most people are familiar with the first occurrences of mechanical manipulators in the context of the radioactive "hot" cells. These manipulators are in common use in many laboratories throughout the country. An actual example of this type of manipulator is shown in Figure 2. This figure shows radioactive material being handled by the operator from behind a protective shield. These manipulators were called master-slave manipulators because the operator would move a "master" manipulator and the "slave" manipulator positioned inside the cell would replicate its movements. This terminology was originated by Ray C. Goertz of the Argonne National Laboratories in the late 1940's [7]. A master-slave manipulator system is characterized by two manipulators in direct linkage. This allows the operator to assemble, and disassemble radioactive components using slightly modified hand tools with safety from radiation.

The early master-slave manipulators were first connected through mechanical linkages and later through flexible steel cables. A more recent example of this type of master-slave manipulator is shown in Figure 3, and is called the MA-11. The MA-11 is manufactured by LaCallene in France. With this type of system, movement of the master produces movement of the slave, and movement of the slave produces movement of the master. Thus, when the slave manipulator encounters an obstacle, the master becomes blocked and the operator can "feel" the remote* environment. Because the manipulators can be moved from both the master and the slave ends, these systems are called bilateral. The ability to feel the environment is called force feedback to the operator. The fact that these manipulators provide force feedback is

---

* Remote in the sense of only a few feet

very important to the operator. Imagine trying to grab an egg without any feeling. One must grasp the egg tightly enough so that it will not slip out of the grasp, but at the same time it must not be grasped too tightly or it will break. Without force feedback this is an extremely difficult task.

The next development of master-slave manipulators was the introduction of electrical control. This was done by mounting potentiometers (or other sensors) and electric motors on both the master and slave manipulators. Thus, when the master manipulator is moved, signals from its potentiometers are electronically passed on to the slave manipulator. This produces a command to the slave manipulator which causes its motors to drive each joint in the correct direction, until the potentiometers indicate that the position is correct.

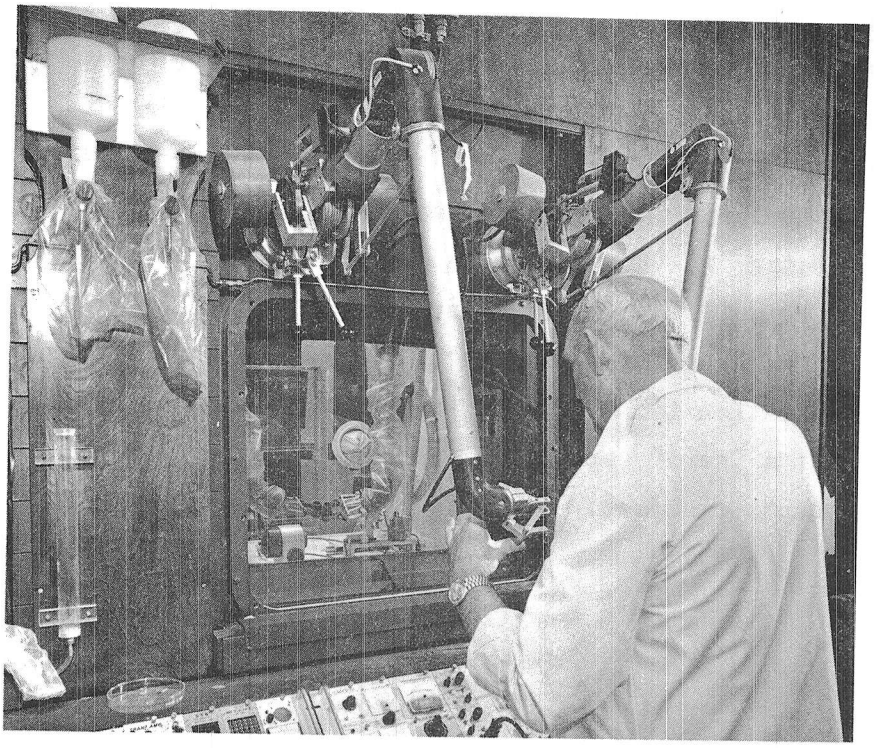


Figure 2 Model-8, Master-Slave Manipulator in a Radioactive Environment

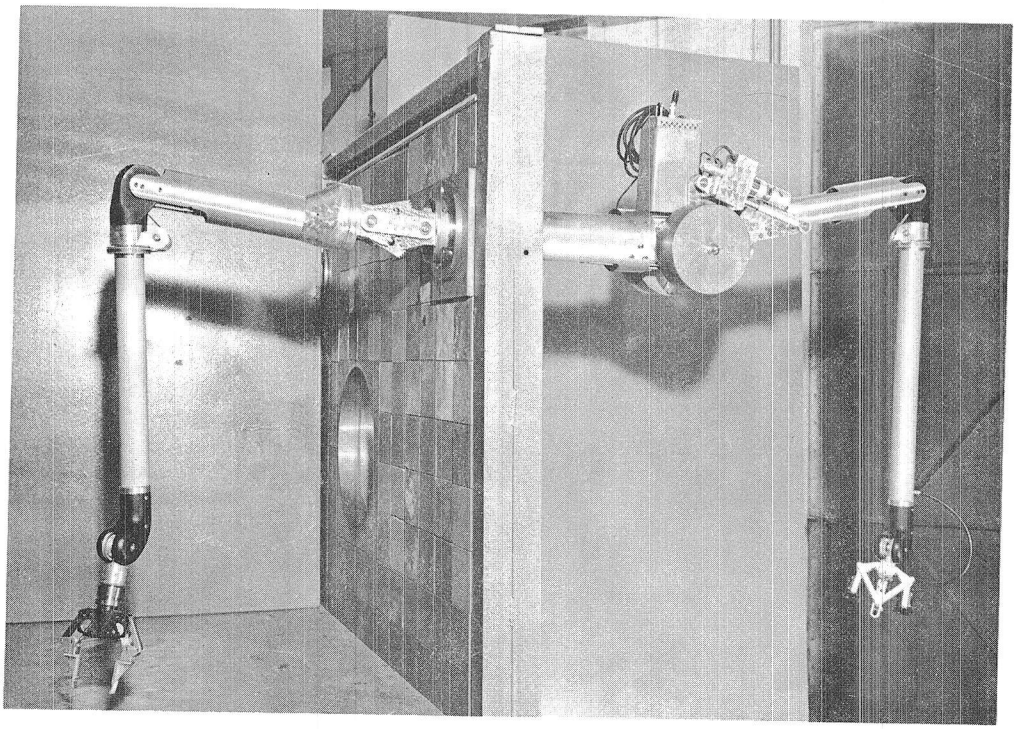


Figure 3 The MA-11 Master-Slave Manipulator

Two such manipulator systems are shown in Figure 4 and Figure 5. Figure 4 shows a master-slave manipulator which was developed from a prosthetic device by Rancho Los Amigos Hospital in Los Angeles and was called the Rancho arm. This figure clearly shows how the movement of the slave manipulator in the background duplicates the movement of the master manipulator in the foreground. A second electrically linked manipulator system is shown in Figure 5. This manipulator system was developed by the Ames Research Center of The National Aeronautics and Space Administration in Mountain View, California, and is based upon their design of a "hard" space suit which was originally designed to be worn by astronauts in space.

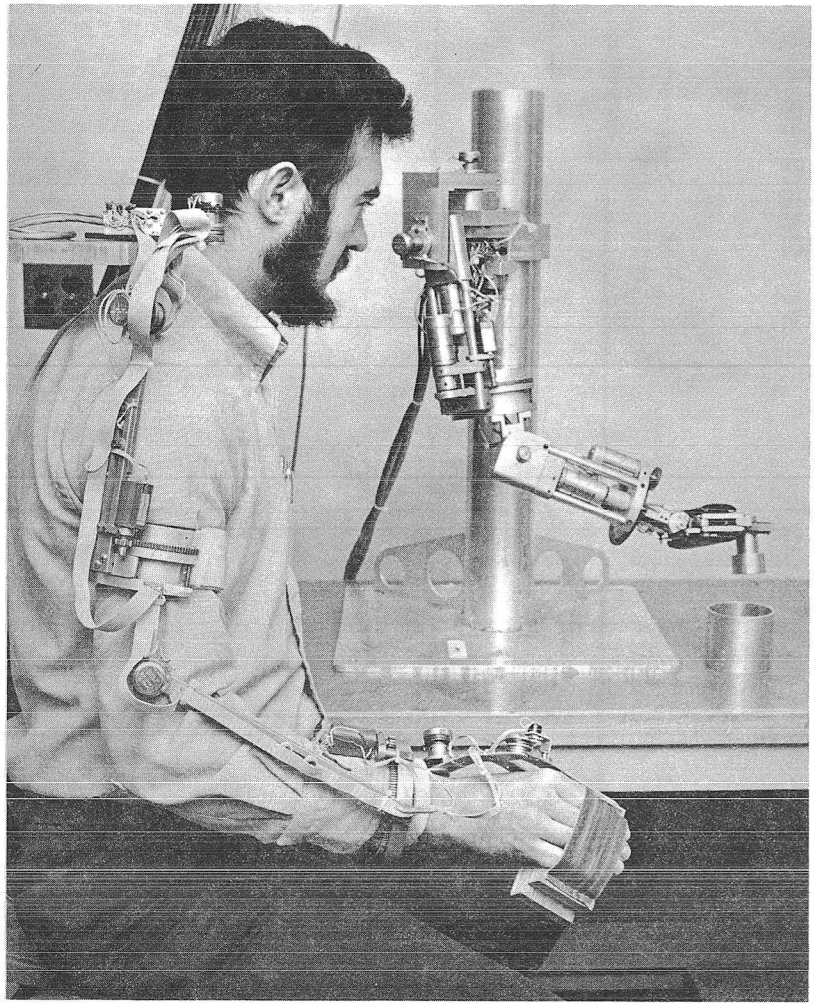


Figure 4 The Rancho Master-Slave Manipulator

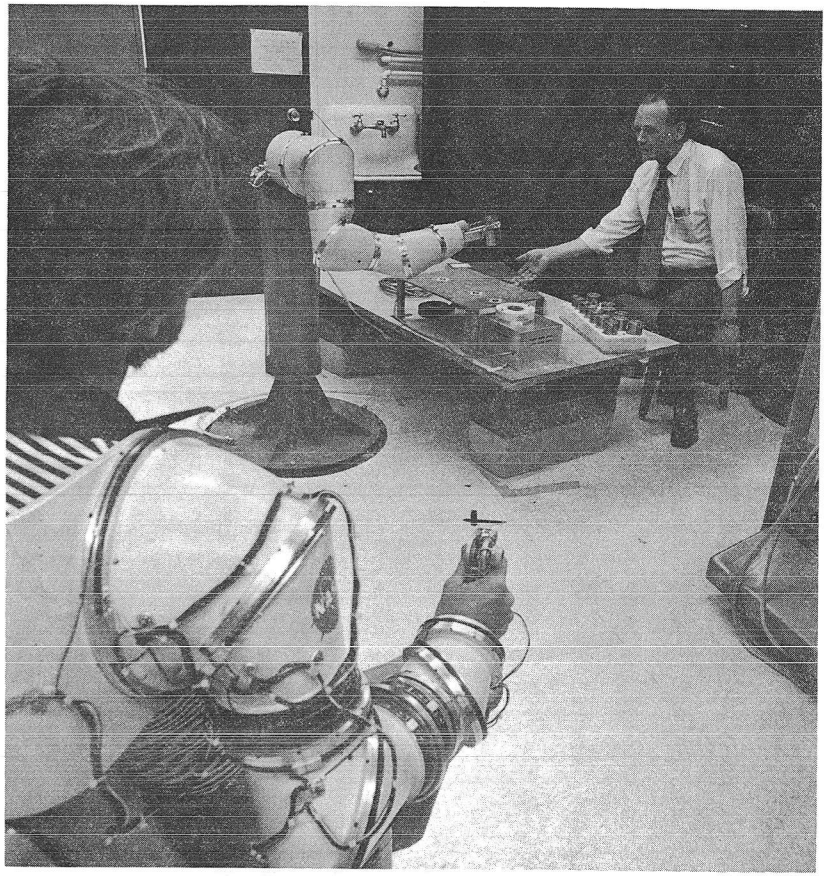


Figure 5 The NASA-Ames Master-Slave Manipulator

Although the manipulators are not directly coupled, force feedback can still be provided to the operator. When the slave manipulator encounters an object, the slave will no longer follow the master directly, and signals from the slave can be passed back to the master to provide force feedback. Now, because the manipulators are connected electronically rather than mechanically, the two manipulators can be separated by large distances rather than just a few feet. This development set the stage for the development of teleoperator systems.

## B. Teleoperators

A teleoperator system is defined as "a general purpose, dextrous, cybernetic machine" [7]. It is characterized as shown in Figure 6.

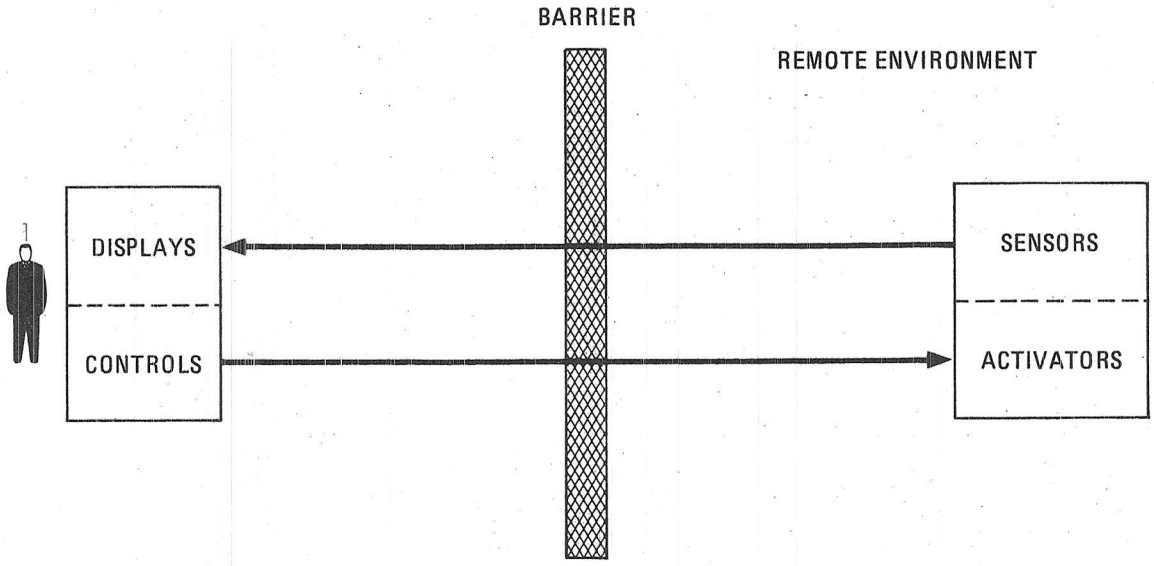



Figure 6 Block Diagram of a Teleoperator System

In these systems, there is a communication link through a barrier. This barrier might be a long distance, task difficulty, or a hostile environment such as under the ocean or out in space. In addition to

force feedback, various sensors are provided at the remote end of the system to provide the operator with indications as to the state of the remote environment. The operator has various controls with which he can control the remote actuators. The sensors might be closed circuit television, auditory sensors, tactile (touch) sensors, or force sensors. The operator's controls might take the form of a master replicate* controller, joysticks similar to the type used by airline pilots, or various buttons and switches. Those interested in teleoperators will find reference [7] worthwhile reading.

## C. Computer Augmented Teleoperators

The development of augmented teleoperator systems was primarily due to the National Aeronautics and Space Administration (NASA) who sponsored much research throughout the country in this area. An augmented teleoperator system is shown schematically in Figure 7.

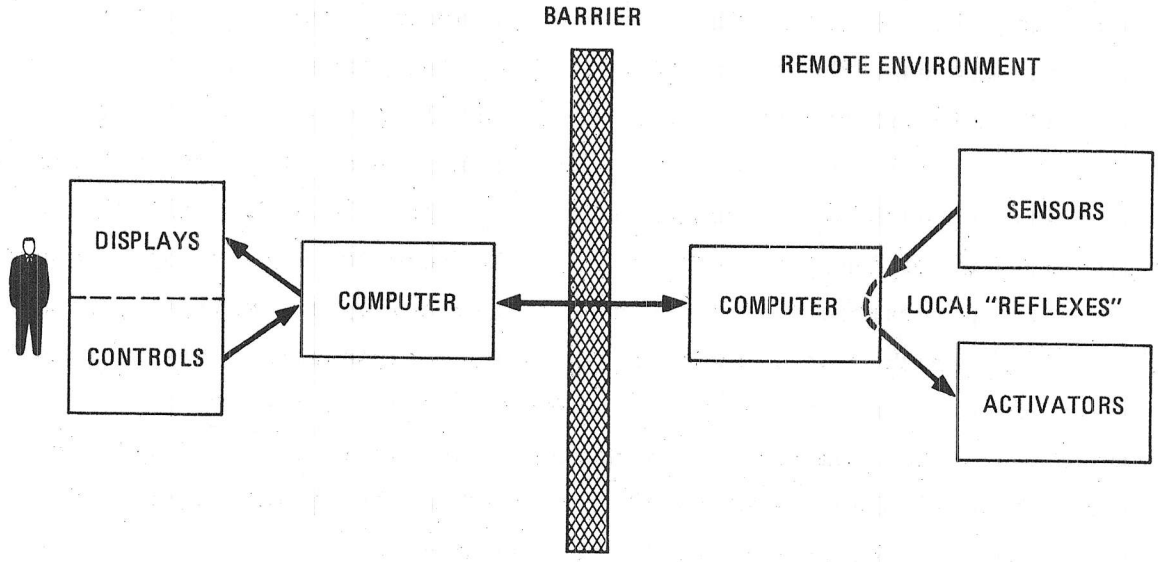



Figure 7 Block Diagram of an Augmented Teleoperator System

---

* A replicate controller is an arm with the same geometry as the slave, frequently scaled down in size.

The essential element in such a system is the addition of one or more local computers to provide intelligence to the system. To understand the benefits of such a system, consider the case where an operator at an earth station is controlling a manipulator on the surface of the moon in order to collect lunar rock samples. In this case, approximately 1.4 seconds are required to transmit signals from the earth to the moon, and another 1.4 seconds to transmit them back. Thus, the operator's view of the remote work site is always 1.4 seconds behind, and the remote manipulator will not even begin to respond to the operator's command for 1.4 seconds. Thus, if something unexpected happens at the remote work site, serious damage might occur before the operator can react to the situation. Another problem is that the operator, in this type of situation, will adopt a complex move-and-wait control strategy in which he moves the manipulator slightly, waits to see what has happened, and again moves it slightly [6]. Operators find this type of control strategy extremely tedious and tiring.

With a remote computer at the work site, local autonomous reactions can be programmed into the system. This is similar to the human autonomous nervous system. When humans inadvertently grab or touch something that is hot, their autonomous nervous system takes over and they quickly pull their hand back. Thus, in similar fashion, the lunar manipulator could be programmed to immediately stop moving if sensors on the manipulator indicate that the manipulator has encountered a hazardous situation such as an unexpected obstacle. This minimizes the possibility of damaging collisions. Furthermore, the remote computer could be programmed to aid the human operator by automatically picking up a lunar rock and depositing it into a container. All the human operator need do, is to move the manipulator to the vicinity of the rock, and then command the manipulator to pick it up. The remote computer would then "take over" control from the operator, complete the task, and then return control to the operator.

D. Industrial Robots

After the introduction of augmented teleoperator systems with local computers that could perform certain tasks automatically, it became increasingly obvious that, under certain circumstances, a completely automatic system would be of great use. In the mid 1960's, at Stanford University, researchers removed the master arm from the Rancho Arm system (shown in Figure 4), and placed the slave manipulator under computer control. Then, using early computer vision techniques, the positions and orientations of each of several blocks which were randomly placed upon a table top were determined. The Rancho manipulator was then commanded to stack the blocks. The arm was affectionately called "Butterfingers" because it frequently dropped one or more of the blocks.

One of the first practical domains such systems were applied to was the factory, and the manipulators used were known as industrial robots. These systems were initially put to work on highly repetitive, boring, and dangerous tasks. Today thousands of industrial robots are "working" in factories in the U.S.A. and overseas and help manufacture many of the products each of use.

One of the first industrial manipulators which was introduced into the automobile assembly line was the Unimate Series-4000 manipulator shown in Figure 8*. This resulted in an increase in productivity and a decrease in worker dissatisfaction. In fact, the Unimate could move faster, further, and lift heavier loads than its human counterpart. Moreover, for such operations as placing workpieces in equipment such as punch presses, accidents were greatly reduced. Even if they did occur, they did not result in the loss of the workers hands, but merely in the loss of mechanical equipment. Unimation, Inc. has sold several thousand of these robots.

In the mid 1970's, another industrial manipulator was introduced by Cincinnati Milacron. This is shown in Figure 9. This manipulator, called the $T^3$ or The Tomorrow Tool, can lift up to 100 lbs, can move at a speed of up to 50 inches/second, has a working volume of 1000 cubic

---

* Figure 8 and Figure 11 are courtesy of Unimation, Inc.

feet, can reach to a height of almost 13 feet, and has an accuracy of 0.050 inches. Thus, it can accomplish tasks that previously could be performed only by several factory workers working together or by special purpose equipment costing much more. The $T^3$ is general purpose as it is completely computer controlled, and has many sensory inputs which can be used to modify its behavior to accommodate to various environmental conditions.

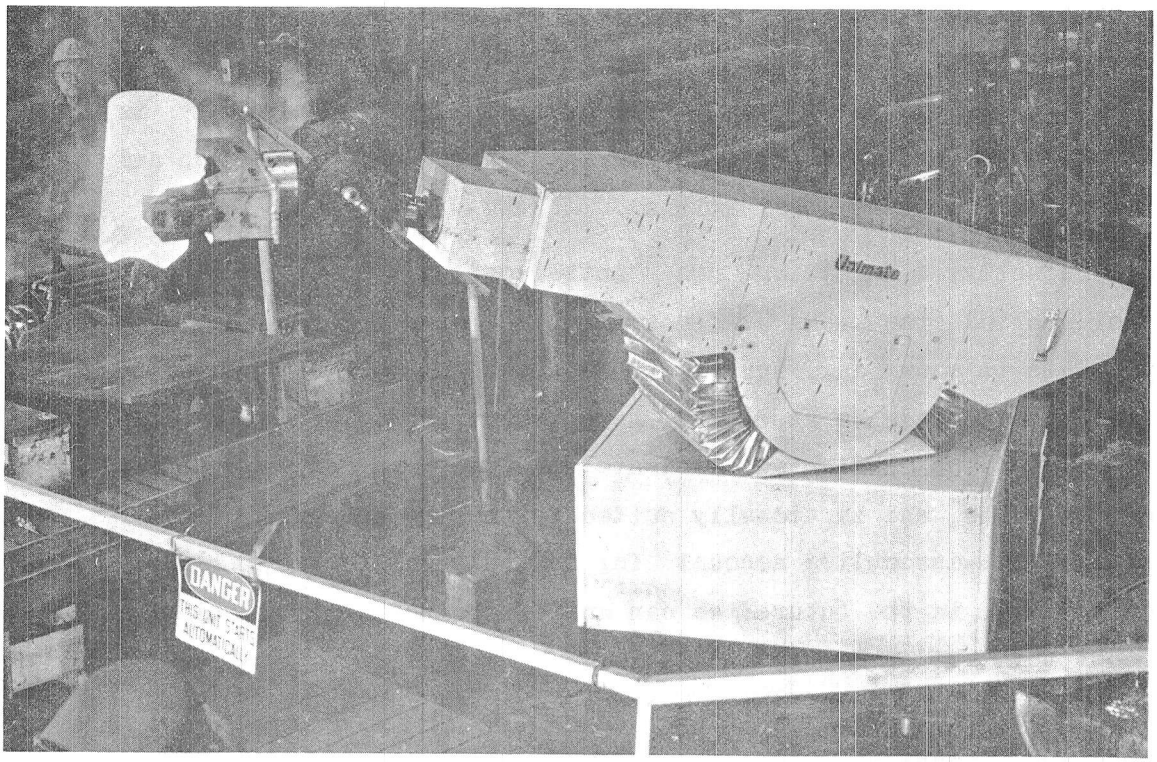



Figure 8 Unimate Industrial Robot

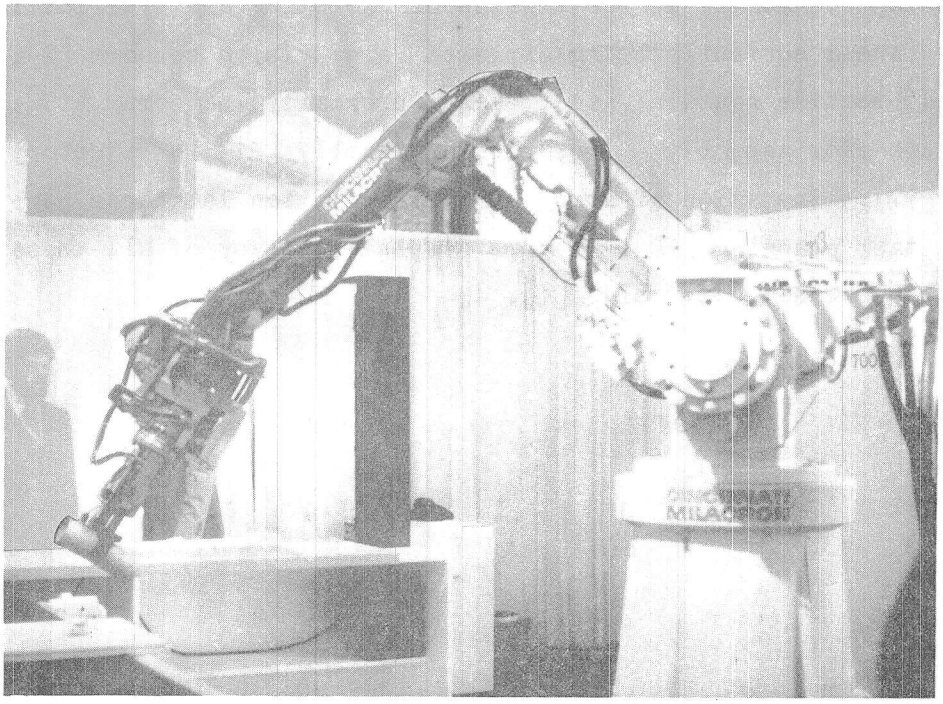


Figure 9 Cincinnati Milacron $T^3$ Industrial Robot

For smaller jobs, ASEA in Sweden, markets the IRB-6Kg manipulator shown in Figure 10. This manipulator can lift a load of 13 lbs and has a reach which is comparable to the human arm. However it is approximately ten times more precise than the Series-4000 or the $T^3$, and is ideally suited for many applications.

Just recently, Unimation has introduced a new industrial manipulator called the PUMA or Programmable Universal Machine for Assembly. This is shown in Figure 11. This manipulator is intended to satisfy the unique needs of assembly. Here the emphasis is on precision and dexterity. The PUMA is accurate to 0.005 inches, can lift approximately 7 lbs, and is ideally suited for making sub-assemblies of small parts. Sub-assemblies account for the bulk of factory assembly operations. Thus, in the future, we can anticipate better products at a reduced cost than are presently available.

## E. The Future of Manipulators

Currently, at many research laboratories throughout the country, research is being conducted in the fields of Robotics and Artificial Intelligence. Areas currently being explored are: vision sensors [2], [8]; force and tactile sensors [11], [14]; proximity sensors [17]; compliant devices for assembly [18]; robot assembly [21]; trajectory calculations [12]; kinematics of arms [9]; manipulation languages [4]; and automatic task planning [10]. A more complete survey of all these areas is given by Abraham, et. al. [1].

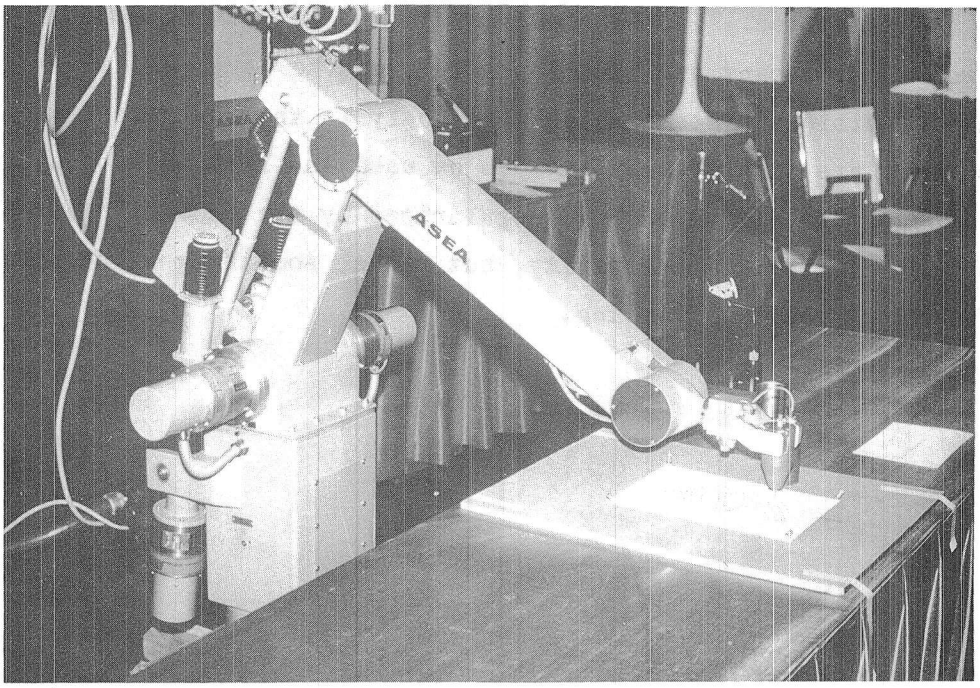



Figure 10 The ASEA IRB-6Kg Industrial Robot

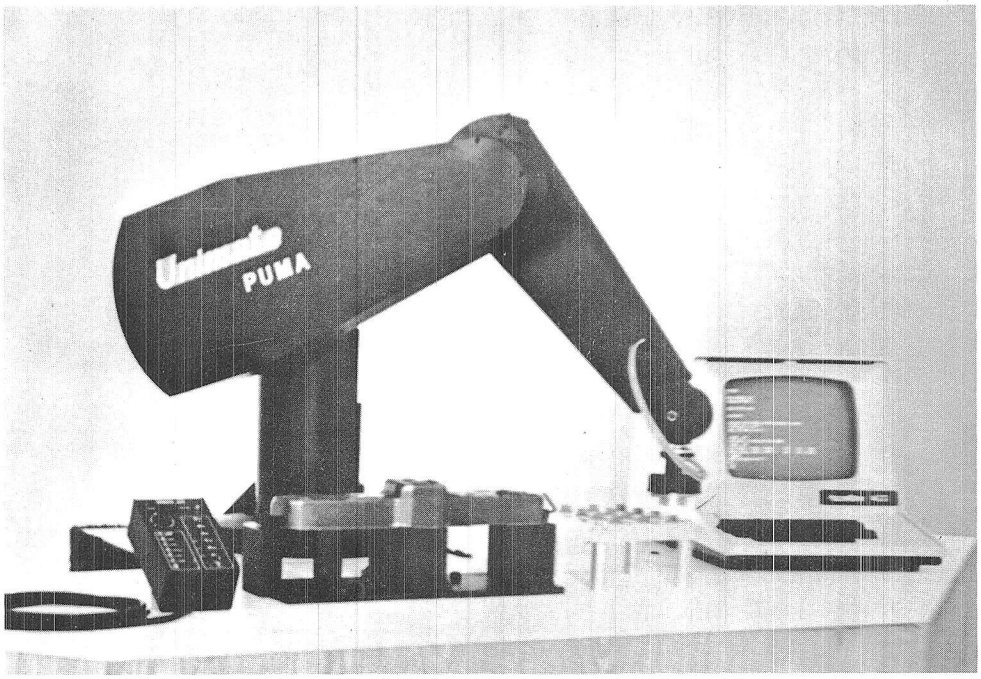



Figure 11 Unimation PUMA Industrial Robot

This research is aimed at providing new types of sensory inputs and increased computer intelligence for mechanical equipment. Even though these areas are only in the research stages, we can expect to see this research foster many new developments over the next decade. In fact, computer "vision" systems which were only in the research stage a few years ago, are finally starting to be used in factories today. Thus, the type of "robot" which was popularized by the movie "Star Wars" is not a technological impossibility, but rather something we should expect the future to eventually bring.

# IV MECHANICAL CONSTRUCTION

## A. Introduction

The MiniMover-5 manipulator is a five-jointed mechanical arm with a lifting capacity of 8 ounces when fully extended. Each joint is controlled by a stepper motor mounted on the body. The manipulator has a resolution* of 0.013 inches. The end of the hand can be positioned anywhere within a partial sphere which has a radius of 17.5 inches, as shown in Figure 12. The maximum speed is from 2 to 6 inches per second, depending upon the weight of the object being handled**. Detailed performance characteristics of the MiniMover-5 are given in Table 2.

The MiniMover-5 hand has a maximum opening of 3 inches. The 2-bar linkages maintain the fingers parallel during opening and closing. A novel control system permits one motor to control both hand opening and grip force. A built-in sensor enables the squeezing force to be controlled from 0 to 3 pounds. The sensor permits adaptive control so that the presence of an object as well as the size of the object may be determined as the hand closes.

---

* Resolution is the smallest amount that the manipulator can move.

** This is known as the load.

Table 2

MiniMover-5 Performance Characteristics

GENERAL

| | |
|---|---|
| Configuration | 5 revolute axes and integral hand |
| Drive | Electric stepper motors-<br>Open loop control |
| Controller | Microcomputer provided by user |
| Interface | TRS-80 or 8-Bit parallel |
| Program language | ARMBASIC for TRS-80<br>( Z-80 driver available ) |
| Power requirement | 12 volts, 4 amps DC |

PERFORMANCE

| | |
|---|---|
| Resolution | 0.013 in (0.3 mm) maximum on each axis |
| Load Capacity | 8 oz (225 gm) at full extension<br>16 oz (450 gm) at half extension |
| Gripping force | 3 lbs (13 N) maximum |
| Reach | 17.5 in (444 mm) |
| Static load force | 4 lbs (18 N) maximum |

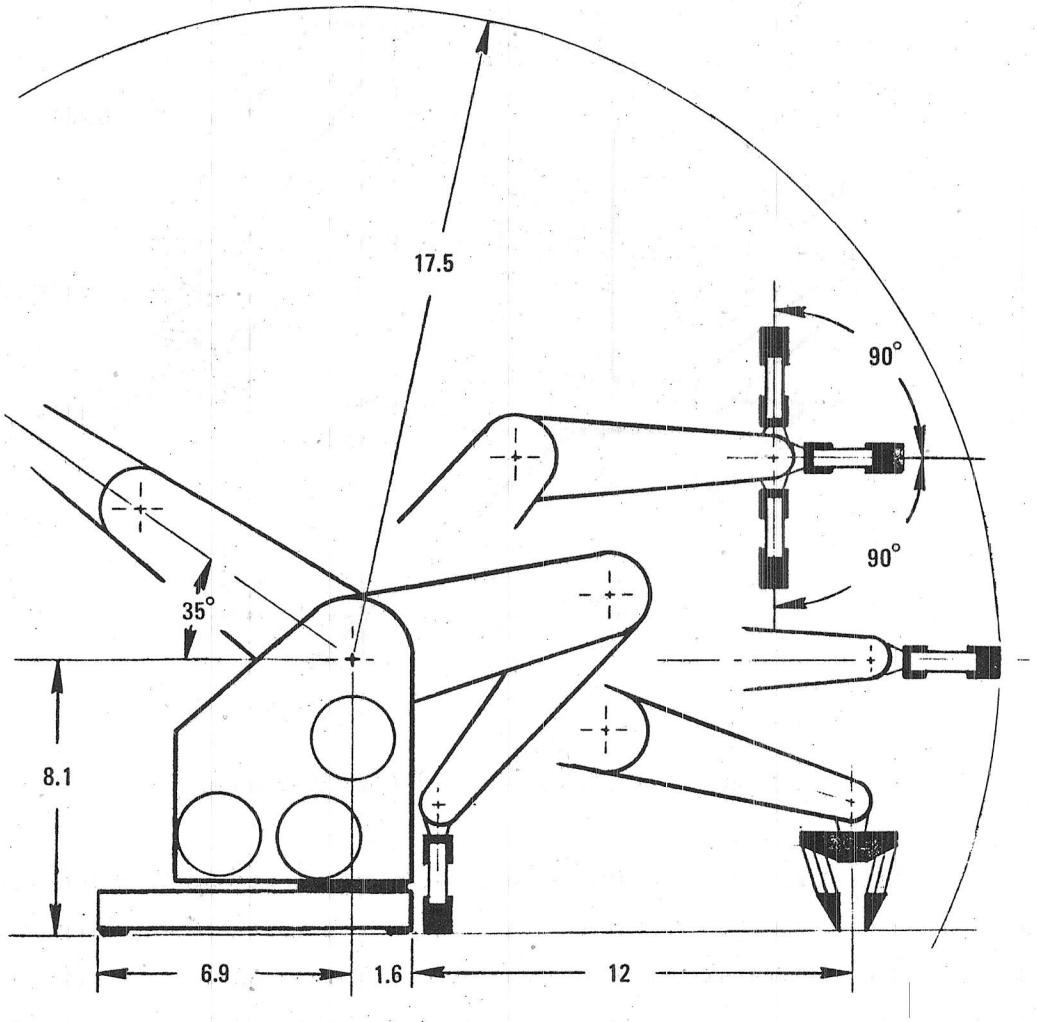
DETAILED PERFORMANCE

| Motion | Range | Speed (full load) | Speed (no load) |
|---|---|---|---|
| Base | ±90 deg | 0.37 rad/s | 0.42 rad/s |
| Shoulder | +144, -35 deg | 0.15 rad/s | 0.36 rad/s |
| Elbow | +0, -149 deg | 0.23 rad/s | 0.82 rad/s |
| Wrist Roll | ±180 deg | 1.31 rad/s | 2.02 rad/s |
| Wrist Pitch | ±90 deg | 1.31 rad/s | 2.02 rad/s |
| Hand | 0-3 in (0-75 mm) | 3 lb/s (13N/s) | 0.80 in/s<br>(20 mm/s) |

PHYSICAL CHARACTERISTICS

| | |
|---|---|
| Arm weight | 6 lbs (3 kg) |
| Interface cable length | 3 ft (900 mm) |

The major structural components are shown in Figure 13. The manipulator consists of a fixed base within which is housed the computer




NOTE: Dimensions in inches.

Figure 12 Operating Envelope of the MiniMover-5

interface card. From the rear of the base extends the interface cable and the D.C. power cord. The body swivels relative to the base on a hollow shaft which is attached to the base. This is known as the base joint.

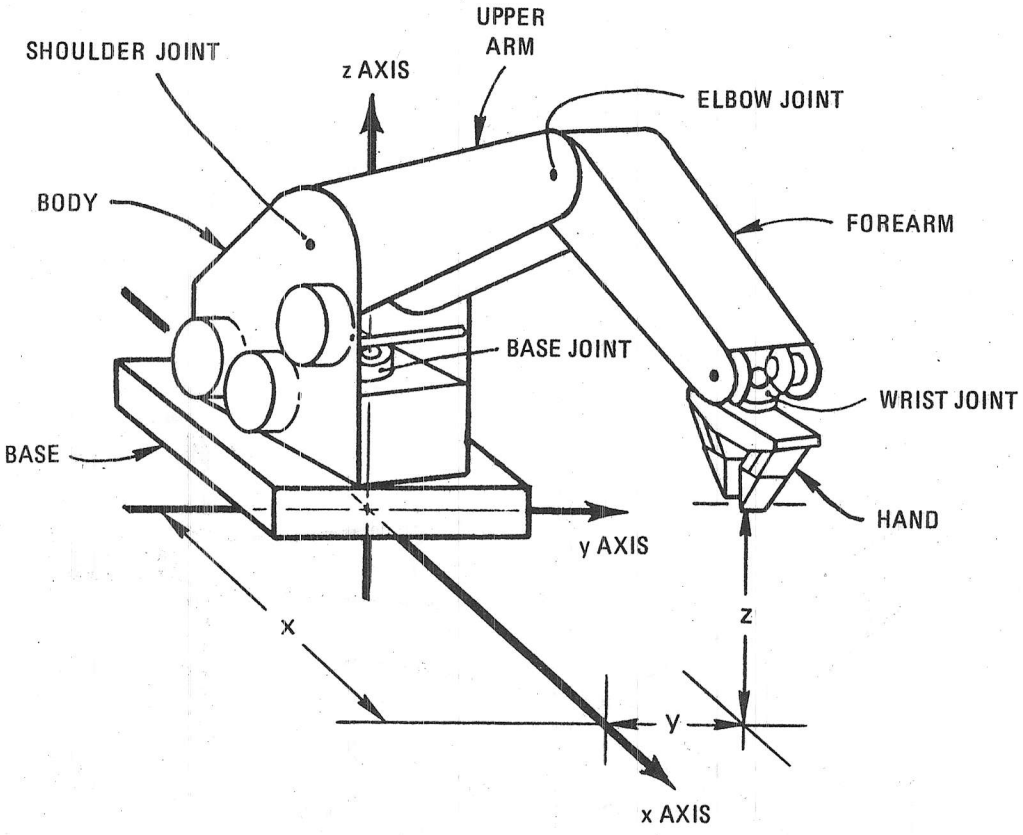



Figure 13 Major Structural Components

Six stepper motors with their corresponding gear reductions, are mounted on the body and control each of the joints. The power wires for each of the motors are passed from the interface card contained within the base, through the hollow shaft to the body. This approach prevents the need for a complicated twisting cable drive system.

At the upper end of the body is attached the upper arm. The upper arm rotates relative to the body on a shaft. This is known as the

shoulder joint. Similarly, the forearm is attached to the upper arm by another shaft. This is known as the elbow joint. Finally, the hand* is attached to the forearm at the wrist joint.

In general, the base, the body, and all the segments are hollow sheet metal parts which provide strength but are still light weight. The body, base, and segments are connected to each other by means of shafts or axles which pass through bearings mounted on the segments.

## B. The Cable Drive System

The motors and drive system are contained almost entirely within the body. The stepper motors drive separate cable drums by means of individual gear reductions. For simplicity, all six drive gears have been mounted on the same shaft. Thus, the drive is very compact and weight is kept to a minimum. From the drive system contained within the body, cables extend to the base, each of the segments, and the hand. The cable drive system is shown in Figure 14. The cables to the base and the other links are wound around the hubs of the drive gears. This provides both the required gear reduction and drums for the cables.

### 1. Base Drive

The base cable (Motor 1) causes the body to rotate on the vertical base axle by driving a pulley mounted on the base. Two small pulleys, located at the bottom of the body, change the base cable direction such that the cable feed is tangent to the surface of both the drive drum and the base pulley.

### 2. Shoulder Drive

The shoulder cable (Motor 2) causes the upper arm to rotate on the horizontal shoulder axle which connects it to the body. This cable passes around a drive pulley which is mounted on the shoulder shaft. This cable terminates on the upper arm housing. Means to maintain the cable under tension is provided at the upper arm termination. Shoulder

---

* The hand is sometimes called the end-effector or gripper

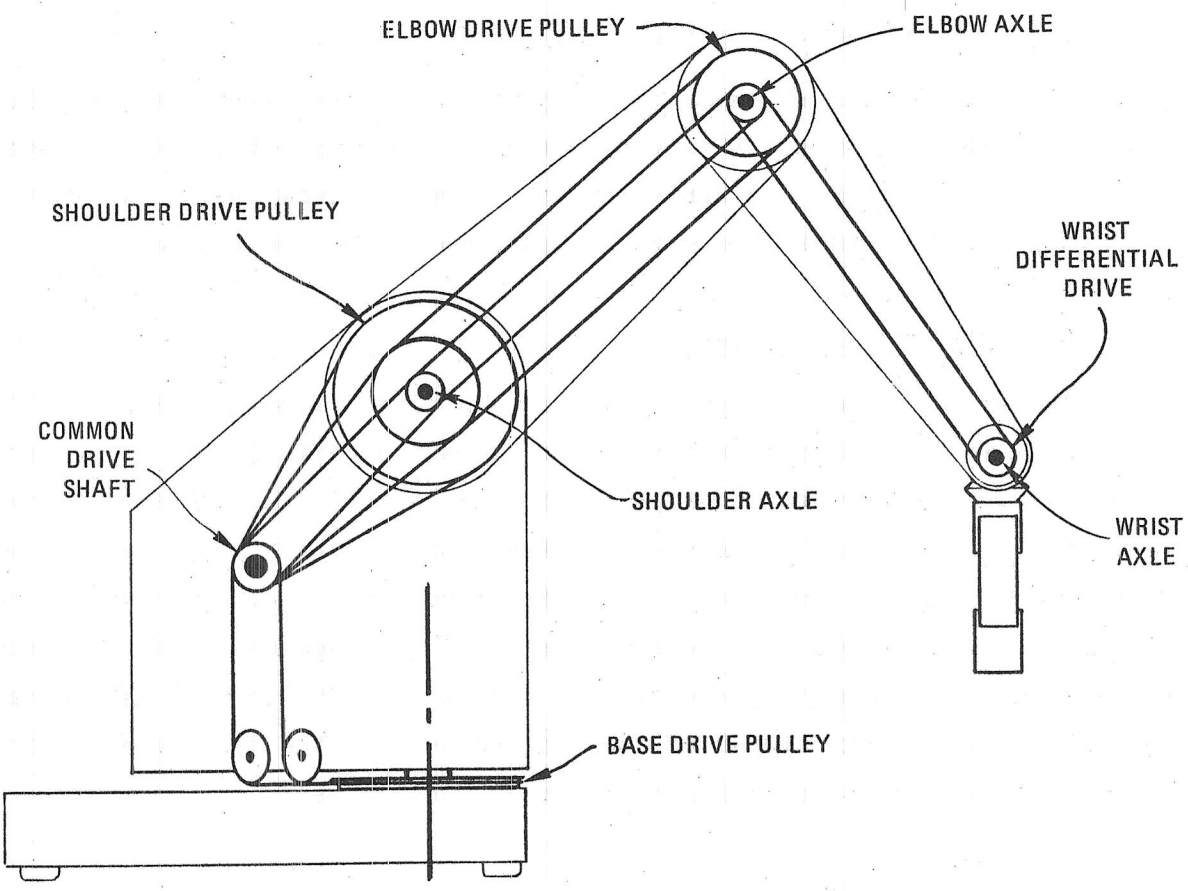



Figure 14 Cable Drive System

rotation causes equal and opposite elbow and wrist rotation so that the orientation of the hand remains unchanged.

3. Elbow Drive

The elbow cable (Motor 3) causes the elbow to rotate on the horizontal elbow axle which connects the forearm to the upper arm. This cable first passes around an idler pulley on the shoulder axle, and then passes around a drive pulley of the same diameter which is attached to the elbow axle. The cable then terminates on the forearm housing. A cable tensioning mechanism is provided at this termination. Elbow rotation causes equal and opposite wrist rotation, thus maintaining hand orientation.

4. Wrist Drive

The right and left wrist cables (Motor 4 and Motor 5) cause the hand to Roll and Pitch relative to the forearm. These cables together control the wrist joint which is shown in Figure 15. Both cables pass around idler pulleys located on the shoulder axle and the elbow axle. Finally they pass around the hubs of bevel gears which turn on the wrist axle. Tension is maintained in both cables by attaching their ends together with a tensioning device.

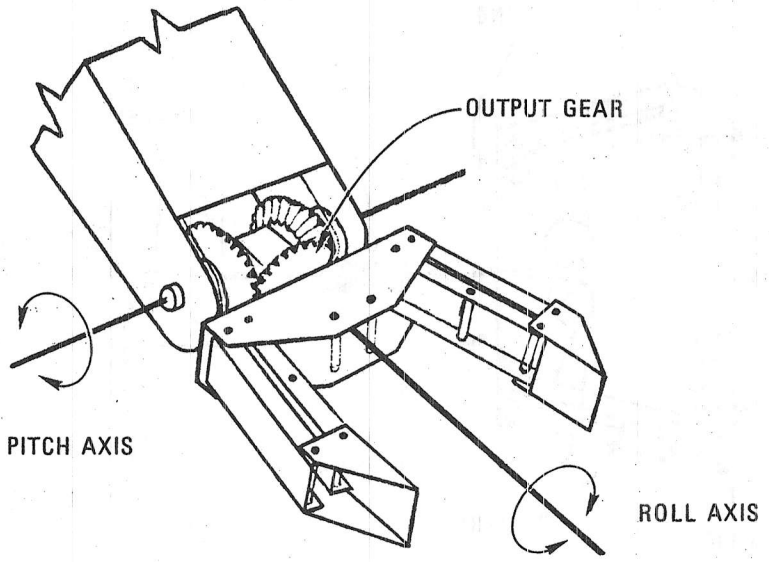



Figure 15 The Wrist Joint

The two bevel gears on the wrist axle mesh with the output gear on the hand axle. The hand axle is at right angles to the wrist axle, and intersects it at the center of the wrist yoke. This configuration forms a differential gear set. Thus, if the left and right wrist cables move in the same direction, they control the Pitch of the hand; if they move in opposite directions they control the Roll of the hand.

C. The Hand

The hand assembly is shown in Figure 16. The hand housing is attached to the output gear of the differential gear set. The hand housing holds two pairs of links, and each pair of links terminates in a finger. The housing, the links, and the fingers are attached to each other by small pins. Torsion springs are located on the pins which attach the links to the hand housing and provide the return force to open the hand as the hand cable is slackened.

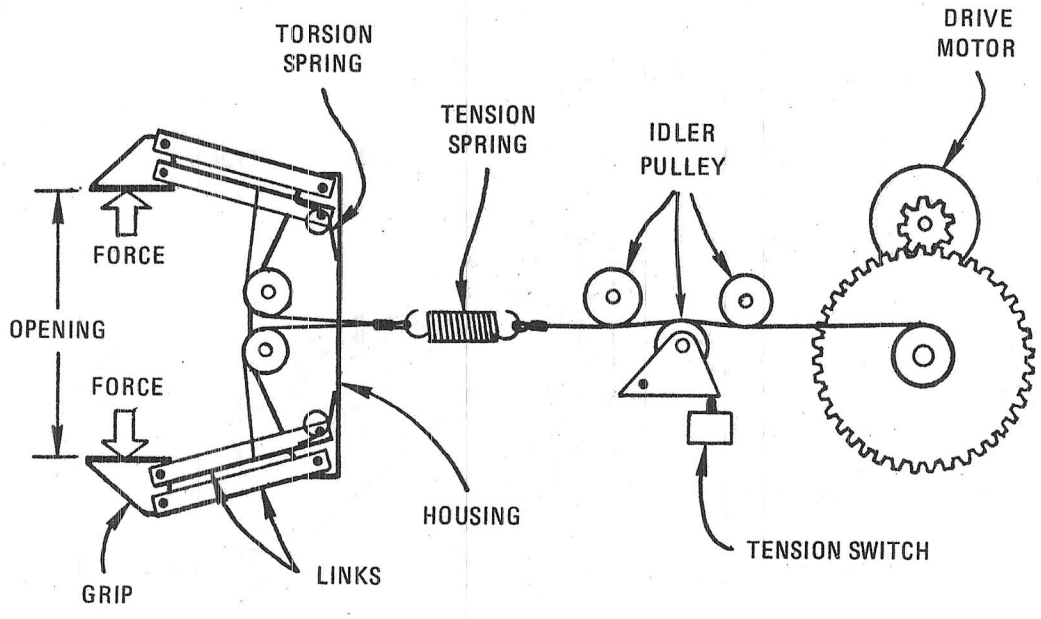



Figure 16 Hand and Drive Mechanism

The hand cabling system is also shown in Figure 16. The hand cable (Motor 6) is attached to the hand drive drum located in the body. It then passes over an idler pulley located on the shoulder axle. From there it passes over an idler pulley which is mounted on a sensing bracket which is attached to the upper arm. Finally, it passes over an idler pulley on the elbow shaft and terminates at a tension spring.

Attached to the other end of the tensioning spring and in line with the sixth cable are two separate link drive cables. These cables pass over two guide pulleys in the wrist yoke and then through the center of the hollow hand axle. When the cables emerge from the hand axle, they pass over separate idler pulleys which are mounted in the hand base. They then pass around idler pulleys which are mounted on the inner links of the hand, and return to and terminate on the shafts of the two pulleys mounted on the hand base. This arrangement forms two block-and-tackles which augment the gripping force of the hand. The use of identical cabling on both links provides for symmetrical closure of the hand.

As previously mentioned, the hand cable passes over a pulley which is attached to a switch. As the hand closes upon an object to be grasped, the cable tension mounts, causing the grip switch to be activated. This switch closure may be used to either stop the drive motor, or to control the gripping force.

The tension spring which is mounted in series with the hand drive cable assembly permits control of the gripping force with a position controlled drive motor. After signals from the grip switch are obtained, additional cable take-up stretches the spring. This converts cable take-up into gripping force. The relationship between gripping force, hand opening, and drive motor rotation is shown in Figure 17. Thus, gripping force, as well as hand opening, can be controlled by the same cable drive.

Finally, it should be noted that when the elbow is commanded to move, the grip cable becomes further wound over its elbow idler pulley. This causes the hand opening to change whenever the elbow is moved. Thus, the software driver may uncouple hand opening from elbow bend by playing out the grip cable by an amount equal to the rotation of the elbow. This allows the hand opening to remain constant whenever the elbow is moved. The ARMBASIC driver provides this uncoupling function.

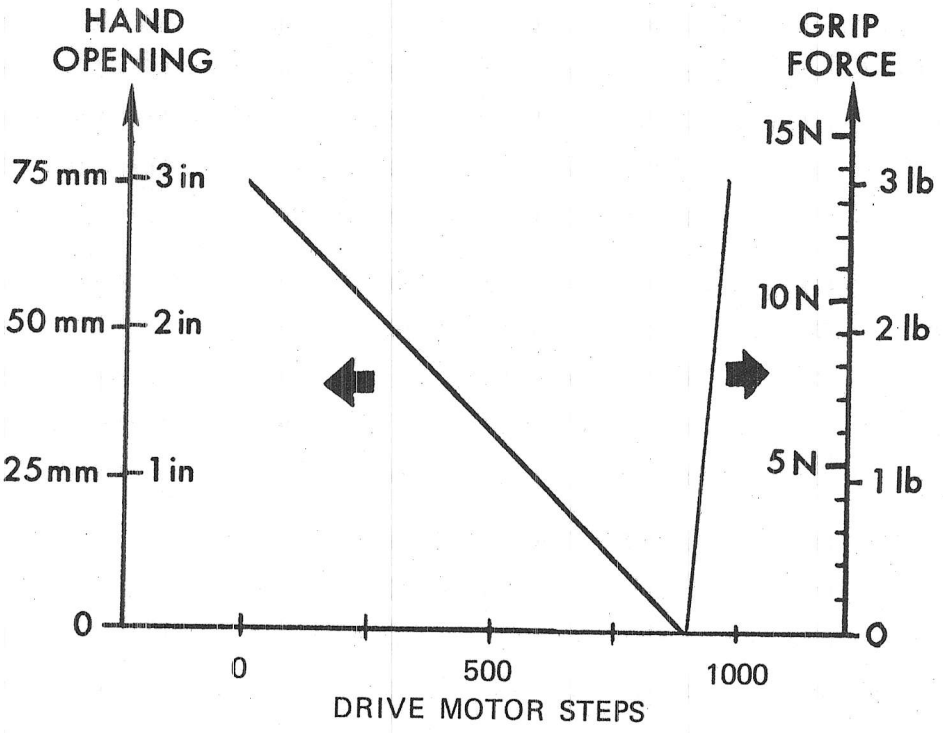



Figure 17 Hand Control Diagram

D. Cable Tension Adjustments

Drive cables for the base, shoulder, and elbow are tensioned by a sliding mechanism that may require occasional adjustment. The base tensioning adjustment is shown in Figure 18. After a period of extended use or an extreme overload, cables may stretch or the tension adjustments may slip and one or more of these joints may develop play. Play (looseness or backlash) may be detected by manually grabbing each arm segment and moving it slightly in each direction. If the segment moves a small amount without back driving its gear train or motor, then the cable needs to be tightened.

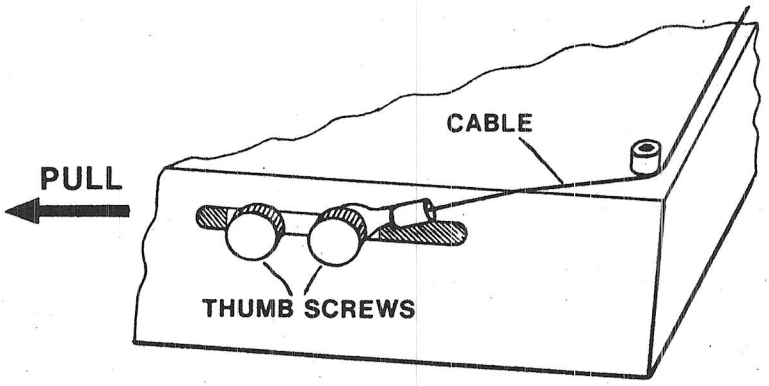



Figure 18 Base Tension Adjustment Mechanism

Tension adjustments are provided at the following locations:

| Drive | Location |
|---|---|
| Base | Right side of base |
| Shoulder | Right side of upper arm |
| Elbow | Left side of forearm |

The adjustment procedure is as follows:

(1) Loosen both thumb screws.

(2) Pull firmly (5-10 lbs) on one thumb screw to tension the cable in the direction shown in Figure 18 and tighten the other thumb screw with the other hand.

(3) Release tension and tighten both thumb screws finger-tight. Do not use pliers.

# V STEPPER MOTOR CONTROL

## A. Fundamentals of Operation

Each of the cable drives is controlled by a stepper motor. The motors used have 4 coils, each driven by a power transistor. The drive is digital, with the transistors either turned on or turned off to obtain the desired pattern of currents in the motor windings. By changing the pattern of currents, a rotating magnetic field is obtained inside the motor. This causes the motor to rotate in small increments or steps. More information on stepper motor control can be found in References [3] and [5]. In the MiniMover-5, each of the four coils in a motor is individually controlled from the computer. The simplified drive circuit is shown in Figure 19. This allows the pattern of motor currents to be controlled by the computer software, simplifying the drive circuit and reducing its cost.

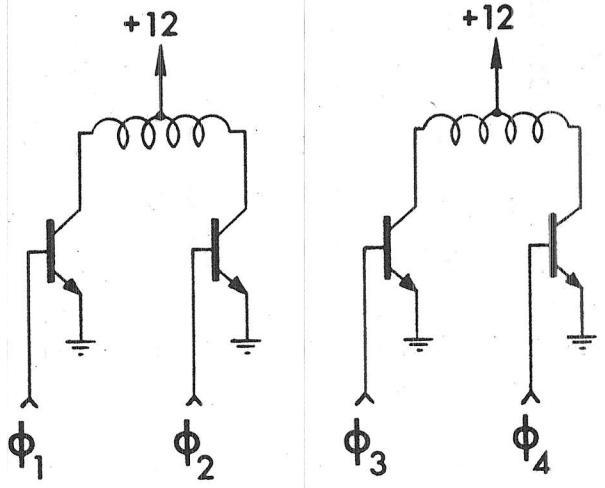



Figure 19 Simplified Stepper Motor Drive Circuit

In order to turn the motor, the sequence of binary phase patterns shown in Figure 20 are output to the desired motor, one pattern per step. To step the motor clockwise the patterns are output sequentially from top to bottom. When the end of the table is reached, it is necessary to wrap around to the other end of the table and continue sequentially. In order to change motor direction, the direction in which the entries are output must be reversed. The particular entries shown generate a sequence known as "half-stepping"; the steps are half of the size specified by the motor manufacturer. Compared to full stepping, half-stepping produces smoother slow-speed motions; doubles the resolution; and reduces the power required. The motors used to drive the MiniMover-5 are specified by the manufacturer at 48 steps per revolution and are stepped at 96 steps per revolution by the method described above.

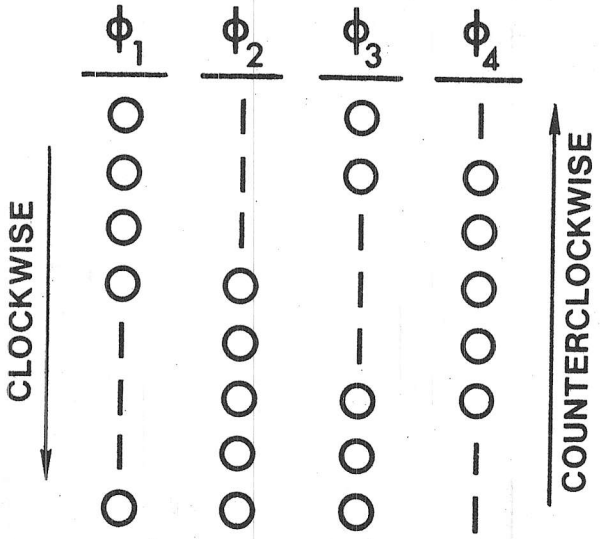



Figure 20 Binary Stepper Motor Phase Patterns

B. Speed-Torque Considerations

The torque output of the stepper motors used on the MiniMover-5 vary with their speed as shown in Figure 21. At slow speeds, maximum torque (lifting capacity) is obtained. Above a critical high speed, no torque is obtained. With intermediate speeds, intermediate torque loads can be maintained. Alternatively with any given torque load we can determine the critical speed above which the motor will not turn.

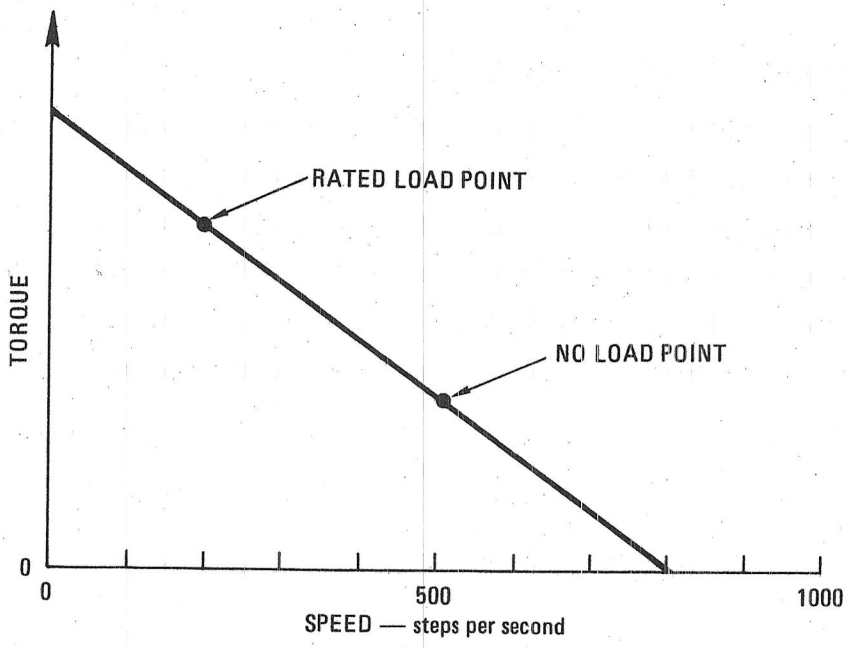



Figure 21 Stepper Motor Speed-Load Tradeoff

The torque required by the motors of the MiniMover-5 depends on the configuration of the arm and the load held in the hand. This relation is a complex trigonometric expression involving the lengths and weights of all the arm members. Instead of solving such an expression before each movement to determine the maximum speed of movement, it is simpler to program for the worst-case.

The worst case is when the members of the arm are at maximum horizontal extension, requiring the maximum motor torque. All other configurations will require less motor torque.

In the worst-case configuration the motor torque also depends on the load held in the hand. With no load, the torque on all the motors is the same (by design) and operation is at the no-load point shown in Figure 21. With the rated load (the arm lifting 8 ounces) the torque on all the motors except the base motor (which does not lift) is approximately equal, and operation is at the rated load point shown in Figure 21.

C. Recommended Operating Speeds

Speed control of the stepper motors can be simplified by considering only a few load conditions such as rated-, half-, and no-load as given in Table 3. With the given load, no slippage* will occur as long as these rates are not exceeded. Users can drive the arm at any speed they wish (including very slowly) but the arm will not always follow if the speed is above the maximum rates given.

Table 3

Maximum No-Slip Stepping Rates

| Load | Half Steps per second | Time Delay (ARMBASIC) |
|---|---|---|
| None | 525 | 20 |
| Half (4 oz.) | 370 | 50 |
| Rated (8 oz.) | 230 | 100 |

* Motor slippage will cause an error between where the arm is and where the computer program thinks it is, and may result in unpredictable performance.

In general we want to perform a task as fast as possible without risking slippage. The following suggestions may prove helpful:

* Lowering a load may be done at no-load speed if shoulder, elbow, and wrist all descend.
* Swiveling a load about the base joint only, may be done at no-load speed.
* Opening the hand may always be done at no-load speed.
* Closing the hand until contact may always be done at no-load speed.

Under the following load situations, special care must be exercised in selecting the proper speed:

* Raising a load with any joint.
* Developing a gripping force once contact has been detected.

# VI ELECTRONIC INTERFACE

## A. Overview

The MiniMover-5 interface has been developed to operate the manipulator from either Radio Shack's TRS-80 Level-II computer or from an 8-bit parallel I/O port (TTL levels). The same interface can be converted from one to the other by the user. Details are given in Section X-A. The theory of operation developed in this Section applies to both versions. A second arm can operate from the same computer by changing one jumper.

The interface is organized as seven, 4-bit parallel outputs and a single 4-bit input as shown in Figure 22. Information on the address lines is used to channel four bits of output data to the appropriate output latch. In this way the computer can control the 4-bit phase patterns on each motor drive with a minimum of hardware. After a phase pattern is sent to a 4-bit latch, it is maintained by the latch until the next pattern is sent. Outputs of the latches control the power driver IC's which drive the motors as discussed in Section V-A. The MiniMover-5 interface is shown in Figure 23

The grip switch is connected to one of the inputs of the auxiliary port, permitting the computer to sense the hand closure. The other inputs and outputs of the auxiliary port are available for experimentation with other sensors or controls as desired. The auxiliary inputs may be used for tactile, proximity, force and/or position sensors which can be mounted on the hand, arm extremities, and/or the table top. Auxiliary outputs may be used to control another stepper motor such as a work turntable (for example), external parts feeders, or to synchronize other equipment with arm operations.

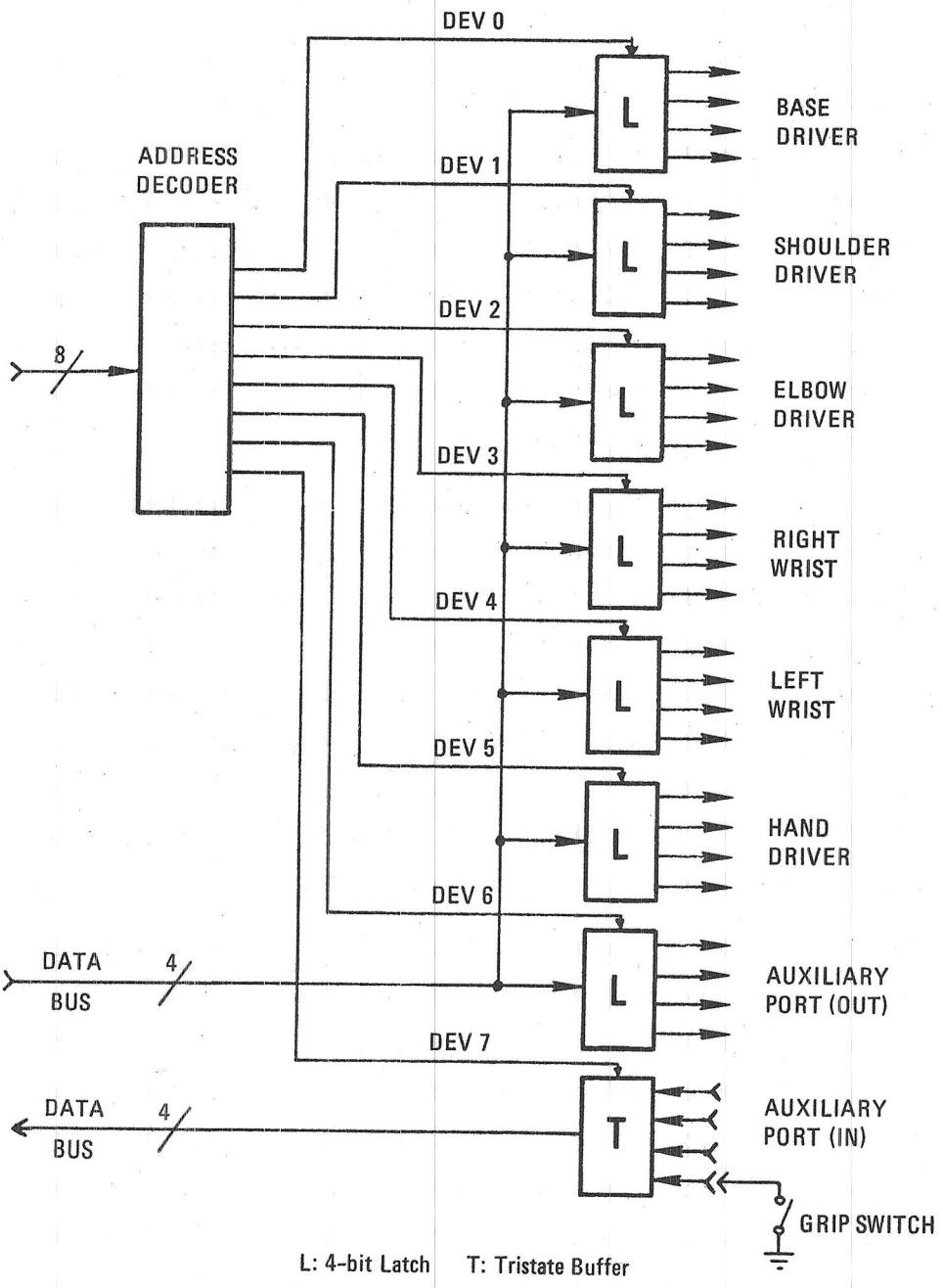



Figure 22 Organization of the Computer Interface

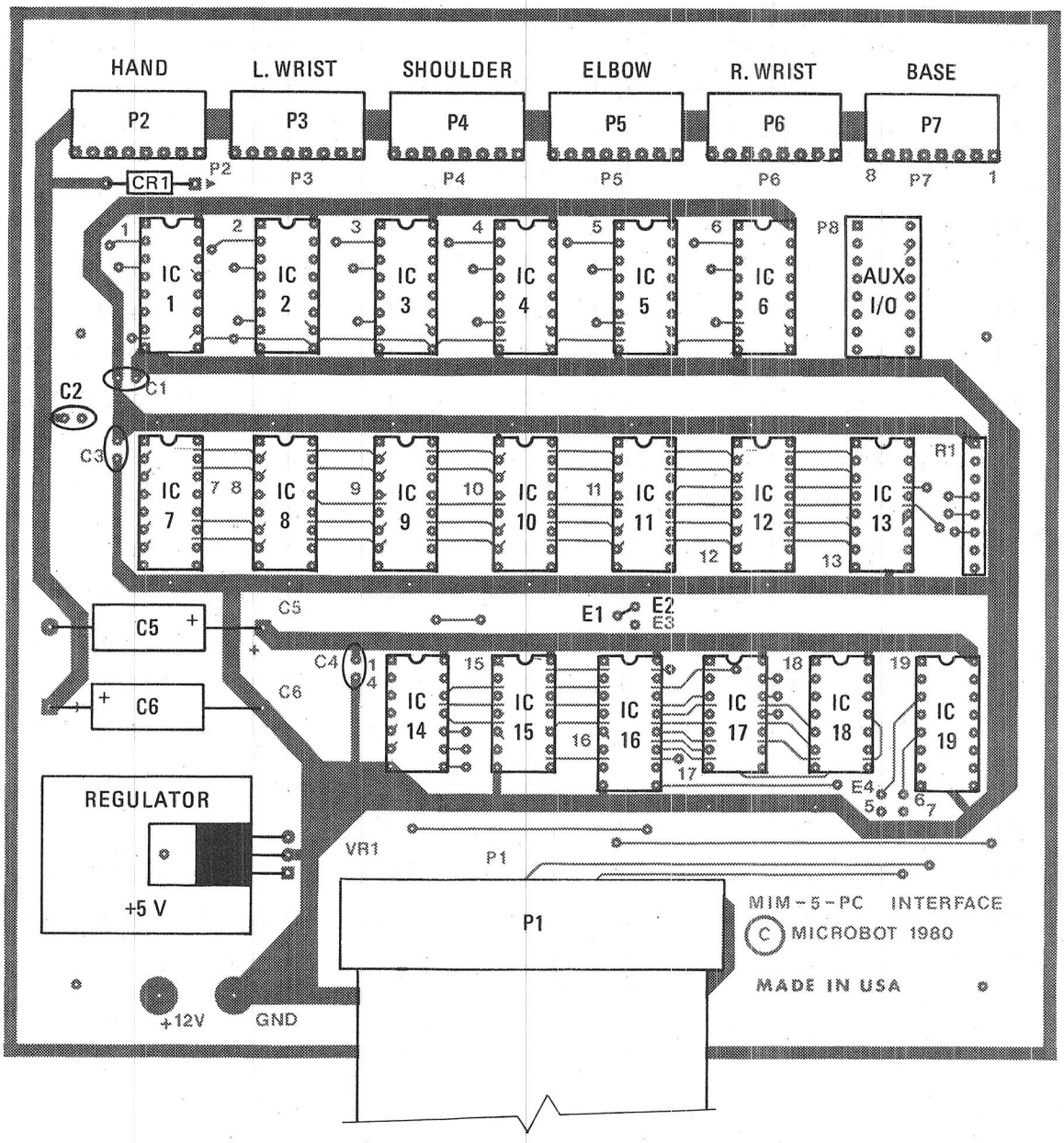



Figure 23 Interface Card Configured for the TRS-80

B. Address Decoding

The address decoding circuitry is shown in Figure 24 and is composed of the following TTL integrated circuits:

* IC17: 74LS00 Quad 2-input NAND gates
* IC16: 74LS138 1-of-8 Decoder/Demultiplexer
* IC14 & IC18: 74LS02 Quad 2-input NOR gates

Address lines A0-A2, which run to the decoder, specify which one of the eight decoder output lines will go low. These lines are, in turn, NOR'd with the negative going OUT pulse from the TRS-80 computer so that when one of the outputs from the decoder is low and the OUT pulse is present, then a positive going device select pulse results. These pulses (DEV0 to DEV6) are used to latch data into the 6 motor driver latches. The auxiliary input latch (DEV7) requires a negative going device select pulse. Finally, the OUT pulse is passed through two inverters to provide a double buffered signal of high drive capability.

During normal operation, the 1-of-8 decoder/demultiplexer is enabled by the signals on pins 4, 5, & 6 of IC16. In order for the address inputs A0-A2 to select one of the output lines, pins 4 & 5 must be low and pin 6 must be high. Any other condition will cause all of the outputs to remain high, regardless of the state of the address inputs. IC17 generates the appropriate decoder enable signals when address inputs A3-A7 match some preset value as determined by jumper J1.

Two sets of addresses are available to the user by changing J1*. This permits two arms to be operated by the same computer. To select the first set of addresses leave the jumper in the position shown in Figure 24 (connecting E1 and E2). A tabulation of the addresses for the MiniMover-5 with the jumper in this position is given below:

---

* J1 is actually an etched trace on the printed circuit card. In order to change J1, the trace must be cut and a new wire installed.

| Address | | | |
|---|---|---|---|
| Decimal | Octal | Hex | Port |
| 152 | 230 | 98 | Base Motor |
| 153 | 231 | 99 | Shoulder Motor |
| 154 | 232 | 9A | Elbow Motor |
| 155 | 233 | 9B | Wrist Right Motor |
| 156 | 234 | 9C | Wrist Left Motor |
| 157 | 235 | 9D | Hand Motor |
| 158 | 236 | 9E | Auxiliary 4-bit Output |
| 159 | 237 | 9F | Auxiliary 4-bit Input |

To use the second set of addresses, change J1 to connect E1 and E3. With the jumper in this position the addresses are:

| Address | | | |
|---|---|---|---|
| Decimal | Octal | Hex | Port |
| 184 | 270 | B8 | Base Motor |
| 185 | 271 | B9 | Shoulder Motor |
| 186 | 272 | BA | Elbow Motor |
| 187 | 273 | BB | Wrist Right Motor |
| 188 | 274 | BC | Wrist Left Motor |
| 189 | 275 | BD | Hand Motor |
| 190 | 276 | BE | Auxiliary 4-bit Output |
| 191 | 277 | BF | Auxiliary 4-bit Input |

C. Output Latches

The output latching circuitry is shown in Figure 25, and is composed of the following TTL integrated circuits.

* IC15: 74LS04 Hex Inverter
* IC7 to IC13: 74LS75 Quad Latch

The signals present on the least significant 4 bits of the TRS-80 data bus (D0-D3) are passed through inverters* to provide data signals

---

* Only four inverters are used for this purpose. The other two are used to buffer the OUT pulse as discussed in Section VI-B.

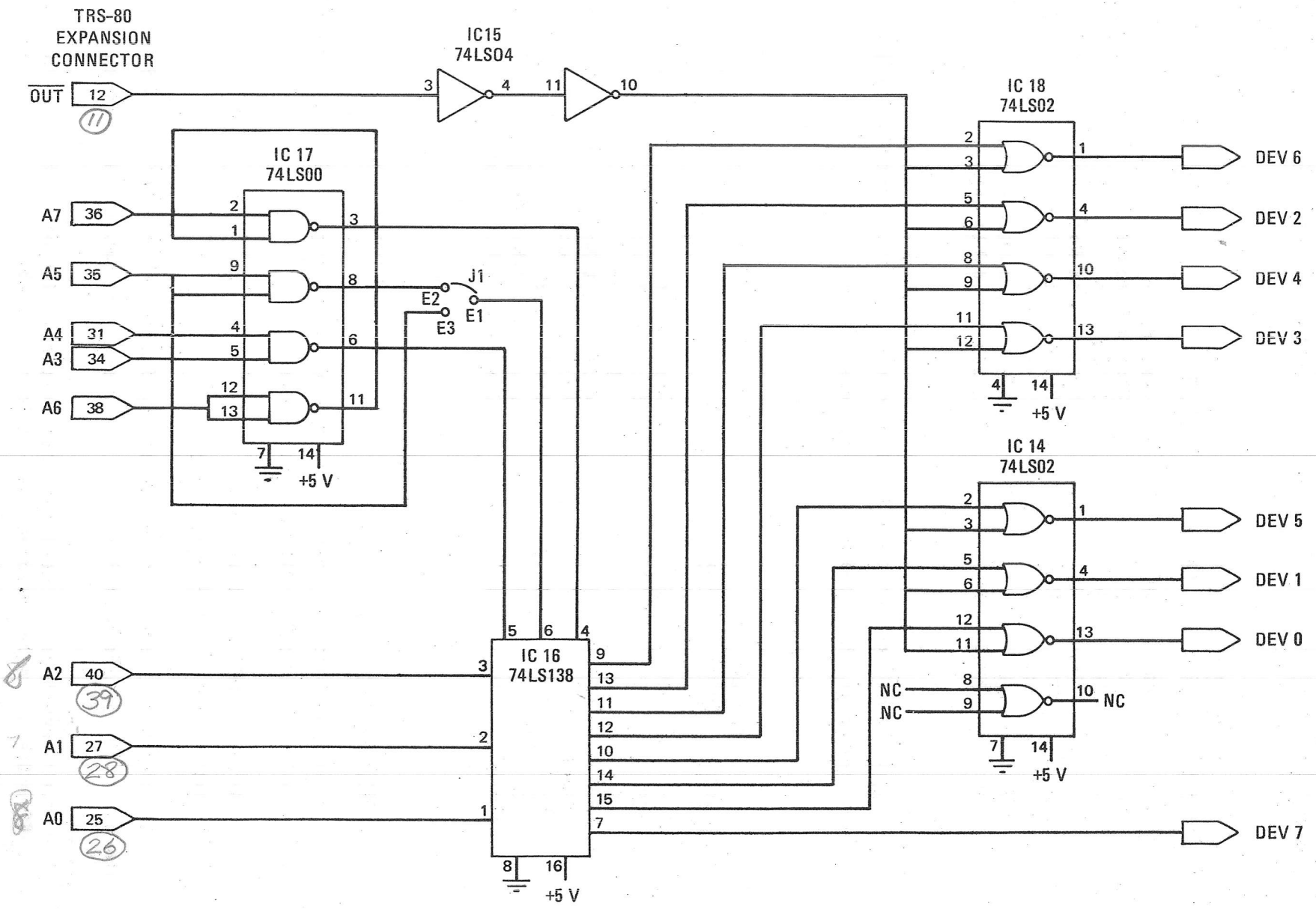



Figure 24 Address Decoding Schematic

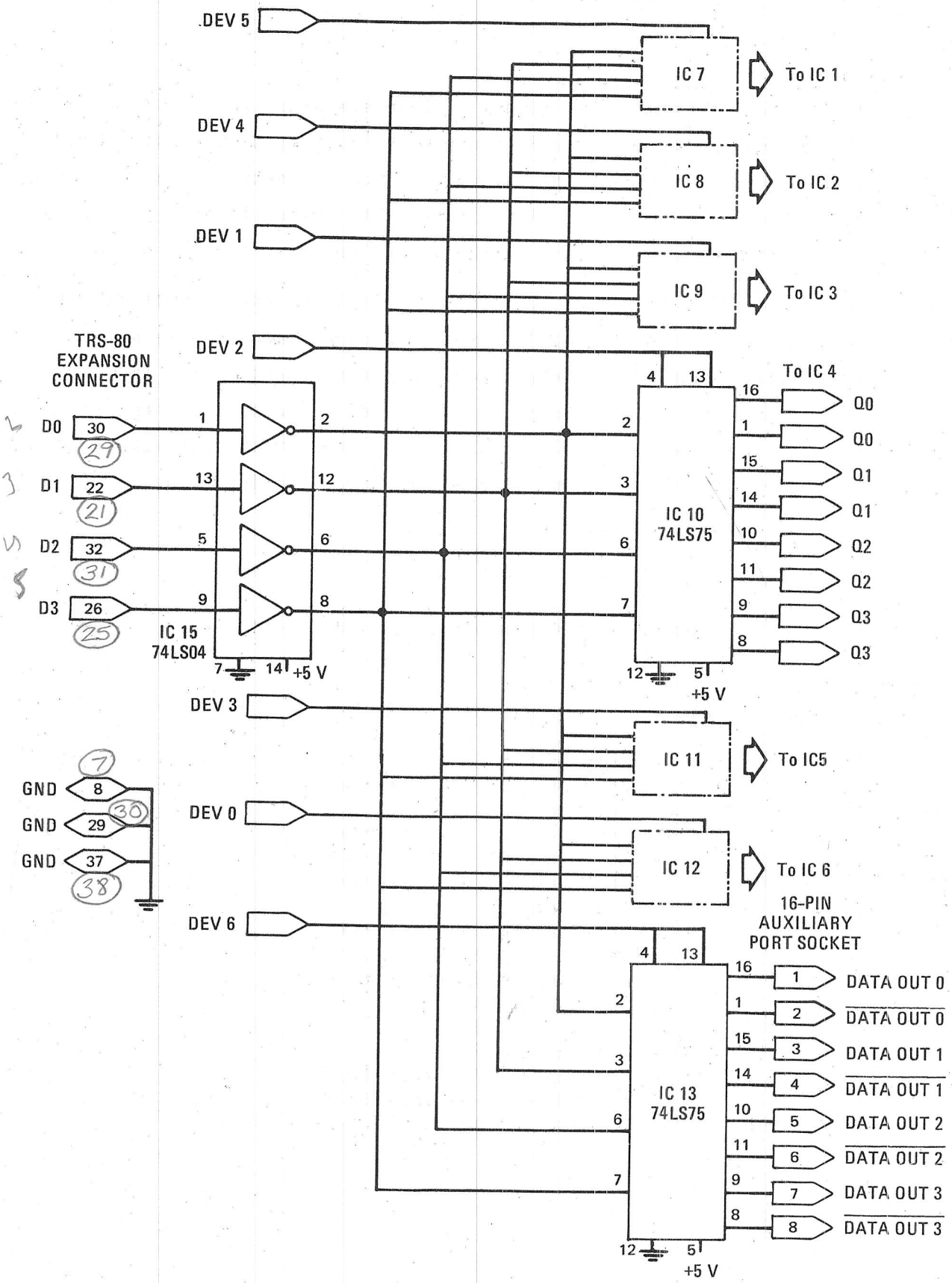



Figure 25 Output Latching Circuitry

of high drive capability. These signals are then routed to the inputs of each of the seven output latches. When a specific output latch is enabled (DEV 0 - DEV 6), the data present on the data bus is stored, and available at the output of that latch. The output latches provide both 4 bits of signal, and 4 bits of their complement.

All 8 lines from the auxiliary output latch are brought out on the auxiliary I/O connector (P8). Since the data pass through an inverter prior to arriving at the output latch the compliment of the data appears at the output. The user may compensate for this by using the inverted outputs from the latch on pins 2, 4, 6, and 8 of the auxiliary I/O connector. Auxiliary output connections are shown in Table 4.

Table 4

Auxiliary I/O Connector Pin Assignments (P8)

| Pin | | Function |
|---|---|---|
| 1 | -- | Data Out bit 0 |
| 2 | -- | Data Out bit 0 - Inverted |
| 3 | -- | Data Out bit 1 |
| 4 | -- | Data Out bit 1 - Inverted |
| 5 | -- | Data Out bit 2 |
| 6 | -- | Data Out bit 2 - Inverted |
| 7 | -- | Data Out bit 3 |
| 8 | -- | Data Out bit 3 - Inverted |
| 9 | -- | Data In bit 2 (from pad E4) |
| 10 | -- | Data In bit 3 (from pad E6) |
| 11 | -- | Data In bit 4 |
| 12 | -- | Data in bit 5 |
| 13 | -- | Data in bit 6 |
| 14 | -- | Data in bit 7 - Grip Switch |
| 15 | -- | Ground -------- Grip Switch |
| 16 | -- | 5 volts (50 ma. max) |

D. Auxiliary Input Port

The auxiliary input circuitry is shown in Figure 26 and is composed of the following components:

* IC19: 74LS366 Tri-State Hex Buffer
* R1: 8-pin Resistor Network

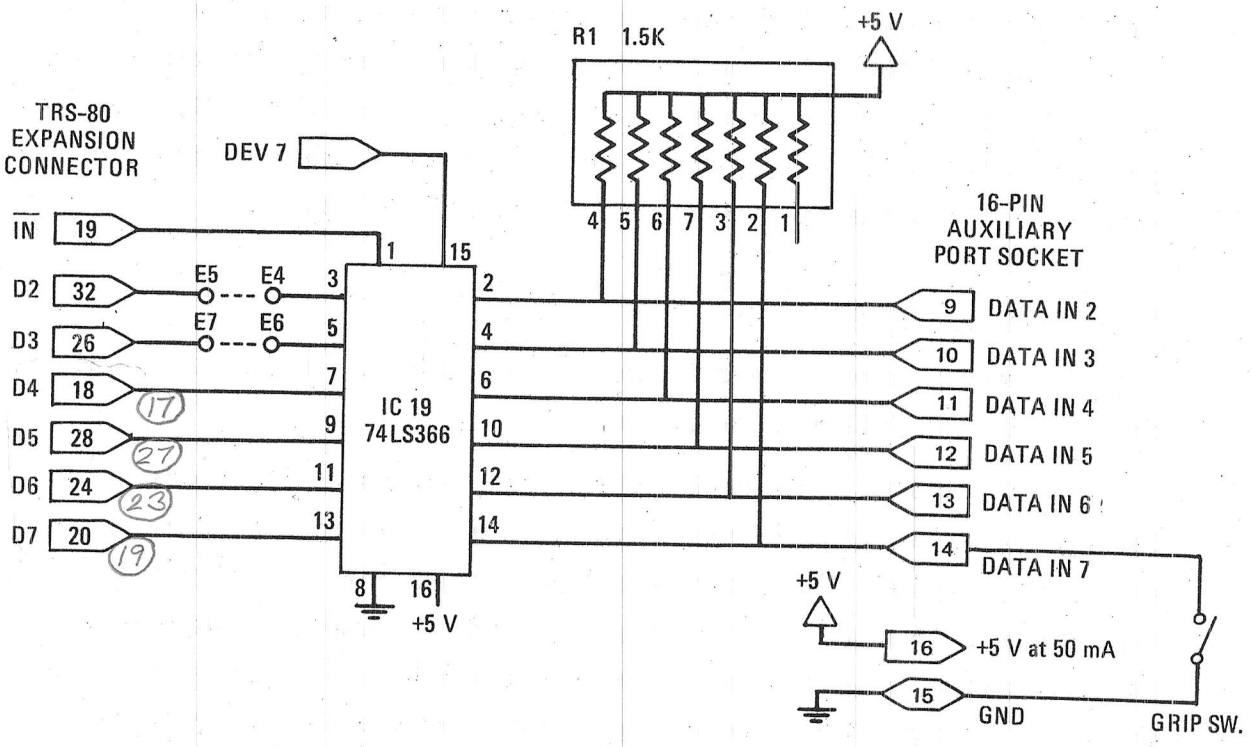



Figure 26 Input Port Circuitry

The input may be used to interface additional micro-switches to the TRS-80 computer. At present only one of the four inputs are used by the MiniMover-5. This input is the switch which indicates that the hand is grasping an object or is closed. The remaining three inputs are available to the user to provide inputs to the computer. Connections to the auxiliary inputs are made on connector P8*, as shown in Table 4.

* On the printed circuit card, the auxiliary I/O connector is a 16 pin DIP socket, and is identified as "Auxiliary I/O Port".

The grip switch is connected between pins 14 and 15. Pin 15 provides the ground to the switches.

Each of the input signals has a pull up resistor (R1) so that in the absence of an input, a high logic level is input to the computer. Any switches that the user desires to add should, when depressed, connect the input line to ground. Since the input buffer is an inverting one, the computer will read any of the activated inputs as a logical "1".

The input buffer is tri-state, such that when it is not being addressed, it remains in a high impedance state. This allows the other latches on the data bus to be active without any interference. When activated by the appropriate address select pulse (DEV 7), the latch goes into the low impedance state and presents the inverted data present on the inputs to the bi-directional computer data bus.

E. Motor Driver Circuits

A stepper motor driver circuit* is shown in Figure 27, and is composed of the following components:

* IC1 to IC6: Sprague UDN 5707A Quad 2-input NAND drivers

The quad 2 input NAND drivers connected as shown in Figure 27, prevent more than two of the motor windings from being on simultaneously, regardless of what value is output from the computer, or what state the latches assume at power-up. This prevents possible harm to the motors or power supply from overheating. More details on motor drivers was given in Section V.

---

* Only one is shown in the figure for clarity.

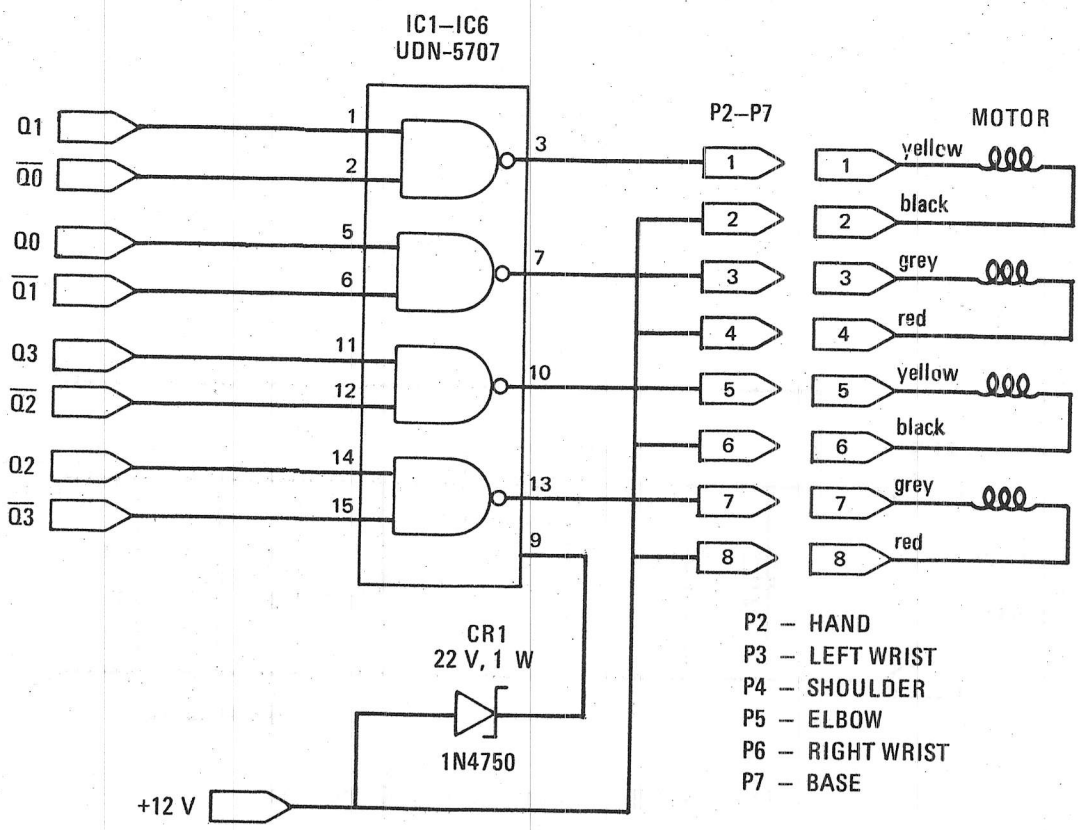



Figure 27 Stepper Motor Driver Schematic

F. Power Supply Circuit

The 5 volt regualator is shown in Figure 28 and is composed of the following components:

* 7805 voltage regulator
* 2-10$\mu$f filter capacitors
* 4-0.1$\mu$f filter capacitors

This circuit provides a regulated source of +5 volts to the on-board logic.

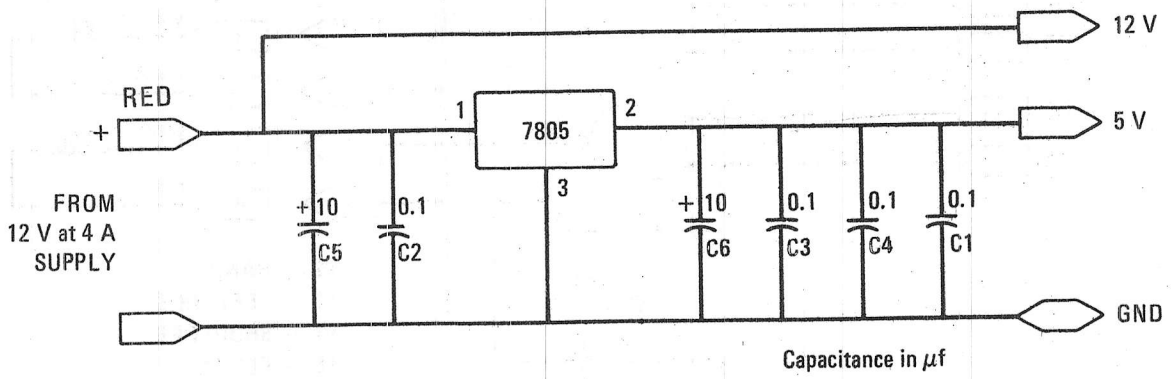



Figure 28 Power Supply Circuitry

# VII ARMBASIC

The MiniMover-5 ARMBASIC consists of 6 commands which allow complete control of the manipulator. These commands are:

* @STEP - moves the manipulator
* @CLOSE - closes the hand
* @SET - allows the manipulator to move under manual control
* @RESET - zeros the arm position and motor currents
* @READ - returns the current position of the manipulator
* @ARM - selects the port number which ARMBASIC will use

Each of the ARMBASIC commands is written in Z-80 assembly language both to conserve memory, and to provide a rapid response of the manipulator. The listings and an explanation of what the assembly language code does is discussed in Section XI.

## A. @STEP Command

The @STEP command causes all of the 6 stepper motors to move simultaneously. The syntax of this command is:

@STEP <D>,<J1>,<J2>,<J3>,<J4>,<J5>,<J6>

where <D> is an expression for the delay between the pulses to the motors, and <J1> to <J6> are expressions for the number of steps that each of the six motors is to be moved.

The delay expression <D>, evaluated to an integer, determines the speed of the motion. The smaller the delay, the faster the resulting motion will be. For a delay value of zero, the system will wait 1.2 milliseconds between steps. For each additional unit of delay, the system will wait an additional 0.03 milliseconds between pulses. This can be expressed as:

Delay = 1.2 + 0.03D (milliseconds)

The joint expressions, <J1>, <J2>,...., <J6>, evaluated to integers, determine the motion of each joint. After evaluation, the sign of each expression indicates the direction each motor should be driven; the magnitude indicates the number of steps. The variables J1 to J6 are as follows*:

* J1 - Base swivel
* J2 - Shoulder bend
* J3 - Elbow bend
* J4 - Right wrist
* J5 - Left wrist
* J6 - Hand

The string of joint variables may be truncated. For example, to omit any movement of the rest of the arm while driving the base, one would write:

@STEP 50,20,0,0,0,0,0

This is equivalent to:

@STEP 50,20

Alternatively, to omit any movement of the base while driving the other joints one would write:

@STEP 50,0,20,20,20,20,20

Notice that an argument is omitted by placing a "0" in the argument string.

Additionally, the @STEP command uncouples the elbow, <J3>, and the hand, <J6> (see Section IV-C), such that a command to move the elbow joint will not affect the hand opening. The user must combine the left and right wrist, <J4> and <J5>, to obtain the desired Pitch and Roll.

---

* Remember that Roll and Pitch movements are achieved by combined movements of the wrist left and wrist right motor.

Finally, the @STEP command linearly coordinates unequal joint variables to obtain coordinated motion. For example, if the base motor is told to move 21 steps, and the shoulder motor is told to move 3 steps, the resultant timing is illustrated in Figure 29. Additional information on the @STEP algorithm is given in Section XI.

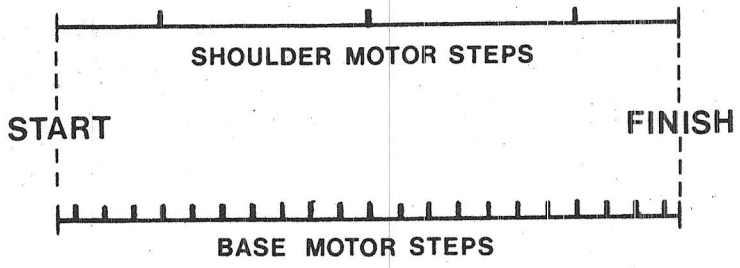



Figure 29 Step Timing Diagram

B. @CLOSE Command

The @CLOSE command causes the hand to close until the grip switch indicates that gripping force has built up*. The syntax of the @CLOSE command is:

@CLOSE <D>

The optional delay expression <D> determines the speed of closing, as discussed in the @STEP command.

C. @SET Command

This command puts the manipulator into "manual" mode such that, by pressing certain keys on the TRS-80 keyboard, each joint of the manipulator can be moved. The syntax of the @SET command is:

---

* This occurs either when the fingers close on an object, or when the fingers close and touch one another.

@SET <D>

The optional delay expression <D> determines the speed of motion as discussed in the @STEP command. The keys and the joints that they control are shown in Figure 30. This mode is terminated by depressing the "0" key (zero key). Note that although the @STEP command controls wrist left and wrist right motions, the @SET command allows Roll and Pitch movements of the wrist. All keys can be depressed simultaneously which will result in all joints moving simultaneously. Releasing the keys will stop the motion. If both keys of the same joint are depressed simultaneously, there will be no resulting motion as the commands will cancel one another.

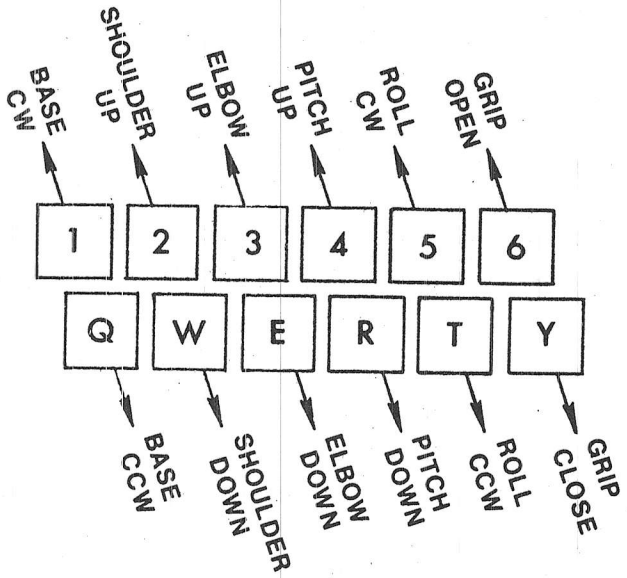



Figure 30 Use of Keyboard for Manual Control

D. @RESET and @READ Commands

As the previously described commands are executed, the number of steps commanded to each joint are counted and maintained in a set of six internal position registers. This permits the programmer to keep track of the commanded* position of the arm under both manual and program control. The @RESET and @READ commands provide access to these registers.

The syntax of the @RESET command is:

@RESET

This command zeros the contents of the internal position registers and the current of all six drive motors. Turning off the motors in this way allows the manipulator to be back driven**. @RESET is used for arm initialization, as discussed in the Section IX-A. The syntax of the @READ command is:

@READ <V1>,<V2>,<V3>,<V4>,<V5>,<V6>

where <V1> to <V6> are optional BASIC variables. This command reads the six position registers into these variables. Thus the current position of the manipulator is available to the user's basic program at any time.

E. @ARM Command

This command selects the port number to which any subsequent ARMBASIC commands will be addressed. The syntax of the @ARM command is:

@ARM <N>

The expression <N> must evaluate to an integer which has a value of one

---

* The commanded position and the actual position of the manipulator are identical, provided that the motors have not slipped, or provided that the arm has not bumped into an object.

** By this we mean that the arm can be moved by grabbing it and moving it around.

or two. If the expression evaluates to a "1", then all subsequent ARMBASIC commands will control the arm which is attached to port 152; if it evaluates to a "2", the arm attached to port 184 will be addressed. If this command is never issued, then ARMBASIC will assume that the arm is attached to port 152.

## F. ARMBASIC Summary

Each of the ARMBASIC commands may be executed immediately by typing them after the ">" prompt of the TRS-80. They may also be included in a BASIC program as numbered statements. A simple example of programming in ARMBASIC is given in Table 5. In this example the arm is moved manually from its starting position to a new position. As soon as control is returned to the program, the arm is moved back to its initial position by the computer. Numerous examples of programming techniques using ARMBASIC are given in Section IX.

Table 5

ARMBASIC Demonstration Program

```
10   @RESET
20   '
30   '
40   PRINT "YOU CAN CONTROL THE ARM USING THE KEYS:
50   PRINT "         1  2  3  4  5  6
60   PRINT "          Q  W  E  R  T  Y
70   PRINT "PRESS THE 0 KEY TO QUIT
80   @SET 200
90   '
100  '
110  @READ A,B,C,D,E,F
120  'THIS READS THE AMOUNT YOU MOVED
130  '
140  '
150  @STEP 200,-A,-B,-C,-D,-E,-F
160  'AND WE MOVE IT BACK TO WHERE IT WAS
170  '
180  '
190  @CLOSE
200  'THIS CLOSES THE JAW
210  '
```

## G. ARMBASIC Relocatability

When ARMBASIC is loaded into memory from cassette, a small relocation program is also loaded and started automatically. This program determines the size of the memory present in the system, and relocates ARMBASIC to the top 1000 bytes of real memory. Thus all memory is available to the user for applications programs.

Should users desire to save part of memory for any assembly language routines, they should respond to the MEMORY SIZE? prompt in the normal manner. When ARMBASIC is loaded, it will be relocated below the memory which was allocated with the MEMORY SIZE? command. The relocation will occur in blocks of 256 bytes. Thus, if 50 bytes was

allocated with the MEMORY SIZE? command, ARMBASIC will be relocated to memory which is 256 bytes from the top of memory, and 206 bytes of memory will be unavailable to the user.

# VIII COORDINATE CONVERSIONS

It is frequently advantageous to describe the configuration of the MiniMover-5 in different coordinate systems. The two most commonly used are *:

* Joint Coordinates. The joint angles of the arm are most convenient for controlling the arm directly from a computer.
* Cartesian Coordinates. The X, Y, Z location of the arm are more convenient for describing an assembly task on a flat table top.

For practical work we need a set of formulas for mathematically converting from one coordinate system to the other. These are called:

* Forward Solution converts from joint angles to Cartesian coordinates.
* Backward Solution converts from Cartesian coordinates to joint angles.

This section describes how both of these coordinate systems are defined and how the forward and backward solutions are derived and implemented.

To best understand the forward and backward solutions the reader should be familiar with basic trigonometry. It is not necessary to understand the solutions, however, to program the MiniMover-5. The material presented in part A of this section is sufficient for using the solution programs (in BASIC) given later in part D. An XYZ applications program given in Section IX-B show how the backward solution is incorporated into a program.

---

* Other coordinate systems commonly used are: hand coordinates with origin at the fingertips and axes aligned with the hand; workspace coordinates, located on, and aligned with a particular work station; moving coordinates located on, and aligned with a conveyor belt or turntable.

## A. Kinematic Model of Arm

Before we can formulate the arm solutions, the relationship between the different parts of the arm must be specified. This can be done in terms of the kinematic model shown in Figure 31. The kinematic model indicates how each joint is articulated, how the joint angles are measured, and the distance between joints.

The greek letter, $\theta$, is frequently used to indicate joint angles in mathematical expressions. The symbols $\theta_1$, $\theta_2$, $\theta_3$, $\theta_4$, and $\theta_5$, respectively, are proportional to the joint expressions $J_1$, $J_2$, $J_3$, $J_4$, and $J_5$ used in the ARMBASIC commands previously discussed in Section VII-A. The $\theta$s, measured in degrees or radians, are related to the J's, measured in motor steps, as shown in Table 6. There are 360 degrees or $2\pi$ radians* in one complete revolution.

Table 6

Conversion Factors Between Motor Steps and Revolute Joint Angles

| Motor | Joint | Steps in one Revolution | Steps per Radian | Steps per Degree |
|---|---|---|---|---|
| 1 | Base | 5893 | 937.9 | 16.37 |
| 2 | Shoulder | 5893 | 937.9 | 16.37 |
| 3 | Elbow | 3465 | 551.5 | 9.63 |
| 4 | Right wrist | 1280 | 203.7 | 3.56 |
| 5 | Left wrist | 1280 | 203.7 | 3.56 |

The distances between joints (segment lengths) are indicated by the constants H, L, and LL shown in Figure 31. H is the distance from the table top to the shoulder joint centerline, L is the distance from shoulder joint to elbow joint, and elbow joint to wrist joint, and LL is the distance from the wrist joint to the center point between the two fingertips. Values for these distances are given in Table 7.

---

* $\pi$ = 3.14159265....

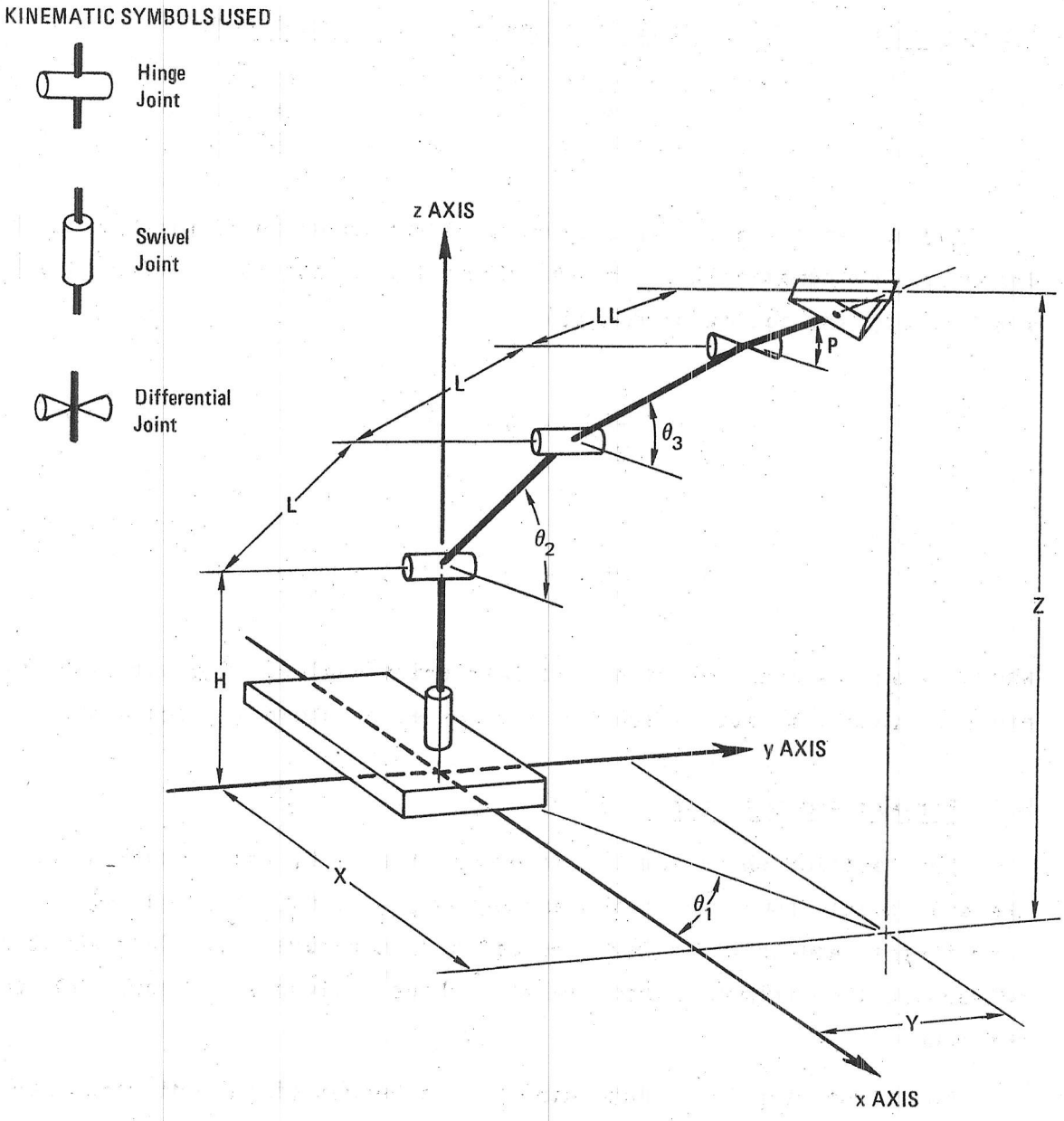



Figure 31 Kinematic Model of the MiniMover-5

Table 7

Lengths of MiniMover-5 Segments

| Segments | Length (inches) | Length (mm) |
|---|---|---|
| H | 7.68 | 195.0 |
| L | 7.00 | 117.8 |
| LL | 3.80 | 96.5 |

Motion at the differential wrist joint shown in Figure 32 can be described mathematically. The Pitch angle, P, and the Roll angle, R, are given by the following equations.

$$P = \frac{1}{2}(\theta_5 + \theta_4) \qquad (1)$$

$$R = \frac{1}{2}(\theta_5 - \theta_4) \quad . \qquad (2)$$

where $\theta_4$ and $\theta_5$ are the right and left wrist angles. The angles P, $\theta_4$, and $\theta_5$, are all measured from the horizontal as shown in Figure 32.

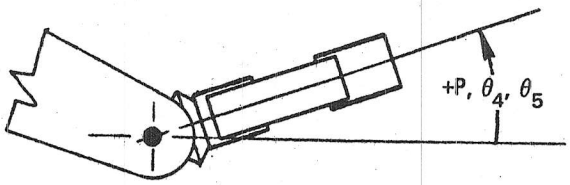
B. Forward Arm Solution

This section shows how to determine the X, Y, and Z coordinates of the end point from the joint angles $\theta_1$, $\theta_2$, $\theta_3$, $\theta_4$, and $\theta_5$. The coordinates and joint angles are defined in Figure 31. This solution relies on the trigonometric relationships* given in Figure 33 for reference.

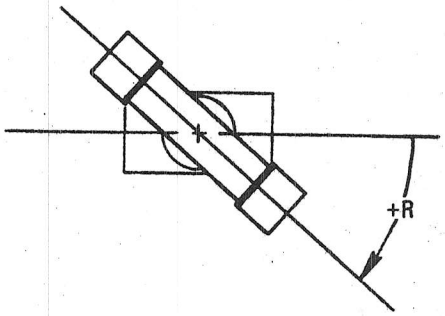
The first step is to determine Z, the height of the end point above the table top, and an intermediate variable RR, the horizontal distance

---

* Readers unfamiliar with trigonometry will find this material covered in basic trigonometry textbooks.




SIDE VIEW




END VIEW

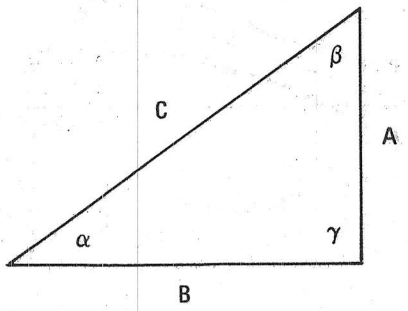
Figure 32 Definition of Roll and Pitch Angles

from the base pivot to the end point. The situation is summarized in Figure 34. Summing the vertical contributions from each link gives the following expression for Z:

$$Z = H + L \sin \theta_2 + L \sin \theta_3 + LL \sin P \quad . \qquad (3)$$

Summing the horizontal contributions gives:

$$RR = L \cos \theta_2 + L \cos \theta_3 + LL \cos P \quad , \qquad (4)$$




**ANGLE FORMULAS:**

$\alpha + \beta + \gamma = 180^\circ$

for a right triangle $\gamma = 90^\circ$

and $\alpha + \beta = 90^\circ$

**PYTHAGOREAN THEOREM:**

$C^2 = A^2 + B^2$, or

$C = \sqrt{A^2 + B^2}$ or $A = \sqrt{C^2 - B^2}$

**RATIOS OF SIDES:**

$\sin \alpha = \frac{A}{C}$ or $A = C \sin \alpha$

$\cos \alpha = \frac{B}{C}$ or $B = C \cos \alpha$

$\tan \alpha = \frac{A}{B}$ or $A = B \tan \alpha$

**ANGLE DEFINED BY INVERSE FUNCTION:**

$\alpha = \tan^{-1}\left(\frac{A}{B}\right)$

Figure 33 Review of Basic Trigonometry

where Pitch angle P is given by

$$P = \frac{1}{2}(\theta_5 - \theta_4) \quad . \qquad (5)$$

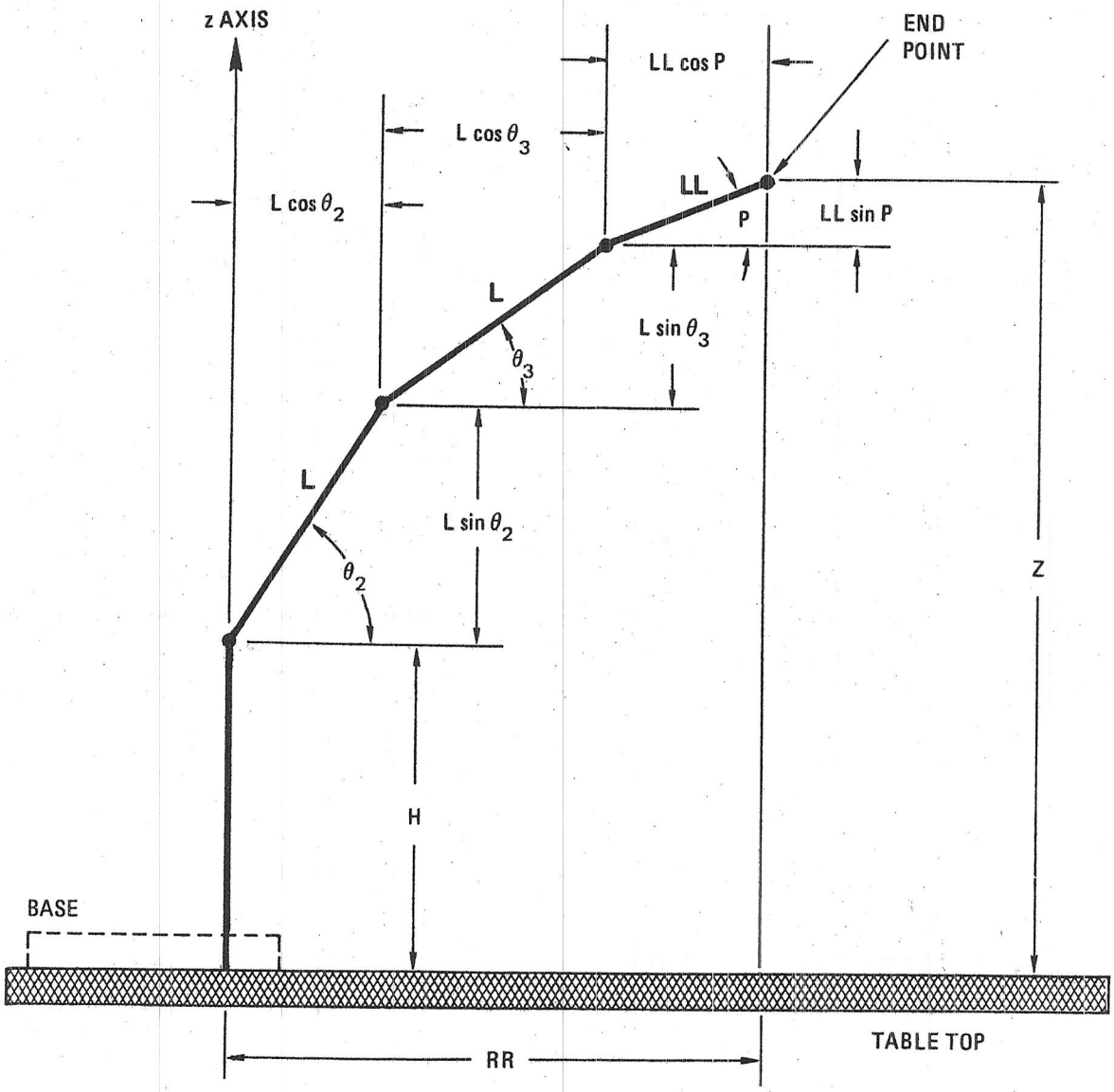



Figure 34 Side View of Kinematic Model

The second step is to determine the X and Y coordinates of the end point from intermediate variable, RR, as shown in Figure 35. By inspection, the coordinates are

$$X = RR \cos \theta_1 \qquad (6)$$

$$Y = RR \sin \theta_1 \quad . \qquad (7)$$

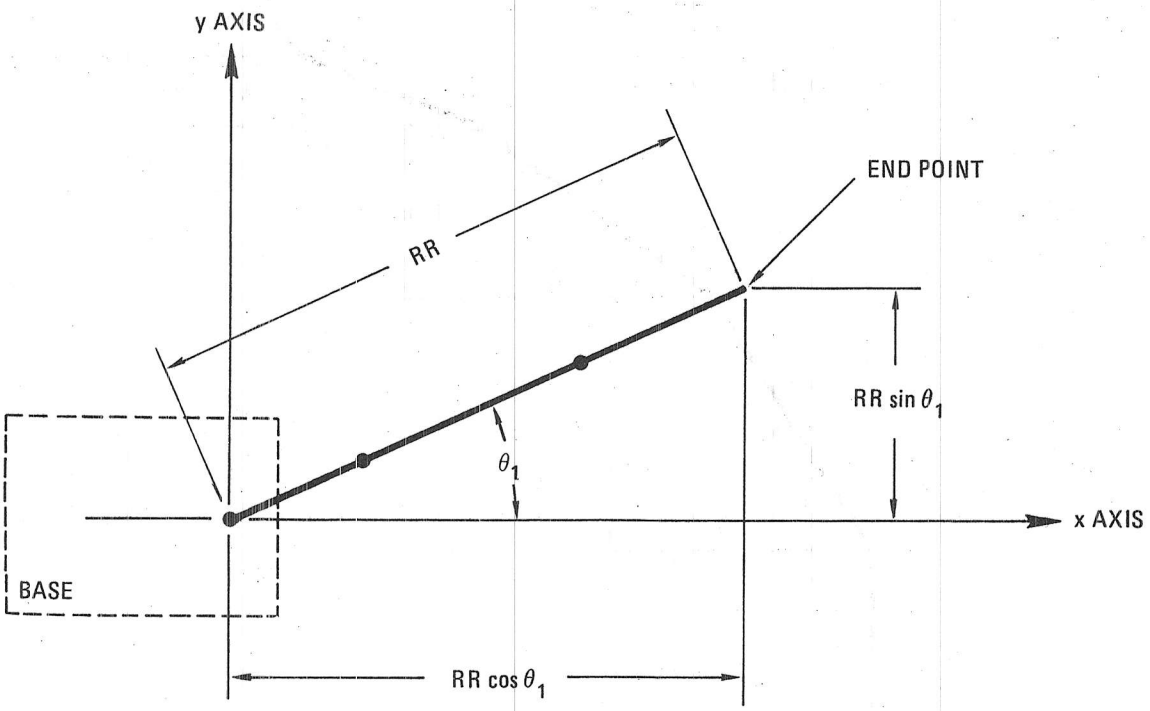



Figure 35 Top View of Kinematic Model

A Summary of this forward solution is given in Table 8. A BASIC program implementing this solution is given in Table 10 [Statements 460 to 510].

Table 8

Summary of Forward Solution

| Step | Operation |
|---|---|
| 1 | $P = ( \theta_5 + \theta_4 )/2$ |
| 2 | $R = ( \theta_5 - \theta_4 )/2$ |
| 3 | $RR = L \cos \theta_2 + L \cos \theta_3 + LL \cos P$ |
| 4 | $X = RR \cos \theta_1$ |
| 5 | $Y = RR \sin \theta_1$ |
| 6 | $Z = H + L \sin \theta_2 + L \sin \theta_3 + LL \sin P$ |

C. Backward Arm Solution

This section shows how to determine the joint angles $\theta_1$, $\theta_2$, $\theta_3$, $\theta_4$, and $\theta_5$ required to position the end point at a desired X, Y, Z position. The coordinates referred to are shown in Figure 31. A review of the formulas used is given in Figure 33.

1. Specifying Position/Orientation--X, Y, Z, P, and R

Before starting the backward solution it is necessary to specify the desired position and orientation of the end point*. The position of the end point is defined by the following three distances:

---

* The end point refers to the end point of the hand or alternatively, the center point between the two fingertips.

X The distance of the desired end point in front of the arm, measured from the base pivot along the X-axis.

Y The distance of the desired end point to the left of the arm, measured from the base pivot along the Y-axis.

Z The vertical height of the desired end point above the table top.

The units of these distances (inches or millimeters) should match the units of the segment lengths shown in Table 7.

The orientation at the end point is defined by the following two angles (see Figure 36):

P The Pitch angle desired, measured in degrees

R The Roll angle desired, measured in degrees

In practice it is difficult to distinguish between positive and negative angles (as $+90^{o}$ and $-90^{o}$, or $+45^{o}$ and $-135^{o}$) by looking at the hand. It is helpful to mark the top of the hand when it is at $0^{o}$ to eliminate this ambiguity.

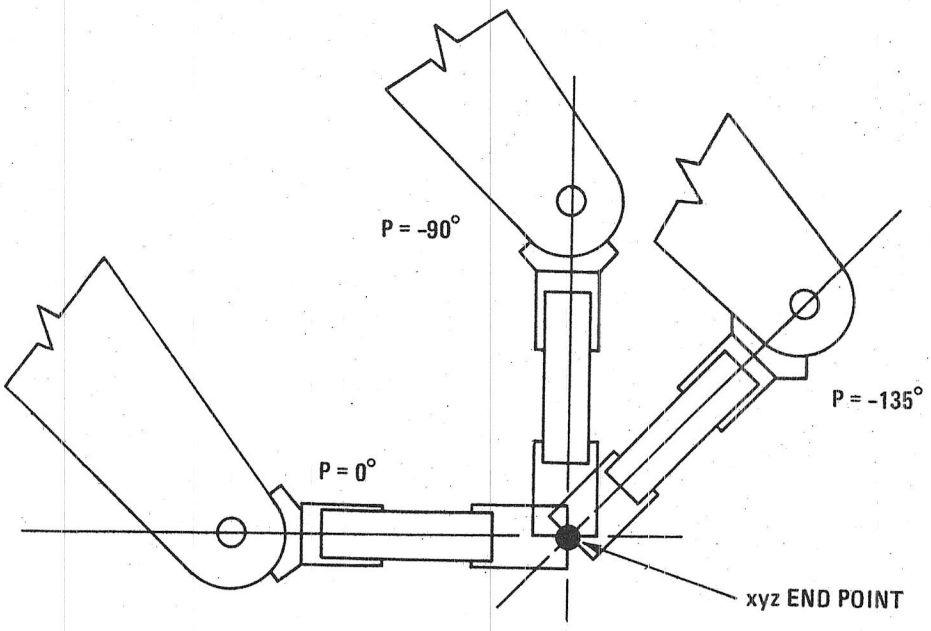
Since the MiniMover-5 does not have a Yaw * motion at the wrist, only limited orientations of the hand in space can be obtained. For example, the hand can only point forward, along the X-axis when the entire arm is positioned forward ($\theta_1 = 0^{o}$).

2. Specifying Roll in Cartesian Frame--R1

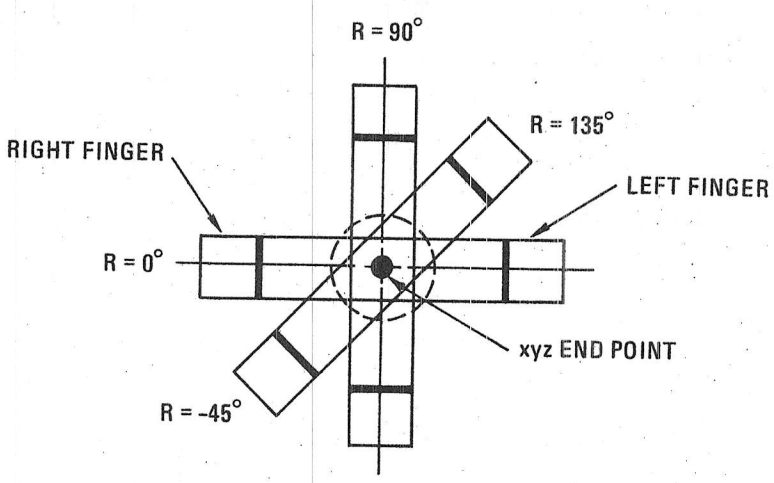
A special and useful case occurs when the hand is pointing straight down ($P = -90^{o}$). In this case Roll can be measured with respect to the arm (R as previously used) or to the Cartesian frame (R' as explained here).

---

* The terms Roll, Pitch, and Yaw are derived from the corresponding motions of aircraft.




(a) DIFFERENT PITCH ANGLES AT SAME ENDPOINT




(b) DIFFERENT ROLL ANGLES AT SAME ENDPOINT.
View looking into front of hand along pitch vector.

Figure 36 Different Hand Orientations

With the hand pointing straight down, subtracting $\Theta_1$ from the commanded Roll angle keeps the hand orientation fixed along the coordinate axes as shown in Figure 37.

In summary:

$$R' = R - \theta_1 \quad . \tag{8}$$

Examples:

$R' = 0^o - \theta_1$ Keeps hand aligned to Y-axis.

$R' = 90^o - \theta_1$ Keeps hand aligned to X-axis.

In the backward solution we introduce a special variable, R1, to differentiate between normal Roll, R, and Cartesian Roll R' as follows:

R1 = 1 For Roll in Cartesian frame. Roll angle is interpreted as hand angle to Y-axis.

R1 = 0 For Roll in arm frame. Roll angle is interpreted as Y hand angle to wrist axis.

With this new variable, Equation (8) can be modified to express both normal and Cartesian Roll as follows:

$$R' = R - \theta_1 R1 \quad . \tag{9}$$

Solving for R gives:

$$R = R' + \theta_1 R1 \quad . \tag{10}$$

3. Backward solution, Step-by-Step

The first step of the backward solution is to determine the base angle, $\theta_1$, and the radius vector, RR, from the base to the end point as shown in Figure 38. Using the Pythagorean Theorem

$$RR = \sqrt{X^2 + Y^2} \qquad (11)$$

$$\theta_1 = \tan^{-1}(Y/X) \quad . \qquad (12)$$

The second step is to find $\theta_4$ and $\theta_5$ from P and R. Using Equation (1) and Equation (2) of the wrist differential previously described, and substituting $(R' + \theta_1 R1)$ for R using Equation (10) gives:

$$\theta_4 = P + R' + \theta_1 R1 \qquad (13)$$

$$\theta_5 = P - R' - \theta_1 R1 \quad . \qquad (14)$$

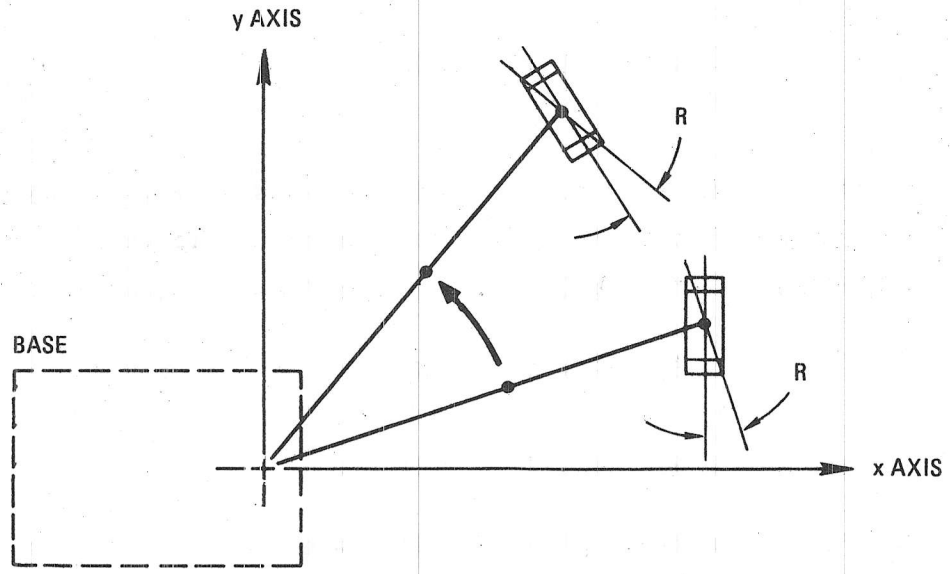
Note that these equations give arm Roll when R1 = 0, because the last term of each equation reduces to zero.

The third step is to work back from the coordinates of the end point to those of the wrist. As in the forward solution, we use the side view of the kinematic model shown in Figure 34. Distances in this view are measured vertically along the Z axis and horizontally along the radius from the base (R axis). Letting $R_e$ and $Z_e$ be the coordinates of the end point in this plane, we can calculate the coordinates of the wrist ($R_w$ and $Z_w$) by using the triangle shown in Figure 39. From this triangle the coordinates of the wrist are:

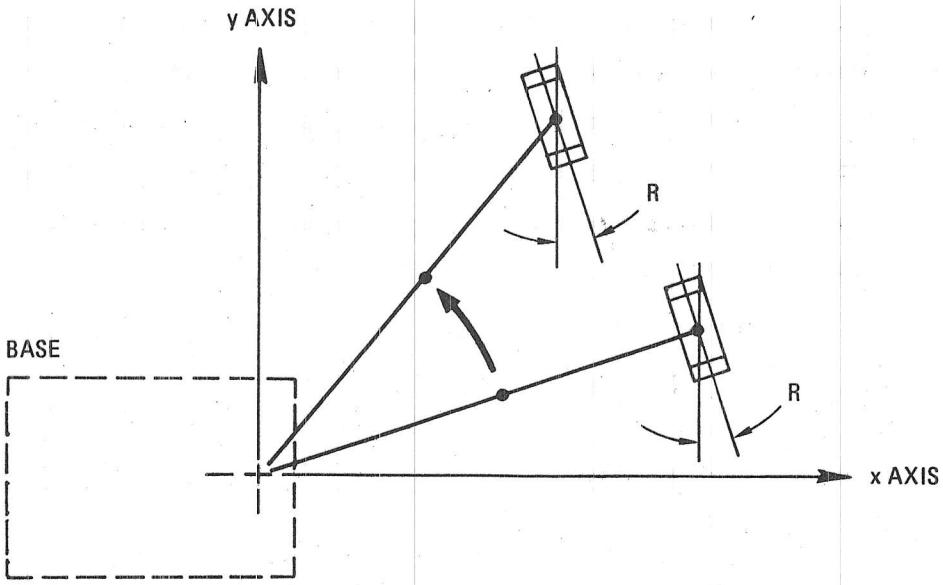
$$R_w = R_e - LL \cos P \qquad (15)$$

$$Z_w = Z_e - LL \sin P \quad . \qquad (16)$$

The fourth step is to define the shoulder-elbow-wrist triangle so that $\theta_2$ and $\theta_3$ can be determined. For this purpose, the translated coordinate system introduced in Figure 40 is used. The origin (0, 0) is at the shoulder and the coordinates of the wrist are now $(R_0, Z_0)$.




(a) R1 = 0, HAND ALIGNED TO ARM




(b) R1 = 1, HAND ALIGNED TO y AXIS

Figure 37 Roll in Arm Frame (a) and Cartesian Frame (b)

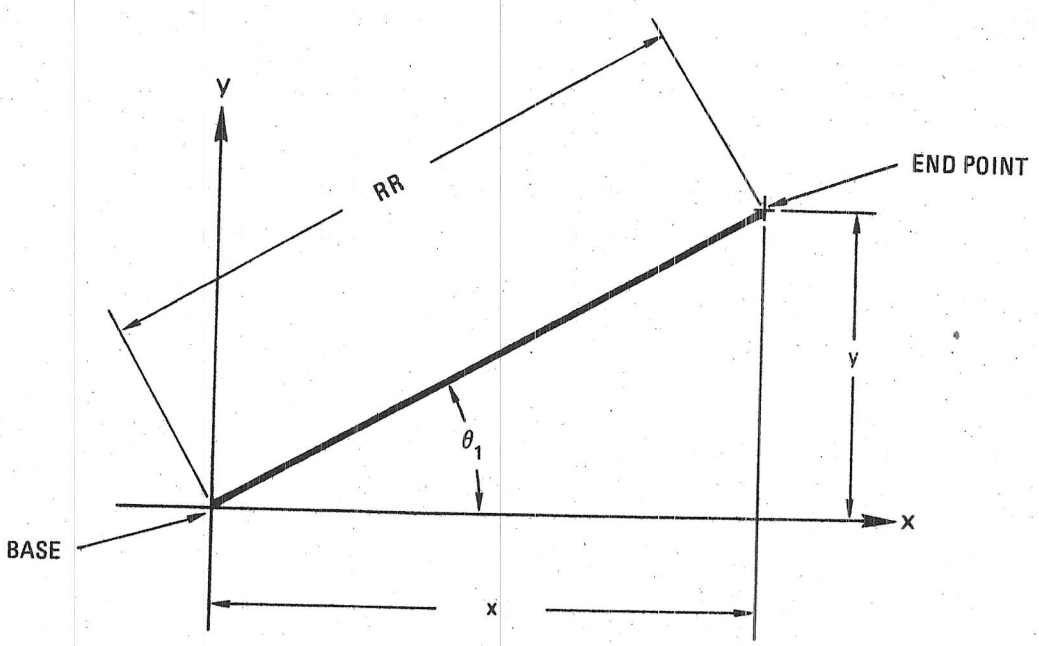



Figure 38 Top View of Arm

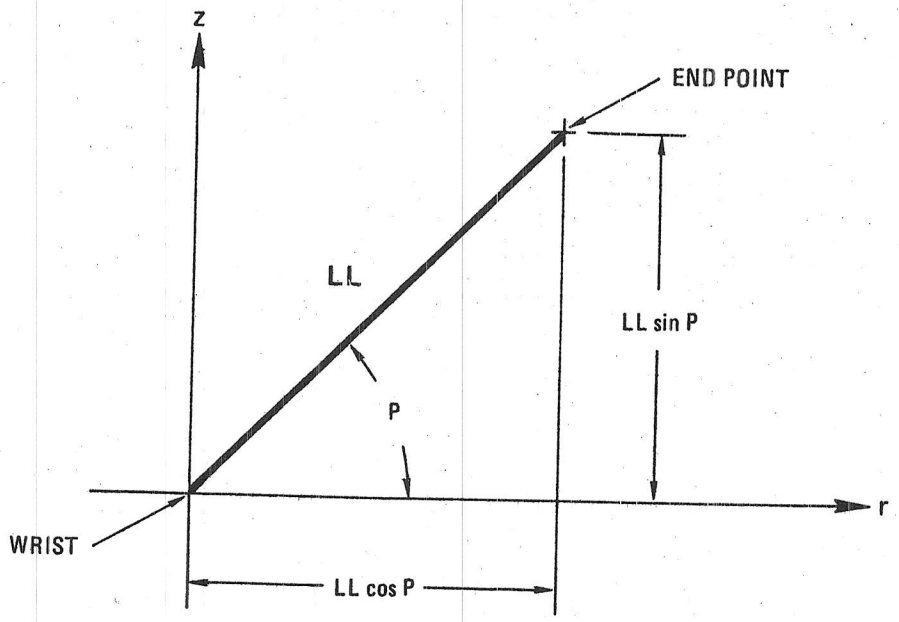



Figure 39 Side View of Hand Triangle in Kinematic Model

The distance from the shoulder to the wrist, $R_0$, is the same as $R_w$ previously determined in Equation (15). This is expressed as:

$$R_0 = R_e - LL \cos P \quad . \tag{17}$$

The height of the wrist above the shoulder, $Z_0$, is just the height of the wrist above the table top, $Z_w$, less the height of the shoulder, H. Thus,

$$Z_0 = Z_w - H \quad . \tag{18}$$

Substituting for $Z_w$ using Equation (16) gives:

$$Z_0 = Z_e - LL \sin P - H \quad . \tag{19}$$

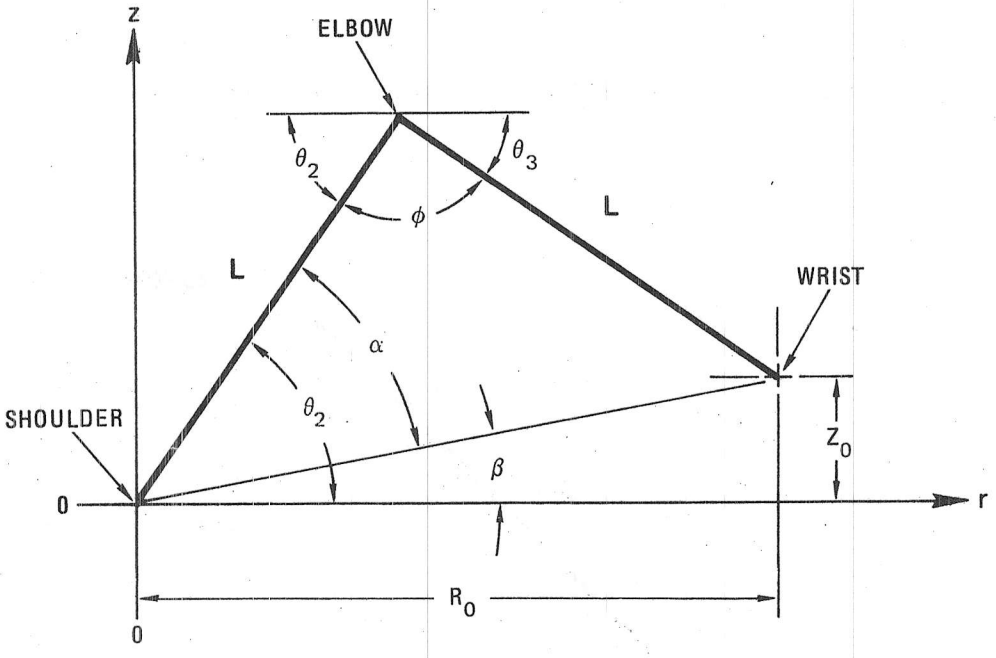



Figure 40 Shoulder-Elbow-Wrist Triangle

The fifth step is to solve the shoulder-elbow-wrist triangle for $\theta_2$ and $\theta_3$. Three new angles, $\alpha$, $\beta$, and $\phi$, are introduced to simplify this solution. We first solve for $\alpha$, $\beta$, and $\phi$.

Since $\tan\beta = (Z_0/R_0)$, we obtain:

$$\beta = \tan^{-1}(Z_0/R_0) \quad . \qquad (20)$$

Pivoting the shoulder-elbow-wrist triangle about the shoulder by $\beta$ gives the simplified triangle shown in Figure 41. The simplified triangle can be partitioned into two right triangles back-to-back *. The length of each base, b, is half that of the total base, $\sqrt{Z_0^2 + R_0^2}$ (using the Pythagorean theorem), or:

---

* The two right triangles are similar because they have equal angles, (equal angles opposite equal sides) and h is a perpendicular bisector of the apex angle, $\phi$.

$$b = \frac{1}{2}\sqrt{Z_0^2 + R_0^2} \quad . \tag{21}$$

The height, h, (using the Pythagorean Theorem) is

$$h = \sqrt{L^2 - b^2} \quad . \tag{22}$$

Since the tangent of $\alpha$ is h/b,

$$\alpha = \tan^{-1}(h/b) \quad . \tag{23}$$

Substituting for h in Equation (23) by using Equation (22) gives

$$\alpha = \tan^{-1}\frac{\sqrt{L^2 - b^2}}{b} \quad . \tag{24}$$

Substituting for b in Equation (24) using Equation (21) gives

$$\alpha = \tan^{-1}\sqrt{\frac{4\ L^2}{R_0^2 + Z_0^2} - 1} \quad . \tag{25}$$

The sixth step is to use $\alpha$ and $\beta$ to determine $\theta_2$ and $\theta_3$. The following three relations are first set up and then solved. At the shoulder (see Figure 40)

$$\theta_2 = \alpha + \beta \quad . \tag{26}$$

At the elbow apex (see Figure 40)

$$\theta_2 + \phi + \theta_3 = 180^\circ \quad . \tag{27}$$

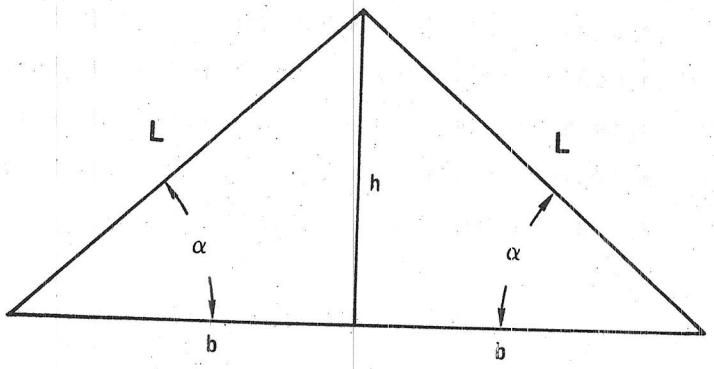



Figure 41 Simplified Triangle

Summing the internal angles of the simplified triangle (see Figure 41) gives $\phi + \alpha + \alpha = 180^{o}$ or

$$\phi = 180^{o} - 2\alpha \quad . \tag{28}$$

Substituting the value of $\theta_2$ from Equation (26) and the value of $\phi$ from Equation (28) into Equation (27) gives

$$\theta_3 = \alpha - \beta \quad . \tag{29}$$

Note however, that the elbow angle, $\theta_3$, is defined as the angle above the horizontal and hence we must change the sign of $\theta_3$.

In summary, the results of the sixth step are:

$$\theta_2 = \alpha + \beta \tag{30}$$

$$\theta_3 = \beta - \alpha \quad . \tag{31}$$

thus completing the backward solution. A summary of the backward solution is given in Table 9.

A BASIC implementation of the backward solution is given in Table 10 [Statements 130 to 370]. The backward solution may be compared to other solutions by the time and number of functions required to perform it. This solution requires 12 additions or subtractions, 10

multiplications, 3 divisions, 3 arc tangents, 2 sines or cosines, and 2 square roots for a total of 32 operations. In BASIC, on the TRS-80, it takes less than one second. Precision from the 6-digit floating point operations is quite adequate. Those interested in more speed might try using integer mathematics and looking up trigonometric functions which are stored in tables, rather than calculating them.

Table 9

Summary of Backward Solution

| Step | Operation |
|---|---|
| 1 | Determine arm constants H, L, LL |
| 2 | Determine X, Y, Z, R, P, and R1 |
| 3 | $\theta_1 = \tan^{-1}(Y/X)$ |
| 4 | $RR = \sqrt{X^2 + Y^2}$ |
| 5 | $\theta_4 = P + R + R1\ \theta_1$ |
| 6 | $\theta_5 = P - R - R1\ \theta_1$ |
| 7 | $X_0 = RR - LL \cos P$ |
| 8 | $Y_0 = Z - LL \sin P - H$ |
| 9 | $\beta = \tan^{-1}(Z_0/R_0)$ |
| 10 | $\alpha = \tan^{-1} \sqrt{4L^2/(R_0^2 + Z_0^2) - 1}$ |
| 11 | $\theta_2 = \alpha + \beta$ |
| 12 | $\theta_3 = \beta - \alpha$ |

## D. BASIC Implementation of Arm Solutions

An implementation of both the forward and backward solutions in the form of a test program written in BASIC is given in Table 10. The joint angles T1, T2, T3, T4, and T5 correspond to the O's used in the previous equations.

Note that several tests have been included in the backward solution to determine if the position requested can be reached by the arm. Such tests or "software limits" are advisable in application programs to keep the arm from hitting its end stops and loosing calibration.

The user may wish to add limits for their own particular situation. For example, when operating on a table top, requests for movement with $Z \leq 0$, (beneath the table top) would be refused. If obstacles in the working area can be described mathematically, collisions with them can also be avoided. The forward solution does not have such a limitation: If the angles are known, the Cartesian position can always be determined.

Table 10

BASIC Implementation of Forward and Backward Solutions

```
10   REM DEFINE CONSTANTS
20   REM
30   H=7.68:  REM SHOULDER HEIGHT ABOVE TABLE
40   L=7.00:  REM SHOULDER TOO ELBOW & ELBOW TO WRIST LENGTH
50   LL=3.8:  REM WRIST TO FINGERTIP LENGTH
60   PI=3.14159
70   C=180/PI:  REM DEGREES IN 1.0 RADIAN
80   REM   ************************************************************
90   REM   *                                                          *
100  REM   * CARRY OUT THE BACKWARD SOLUTION CHECKING ALL JOINT LIMITS *
110  REM   *                                                          *
120  REM   ************************************************************
130  PRINT "ENTER X, Y, Z, PITCH, ROLL, AND R1"
140  INPUT X,Y,Z,P,R,R1
150  REM CONVERT TO RADIANS
160  P=P/C:  R=R/C
170  RR=SQRT(X*X+Y*Y)
180  IF RR<2.25 THEN GOTO 5000:  REM MIN END POINT DISTANCE TO BODY
190  IF X=0 THEN T1=SGN(Y)*PI/2 ELSE T1=ATN(Y/X)
200  IF X<0 THEN GOTO 5000:  REM CANNOT REACH BEHIND BASE PIVOT
210  T5=P+R+R1*T1
220  T4=P-R-R1*T1
230  REM MAX WRIST DIFFERENTIAL ROTATION 270 DEGREES
240  IF T4 < -270/C OR T4 > 270/C THEN GOTO 5000
250  IF T5 < -270/C OR T5 > 270/C THEN GOTO 5000
260  R0=RR-LL*COS(P)
270  Z0=Z-LL*SIN(P)-H
280  IF R0 < 2.25 THEN GOTO 5000:  REM MIN WRIST DISTANCE TO BODY
290  IF R0 = 0 THEN B=SGN(Z0)*P1/2 ELSE B=ATN(Z0/R0)
300  A=R0*R0 + Z0*Z0
310  A= 4*L*L/A - 1
320  IF A < 0 THEN GOTO 5000:  REM PAST MAX REACH OF ARM
330  A=ATN(SQRT(A))
340  T2=A+B
350  T3=B-A
360  IF T2>144/C OR T2<-127/C THEN GOTO 5000:  REM SHOULDER RANGE
370  IF T2-T3<0 OR T2-T3>149/C THEN GOTO 5000:  REM ELBOW RANGE
380  REM
390  PRINT "T1,T2,T3,T4,& T5 (IN DEGREES)"
400  PRINT T1*C;T2*C;T3*C;T4*C;T5*C
```

```
410  REM ************************************************************
420  REM * NOW CHECK OUT THE BACK SOLUTION BY FINDING OUT WHERE WE ARE.*
430  REM *                                                           *
440  REM *                    THE FORWARD SOLUTION                   *
450  REM ************************************************************
460  P=(T5+T4)/2
470  R=(T5-T4)/2-R1*T1
480  RR=L*COS(T2)+L*COS(T3)+LL*COS(P)
490  X=RR*COS(T1)
500  Y=RR*SIN(T1)
510  Z=H+L*SIN(T2)+L*SIN(T3)+LL*SIN(P)
520  PRINT "X,Y,Z,P,R";X,Y,Z,P*C,R*C
530  GOTO 130
5000  PRINT "*************** OUT OF REACH ***************
5010  GOTO 130
5020  END
```

E. Kinematic Model of the Hand

The opening of the hand is proportional to the number of steps of the hand drive motor. The constant of proportionality is:

$$S_6 = 309 \text{ steps/inch } (12.2 \text{ steps/mm}).$$

Although the length of the hand, LL, has been treated as a constant in the previous calculations, it varies slightly with hand opening, as shown in Figure 42. The effect is small, $\pm$ 0.10 in ($\pm$ 2.5 mm), but for more precise work it may be necessary to take this into account.

The hand length, LL, may be expressed as the sum of a fixed length, $L_1$, and a varying length that depends on hand opening, G, by the following formula:

$$LL = L_1 + \sqrt{L_2^2 - (G - G_0^2)/2} \quad , \qquad (32)$$

where:

$L_1$ = 2.20 in (55.9 mm)

$L_2$ = 1.70 in (43.2 mm)

$G_0$ = 1.52 in (38.6 mm) .

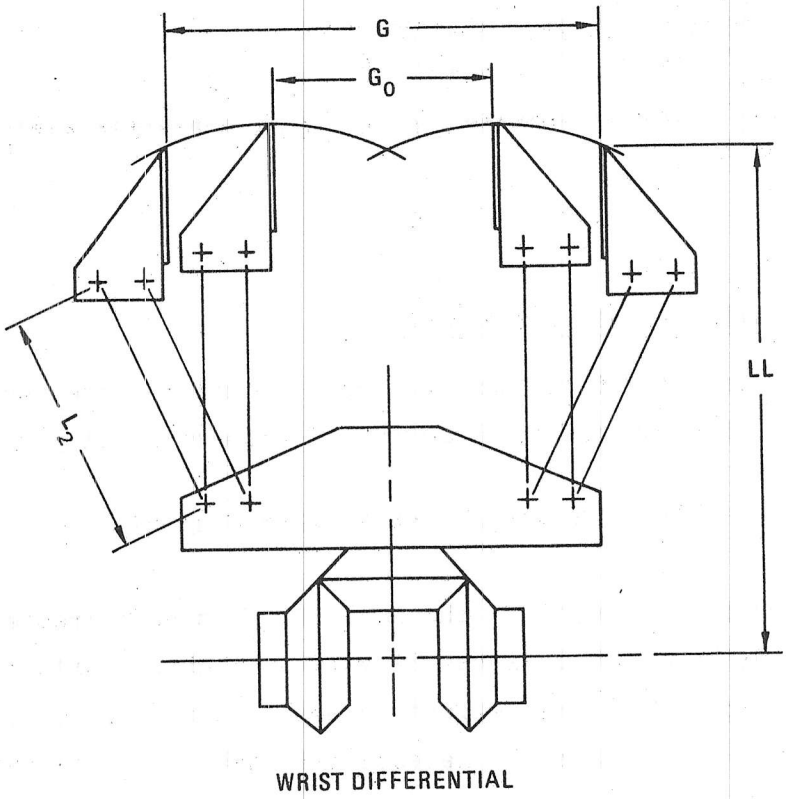



Figure 42 Hand Length Variation versus Hand Opening

The hand opening, G, may be converted to motor steps and vice-versa by using the proportionality constant, $S_6$, given above.

Varying hand length may be taken into consideration in both the forward and backward solutions. Before starting either solution, the correct value of LL would be computed from the hand opening using Equation (32).

# IX APPLICATIONS PROGRAMS

This section discusses several applications programs which use the ARMBASIC commands to move the MiniMover-5 manipulator. These programs may be entered and run just as they appear here. After some study, these programs can be modified to perform slightly differently. Once these programs are thoroughly understood, the user will be better able to write other new applications programs.

## A. Initialization Requirements

Before a program is run, the arm should be moved to a known initial position. This is done for two reasons. The first is to have the wrist motions centered so that Roll and Pitch will have available their full range of motion without the wrist cables binding. The second is to match the position of the joints with the starting position assumed by the program. The differential cable drives to the wrist should always be centered when starting up. This is necessary because it is difficult to tell the state of the differential drive gears by looking at the hand. Centering the differential can be accomplished by looking under the forearm, while manually turning the two large plastic drive gears at the back of the base*. The large drive gears should be moved until the tensioners are centered, as shown in Figure 43.

Matching the position of the joints to the position assumed by the program (initialization) is usually necessary. It is always required in Cartesian coordinate programs and other programs requiring the arm to "know" about objects within its environment. Initialization is not required in either the pick-and-place teach program or the trainer program discussed in this section.

---

* Power must be turned off or the joints freed, under power, by typing @RESET.

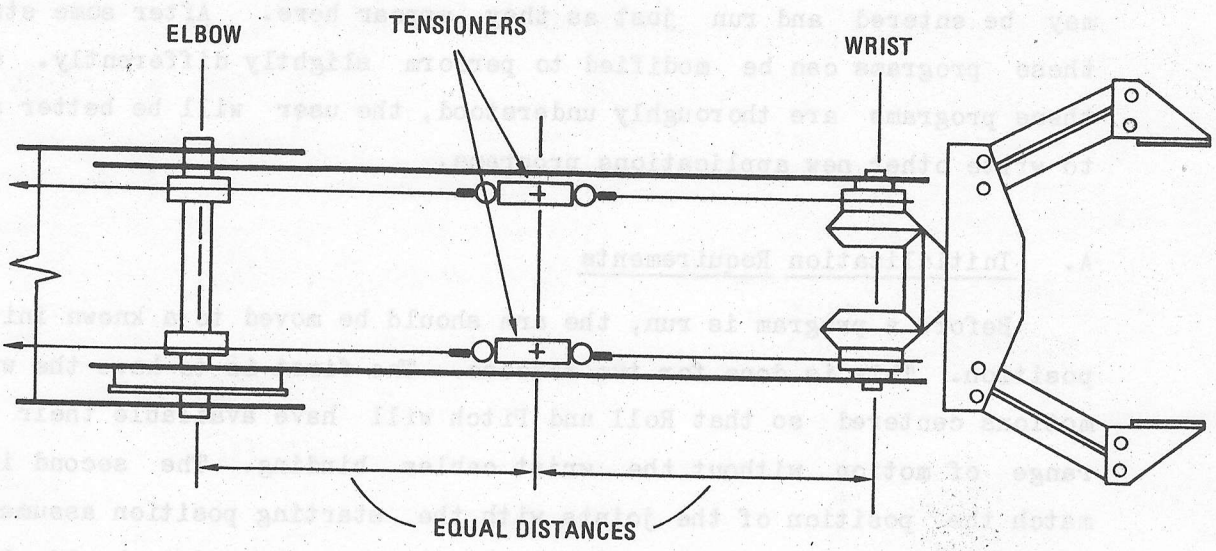



Figure 43 Centering the Wrist Differential Drives

When position initialization is required, the arm is moved, either manually or by using the @SET command, to a fixed position and orientation. An initialization aid could be a mark or an object on the table top which the hand is brought to touch. Vertically positioning the hand end point at a mark on the work table, which is at a known distance in front of the base pivot, can be used to define X, Y, Z, Roll, and Pitch.

B. Pick-and-Place Programs

To illustrate the use of ARMBASIC in manipulation, several methods of programming a pick-and-place task are discussed. The pick-and-place task consists of repeatedly picking objects up at one place and setting them down in another. This task is common in industry where parts from a feeder are placed in an assembly or on a moving conveyor. We will assume an unending supply of parts at the pick-up point and continual removal of parts at the placement point.

The task is defined in Figure 44. Pickup is from A and Placement is at D. The approach and departure trajectories, AB and CD are generally required for practical reasons: an object on flat surface is best lifted before it is moved or a part placed on a shaft must be moved along the axis of the shaft until it is free to be moved.

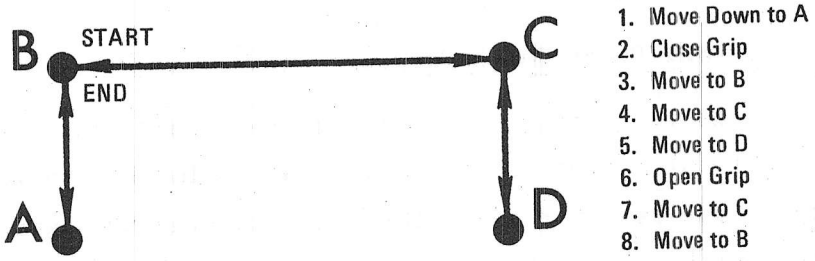



Figure 44 Description of the Pick-and-Place Task

A simple sequence of ARMBASIC commands for performing the pick-and-place task is shown in Table 11. Movements are performed by the @STEP command, including opening the hand [Statement 60]. The elements of the vectors

$$P = (P1, P2, P3, P4, P5)$$

$$Q = (Q1, Q2, Q3, Q4, Q5)$$

$$R = (R1, R2, R3, R4, R5)$$

are the number of steps required for each joint in order to move the arm from A to B, B to C, and C to D respectively. S is the number specifying the speed and GR is the number of steps the hand is to be opened to release the part.

Table 11

Simple ARMBASIC Implementation of Pick-and-Place Program

```
10  @STEP  S,-P1,-P2,-P3,-P4,-P5
20  @CLOSE
30  @STEP  S, P1, P2, P3, P4, P5
40  @STEP  S, Q1, Q2, Q3, Q4, Q5
50  @STEP  S, R1, R2, R3, R4, R5
60  @STEP  S, 0,  0,  0,  0,  0,  GR
70  @STEP  S,-R1,-R2,-R3,-R4,-R5
80  @STEP  S,-Q1,-Q2,-Q3,-Q4,-Q5
90  GOTO  10
```

1. Pick-and-Place Teach Program

The above simple approach is not very practical since it requires that the number of steps on each joint be known in advance. One practical approach for small tasks is entering the data manually, by moving the arm to each of the four positions (A, B, C and D of Figure 44) and letting the computer count and remember the number of steps each joint moves. This is frequently called "teach control."

The teach program shown in Table 12 illustrates the principal. Operating interactively with the computer, the operator manually positions the arm to each of the four positions to specify the pick-and-place task. Each position is obtained by moving the arm, joint-by-

joint, using the keys in the upper left hand corner of the keyboard, as previously described under the @SET command (Section VII-C). The desired hand opening is also conveyed by teach control. Only the speed is entered numerically.

Table 12

Pick-and-Place Teach Program

```
10 REM          ************************************************
20 REM          *                                              *
30 REM          *                  MINIMOVER-5                 *
40 REM          *                                              *
50 REM          *              PICK - AND - PLACE              *
60 REM          *                                              *
70 REM          *             MANUAL TEACH PROGRAM             *
80 REM          *                                              *
90 REM          ************************************************
100 @RESET:  REM         ZERO COUNTERS
110 PRINT "PICK - AND - PLACE ROUTINE"
120 PRINT "     USE MANUAL KEYS TO POSITION ARM"
130 INPUT "SPEED ="; S
140 PRINT "POSITION HAND ON PART, TYPE O WHEN DONE"
150 PRINT "     ADJUST HAND OPENING TO CLEAR PART"
160 @SET
170 @READ A1,A2,A3,A4,A5,GO
180 @CLOSE:  REM     CLOSE HAND AND MEASURE PART
190 @READ A1,A2,A3,A4,A5,GO
200 G = GO - GC  :REM    GRIP SIZE OPEN LESS GRIP SIZE CLOSED
210 PRINT "POSITION PART ABOVE PICKUP SITE, TYPE O WHEN DONE"
220 @SET
230 @READ B1,B2,B3,B4,B5
240 PRINT "POSITION PART ABOVE PLACEMENT SITE, TYPE O WHEN DONE"
250 @SET
260 @READ C1,C2,C3,C4,C5
270 PRINT "POSITION PART AT PLACEMENT SITE, TYPE O WHEN DONE"
280 @SET
290 @READ D1,D2,D3,D4,D5
300 REM
310 REM    RELEASE PART AND RETURN TO B
320 REM
330 @STEP S,0,0,0,0,0,G
340 @STEP S, C1-D1, C2-D2, C3-D3, C4-D4, C5-D5
350 @STEP S, B1-C1, B2-C2, B3-C3, B4-C4, B5-C5
360 REM
370 REM    WAIT UNTIL READY
380 REM
390 INPUT "TYPE G TO GO"; R$
400 IF R$ = "G" THEN 410 ELSE 390
410 REM
420 REM    RUN THE PICK - AND - PLACE PROGRAM
430 REM
1000 @STEP S, A1-B1, A2-B2, A3-B3, A4-B4, A5-B5
1010 @CLOSE
```

```
1020 @STEP S, B1-A1, B2-A2, B3-A3, B4-A4, B5-A5
1030 @STEP S, C1-B1, C2-B2, C3-B3, C4-B4, C5-B5
1040 @STEP S, D1-C1, D2-C2, D3-C3, D4-C4, D5-C5
1050 @STEP S,0,0,0,0,0,G
1060 @STEP S, C1-D1, C2-D2, C3-D3, C4-D4, C5-D5
1070 @STEP S, B1-C1, B2-C2, B3-C3, B4-C4, B5-C5
1080 GOTO 1000
1090 END
```

Those interested in the details of the teach program will note that the values of the software position registers are read by the @READ command after the arm is positioned at each of the four locations. The position registers are stored in the vectors A, B, C and D. The elements of these vectors

$$A = (A1, A2, A3, A4, A5)$$

$$B = (B1, B2, B3, B4, B5)$$

$$C = (C1, C2, C3, C4, C5)$$

$$D = (D1, D2, D3, D4, D5)$$

are the values of the position registers for each joint. The P, Q and R vectors of the simple program (Table 12) are obtained by subtraction as follows: P=B-A, Q=C-B, R=D-C. The desired hand opening, GR, is measured by reading the grip position register before and after closing the hand, using the @CLOSE command [Statements 170-200 in Table 12].

## 2. Pick-and-Place Cartesian Program

When information on pickup or placement positions originates in a computer, teach programming is often not practical. For example, to move pieces on a chess or checker board would require manually teaching the 64 board locations and the 64 corresponding lift-off points! In this case a coordinate system defining the workspace (or "world") would be more practical. In many situations Cartesian world coordinates are the most convenient. Placing a sheet of graph paper on the work table facilitates reading of the pick-up and set-down coordinates directly. The mathematics of coordinate conversions have been discussed in the previous section.

The Cartesian coordinate control program shown in Table 13 shows how this is done with the MiniMover-5 in ARMBASIC. For this program, the locations of the four positions of the pick-and-place task (A, B, C, D) are defined by their Cartesian coordinates (see Figure 13). The X, Y, Z coordinates (in inches) and Pitch and Roll angles (in degrees) at the fingertips are:

A = (8, 0, 0.5, -90, 0)
B = (8, 0, 1.5, -90, 0)
C = (6, 5, 1.5, -90, 90)
D = (6, 5, 0.5, -90, 90) .

Note that the object is gripped at a point 0.5 inches above the table top (Z = 0) and is raised and lowered 1.0 inch for pick-up and set-down. Its X location is changed from 8 to 6 inches and its Y location changed from 0 to 5 inches. The angular orientation of the hand is always straight down (Pitch = -90 degrees), but the object is rotated 90 degrees on its vertical axis between pick-up and set-down (Roll changes from 0 to 90 degrees).

Table 13

Pick-and-Place X-Y-Z Control Program

```
10 REM          ************************************************
20 REM          *                                              *
30 REM          *                MINIMOVER-5                   *
40 REM          *                                              *
50 REM          *         CARTESIAN COORDINATE CONTROL         *
60 REM          *                                              *
70 REM          *                  PROGRAM                     *
80 REM          *                                              *
90 REM          ************************************************
100 REM        DEFINE ARM CONSTANTS
101  H=8.1        :REM    SHOULDER HEIGHT ABOVE TABLE
102  L=7.0        :REM    SHOULDER TO ELBOW AND ELBOW TO WRIST LENGTH
103  LL=3.8       :REM   WRIST TO FINGERTIP LENGTH
104 REM
110 REM        DEFINE OTHER CONSTANTS
111  PI=3.14159
112  C=180/PI     :REM    DEGREES IN 1.00 RADIAN
113  R1=1         :REM    FLAG FOR WORLD COORDINATES
114 REM
120 REM        DEFINE ARM SCALE FACTORS
121  S1=937.8:   S2=S1    :REM  STEPS/RADIAN, JOINTS 1 & 2
122  S3=551.4             :REM  STEPS/RADIAN, JOINT 3
123  S4=203.7:   S5=S4    :REM  STEPS/RADIAN, JOINTS 4 & 5
124  S6=309               :REM  STEPS/INCH, HAND
125 REM
130 REM         INITIALIZATION
131 DIM UU(7,50)          :REM ROOM FOR 50 MOVES
132 U=0
133 @RESET
134 REM
140 REM              READ IN FIRST DATA LINE FOR INITIALIZATION
141 READ X,Y,Z,P,R,GR,S
150 PRINT "SET ARM TO THE FOLLOWING POSITION & ORIENTATION"
151 PRINT "   USING KEYBOARD, TYPE 0 WHEN FINISHED"
152 PRINT "       X = "; X; "INCHES"
153 PRINT "       Y = "; Y; "INCHES"
154 PRINT "       Z = "; Z; "INCHES"
155 PRINT "   PITCH = "; P; "DEGREES"
156 PRINT "    ROLL = "; R; "DEGREES"
157 PRINT "    HAND = "; GR/S6;  "INCHES"
170 @SET
200 GOSUB 5000             :REM GET JOINT ANGLES FOR INITIALIZATION
210 REM  W  IS WHERE WE ARE AT
220 W1=INT(S1*T1)
230 W2=INT(S2*T2)
```

```
240 W3=INT(S3*T3)
250 W4=INT(S4*T4)
260 W5=INT(S5*T5)
270 REM
300 REM READ IN NEXT POINT
308 READ X,Y,Z,P,R,GR,S
309 IF X < -100 GOTO 1000
310 PRINT "MOVE TO X,Y,Z,P,R,GR,S"
311 PRINT X,Y,Z,P,R,GR,S
320 GOSUB 5000              :REM GET JOINT ANGLES FOR NEXT POINT
325 REM         C IS THE CHANGE (IN COUNTS)
330 C1=INT(S1*T1)-W1
340 C2=INT(S2*T2)-W2
350 C3=INT(S3*T3)-W3
360 C4=INT(S4*T4)-W4
370 C5=INT(S5*T5)-W5
400 REM     UPDATE WHERE WE WILL BE
410 W1=W1+C1:W2=W2+C2:W3=W3+C3:W4=W4+C4:W5=W5+C5
420 @STEP S,C1,-C2,C3,-C4,C5
425 U=U+1
430 UU(1,U)=C1
431 UU(2,U)=-C2
432 UU(3,U)=C3
433 UU(4,U)=-C4
434 UU(5,U)=C5
450 UU(6,U)=GR
455 UU(7,U)=S
460 IF GR <0 GOTO 500
461 REM   OPEN HAND IF GR>0
470 @STEP S,0,0,0,0,0,GR
480 GOTO 300
490 REM   CLOSE HAND AND SQUEEZE IF GR < 0
500 @CLOSE
520 @STEP 100,0,0,0,0,0,GR
530 GOTO 300
540 REM
1000 PRINT " RUN THE PROGRAM"
1010 FOR I=2 TO U
1020 @STEP UU(7,I),UU(1,I),UU(2,I),UU(3,I),UU(4,I),UU(5,I)
1030 IF UU(6,I)<0 GOTO 1060
1040 @STEP UU(7,I),0,0,0,0,0,UU(6,I)
1050 GOTO 1100
1060 @CLOSE
1080 @STEP 100,0,0,0,0,0,UU(6,I)
1100 NEXT I
1110 GOTO 1010
2000 END
2010 REM
5000 REM      SUBROUTINE TO CALCULATE JOINT COORDINATES
5010 REM                ENTER WITH X,Y,Z,PITCH,ROLL, AND R1
5015 REM                RETURN WITH JOINT ANGLES T1 TO T5 (IN RADIANS)
5020 P=P/C:R=R/C
```

```
5030 RR=SQRT(X*X+Y*Y)
5040 IF RR<2.25 THEN 6000
5050 IF X=0 THEN T1=SGN(Y)*PI/2 ELSE T1=ATN(Y/X)
5060 IF X<0 GOTO 6000
5070 T4=P+R+R1*T1
5080 T5=P-R-R1*T1
5090 IF T4<-270/C OR T4>270/C GOTO 6000
5100 IF T5<-270/C OR T5>270/C GOTO 6000
5110 RO=RR-LL*COS(P)
5120 ZO=Z-LL*SIN(P)-H
5130 IF RO<2.25 GOTO 6000
5140 IF RO=0 THEN G=SGN(ZO)*PI/2 ELSE G=ATN(ZO/RO)
5150 A=RO*RO + ZO*ZO
5160 A=4*L*L/A-1
5170 IF A<0 GOTO 6000
5180 A=ATN(SQRT(A))
5190 T2=A+G
5200 T3=G-A
5210 IF T2>144/C OR T2<-127/C GOTO 6000
5220 IF (T2-T3)<0 OR (T2-T3)>149/C GOTO 6000
5230 RETURN
6000 PRINT "********** OUT OF REACH **********"
6010 END
6020 REM
9000 REM   COORDINATE DATA FOR PICK-AND-PLACE TASK
9010 REM        STATEMENT 10000 IS INITIALIZATION POINT
9020 REM        STATEMENT 10010-10070 DEFINE TRAJECTORY
9030 REM        STATEMENT 10080 SIGNALS NO MORE DATA
10000 DATA      5,   0,    0,  -90,   0,    0,   50
10010 DATA      8,   0,  1.5,  -90,   0,    0,   50
10020 DATA      8,   0,  0.5,  -90,   0,  -30,   50
10030 DATA      8,   0,  1.5,  -90,   0,    0,  100
10040 DATA      6,   5,  1.5,  -90,  90,    0,   50
10050 DATA      6,   5,  0.5,  -90,  90,  300,   50
10060 DATA      6,   5,  1.5,  -90,  90,    0,   50
10070 DATA      8,   0,  1.5,  -90,   0,    0,   50
10080 DATA   -999,   0,    0,    0,   0,    0,    0
```

Those interested in programming details may wish to study the Cartesian coordinate program further. Data for the program is located after Statement 10000. The first data statement gives the initialization point, so that the computer will know where the arm is. Data statements also include a sixth parameter governing the hand opening (if positive), or the squeezing force (if negative) after each move. A seventh parameter is included which governs the speed setting during each move. The actual arm solution beginning at Statement 5000

includes several tests to determine if the position and orientation requested is within the reach of the arm. During the first phase of the program, [Statements 200 to 530], the arm is moved from data point to data point, as the solutions are performed. Solutions take less than one second. The joint angles obtained are stored in array UU. After all the solutions are obtained, control is transferred to Statement 1000 where the pick-and-place cycle is run repeatedly without coordinate conversions from the precalculated information in UU.

Those who analyze the Cartesian coordinate program carefully will find that it is general purpose and can generate arbitrary motion trajectories with up to 50 movements. This is sufficient for many assembly tasks as building structures from blocks. The power of the ARMBASIC approach can be noted by comparing the simplicity of this program, which is less than 100 lines of code, with previous approaches which have required 8K words, or more, of assembly language coding to do a similar type of task.

## C. MiniMover-5 Trainer Program

This program, written in ARMBASIC, is shown in Table 14. The trainer allows the user to manually define up to 100 positions. The user can then command the computer to move the arm sequentially, in either direction, to a specified position. For example, starting with position 0 and ten blocks on the table, the user can control the arm to stack the ten blocks. Then, telling the program to move back to position 0 will cause the blocks to be unstacked and returned to their initial position; telling the computer to go to the final position will stack them once again.

Table 14

MiniMover-5 Trainer Program

```
1  REM          ************************************************
2  REM          *                                              *
3  REM          *                MINIMOVER-5                   *
4  REM          *                                              *
5  REM          *              TRAINER PROGRAM                 *
8  REM          *                                              *
9  REM          ************************************************
10 DIM A(6,100)
20 I9=I1=0
30 FOR N=1 TO 6
40 A(N,0)=0
50 NEXT N
60 '
70 CLS
80 PRINT "STARTING POSITION  ",0,
90 PRINT "CURRENT POSITION  ",I1,
100 PRINT "ENDING POSITION  ",I9,
110 '
120 PRINT "C=CLEAR ALL POSITIONS"
130 PRINT "S=SET NEXT POSITION"
140 PRINT "M=MOVE TO OTHER POSITION"
150 INPUT B$
160 IF B$="C" THEN 280
170 IF B$="S" THEN 330
180 IF B$="M" THEN 410
190 GOTO 120
200 '
210 PRINT "USE THE FOLLOWING KEYS TO MOVE THE ARM"
220 PRINT "          1   2   3   4   5   6   "
230 PRINT "            Q   W   E   R   T   Y "
240 PRINT "PRESS 0 (ZERO) WHEN FINISHED"
250 @SET 200
260 RETURN
270 '
280 PRINT "HOME THE ARM"
290 GOSUB 210
300 @RESET
310 GOTO 20
320 '
330 I1 = I1 + 1
340 IF I1 > 100 THEN 70
350 PRINT "SET THE ARM FOR POSITION  ",I1
360 GOSUB 210
370 @READ A(1,I1),A(2,I1),A(3,I1),A(4,I1),A(5,I1),A(6,I1)
380 IF I9 < I1 THEN I9 = I1
```

```
390 GOTO 70
400 '
410 PRINT "MOVE TO POSITION  "
420 INPUT K
430 IF K < 0 OR K > I9 THEN 70
440 IF K = I1 THEN 70
450 IF K < I1 THEN 530
460 '
470 FOR J = I1 TO K-1
480 @STEP 200, A(1,J+1)-A(1,J), A(2,J+1)-A(2,J),
               A(3,J+1)-A(3,J), A(4,J+1)-A(4,J),
               A(5,J+1)-A(5,J), A(6,J+1)-A(6,J)
490 NEXT J
500 I1 = K
510 GOTO 70
520 '
530 FOR J = I1 TO K+1 STEP -1
540 @STEP 200, A(1,J-1)-A(1,J), A(2,J-1)-A(2,J),
               A(3,J-1)-A(3,J), A(4,J-1)-A(4,J),
               A(5,J-1)-A(5,J), A(6,J-1)-A(6,J)
550 NEXT J
560 I1 = K
570 GOTO 70
```

D. Thickness Measuring Program

The program shown in Table 15 uses the hand, as a measuring device, to determine the thickness of objects placed between its fingers. When the program is first run, the @STEP command is used to close the hand with no object within its grasp. The internal position registers are then reset using the @RESET command. This is the calibration phase of the program [Statements 150 and 160].

Next, whenever the ENTER key is pressed, the hand closes upon the object to be measured. The internal position register corresponding to the grip size is then read using the @READ command. The grip value is then scaled into inches and printed out. Finally the hand re-opens to measure another object. The hand scale factor, $S_6$, which is discussed in Section VIII-E, is used to convert from the number of steps of the hand to the object size in inches.

In this program, a distinction is made between "thick" and "thin" objects to demonstrate a decision making capability. If the measured

thickness, T1 [Statement 230], is less than 0.015 inches, a separate branch is taken. This technique may be used to determine if an object was indeed gripped, or if the fingers merely closed upon themselves.

Table 15

Thickness Measuring Program

```
100  PRINT "*************************************************************
110  PRINT "*     PROGRAM TO MEASURE THE THICKNESS (IN INCHES) OF      *
120  PRINT "*           OBJECTS PLACED BETWEEN THE FINGERS             *
130  PRINT "*                        OF THE HAND                       *
140  PRINT "*************************************************************
150  @CLOSE 20
160  @RESET                                :REM CALIBRATE THE HAND
170  @STEP 20,0,0,0,0,0,500
180  PRINT "PRESS ENTER TO MEASURE OBJECT"
190  INPUT D                               :REM INPUT DUMMY VARIABLE
200  @CLOSE 20
210  @READ D,D,D,D,D,G
220  T1 = G/309                            :REM CONVERSION TO INCHES
230  IF T1 < 0.015 THEN 270
240  PRINT "THICKNESS IS ....",T1
250  PRINT "-----------------------------------------------------------"
260  GOTO 170
270  PRINT "THICKNESS IS APPROXIMATELY 0, THAT IS ....",T1
280  GOTO 250
```

E. Operating Two Arms from the Same Computer

Two arms can be operated in parallel from the same computer by merely changing the jumper, J1, in one of them. In order to operate them, the desired arm is selected by the @ARM 1 or @ARM 2 command prior to moving that arm. Once the arm has been moved to the desired position, the second arm can be moved by again selecting it with the @ARM command. Once the arm number has been specified, the ARMBASIC commands will operate as previously described, except they will be controlling a different arm.

Using the above approach, each arm can be controlled sequentially, but not simultaneously. For those users who are proficient in Z-80 assembly language, the software motor driver given in Section XI-C can be modified so that both arms can be controlled simultaneously.

## F. Suggested Advanced Applications

The programs described here are intended to give the reader an idea of the type of manipulation work that can be performed with the MiniMover-5. The serious user will certainly want to develop further software on their own. This section gives some ideas on the type of projects that could be undertaken.

### 1. 3-D Measurement

The MiniMover-5 manipulator can be used as a measurement tool (coordinate measurement machine) to determine the X, Y, Z coordinates of objects within its working volume. First, initialize the arm by touching the end point to a point of known X, Y, and Z. Then, touch any other object with the end point using manual control (@SET command). Read the position of the arm (@READ command) and apply the forward solution described in Section VIII-B. This gives the X, Y, Z coordinates of the end point.

Such a 3-Dimensional measurement system called "Pointy" has been developed for the Stanford Arm at Stanford University. Using this technique, a disk file of fixed objects and their shapes can be stored for later use. The X, Y, Z locations may be accessed by name. For example, another program could access the file to find the position of the "saucer" in order to put the "cup" on it. Using Pitch and Roll axes data, measurements of orientation can also be made. By using a special measurement "tool", like a miniature ice pick rather than the center of the fingertips, measurement precision could be increased.

## 2. Sensor Control

Additional sensors can be easily interfaced to the host computer using the auxiliary input port provided on the interface card. Simple on/off sensors such as switches or those having a digital output can be connected directly. Suggested applications include:

(1) Placing micro-switches on the fingers to "feel" for objects.

(2) Placing a commercially available light-emitting diode/photo transistor range sensor on the fingertips to search for, and locate objects on the table top automatically.

(3) Placing a light emitting diode on one finger and a photo transistor on the other to sense the presence of objects in the open hand.

(4) Installing additional micro-switches on the body of the arm for automatic initialization.

## 3. Motions in Hand Coordinates

In addition to the Cartesian coordinate system described in Section VIII, motions in hand coordinates are often useful. A hand coordinate system suggested by Whitney [20] is shown in Figure 45*. Hand coordinates are often useful for moving objects held in the hand such as removing a peg from a hole.

When sensors are mounted on the hand, hand coordinates are essential for motion under their control. For example, to activate a touch sensor installed on the fingertip, we need to command the hand to move along the "reach" axis (see Figure 45). This is true for both a reach-until-touch strategy or for a reflex strategy where the hand retracts whenever the switch "touches" anything.

Motion in hand coordinates can be derived from X, Y, and Z, coordinates and Roll and Pitch angles using trigonometric relations. For example, to move a distance, L, along the reach axis we need only change the X, Y, and Z command coordinates as follows:

---

* In this Figure, Twist is equivalent to Roll, Tilt is equivalent to Pitch and Turn is equivalent to Yaw.

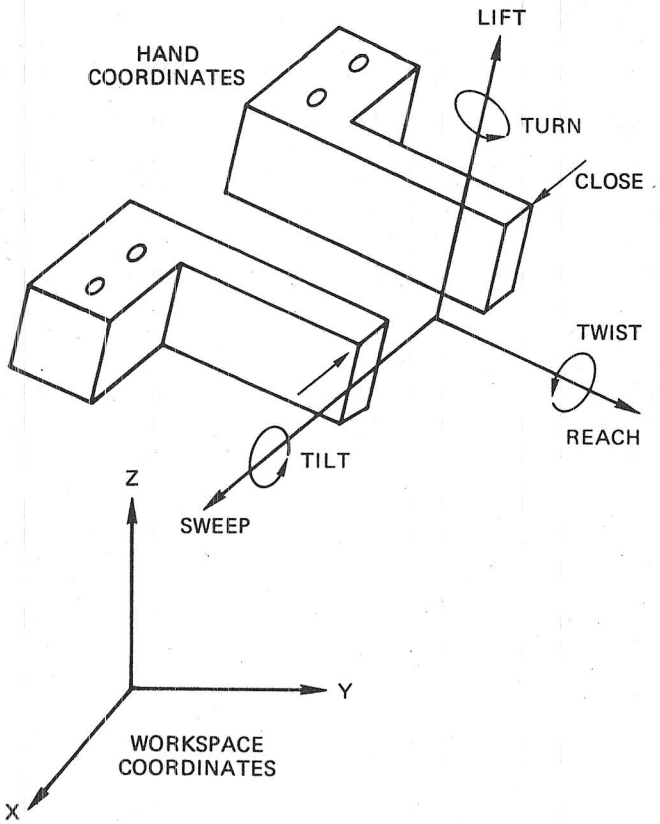



Figure 45 Definition of Hand Coordinate System

$$X' = X + L \cos P \cos \theta_1$$

$$Y' = Y + L \cos P \sin \theta_1$$

$$Z' = Z + L \sin P$$

where P is the Pitch angle and $\theta_1$ is the base angle. The complete solution is left to the reader.

4. Path Control

Motion of the MiniMover-5 under the @STEP command is along curved paths defined by the rotation of the arm segments around their joints. Often, straight line motion is required. This is especially true when drawing a straight line with a hand held pen, extracting a peg from a hole, sliding an object along a table top, etc.

A system for linear path control has been proposed by Paul [13], and could be implemented for the MiniMover-5. Paul's technique

involves linear interpolation between the end points of the desired trajectory to find a number of intermediate "bench mark" points which are on the desired trajectory. Curvilinear motion between the bench mark points will approximate a straight line if the bench mark points are sufficiently close together.

## 5. Matrix Algebra

Solutions for different coordinate systems can be expressed more elegantly in terms of 4 X 4 Denavit-Hartenburg matrices [13]. The BASIC programming language is well suited to the required matrix operations such as addition, multiplication and inversion. Developing the matrix expressions for arm position would allow arm control in arbitrary, even moving coordinate systems. Arbitrary tools held in the hand can also be modelled and controlled using this technique.

## 6. Interactive Control Language

It would be more convenient to "talk" to the arm in an English-like language rather than the simple ARMBASIC commands [4]. Examples are:

* Pick up screwdriver
* Put screwdriver in box
* Pick up box
* Turn box over

Examples of this nature are similar to artificial intelligence work currently being performed at various research laboratories and universities. Some high level commands require knowledge of where objects are (world model), and need to plan motions through intermediate points to accomplish a given task. There is room for many new ideas in this area.

# X OPERATION FROM AN 8-BIT PARALLEL PORT

The MiniMover-5 may be shipped in either the TRS-80 or 8-bit parallel configuration. In either case, the printed circuit card in the base is identical, and may be converted from one to the other by following the procedure outlined in this section. Briefly, this procedure entails removing/inserting two integrated circuits and various jumpers. This discussion is written from the viewpoint of converting the TRS-80 version into the 8-bit parallel version. In order to convert back from the 8-bit parallel version, the reader need only reverse this procedure.

## A. Circuit Card Modifications

Before performing this modification, the MiniMover-5 should be unplugged from the host computer and disconnected from the 12-volt power supply.

The printed circuit card is accessible by removing the four rubber feet from under the base. Once the base has been opened, the user will find the card as shown in Figure 23. Although all of the circuit card components are TTL logic and relatively immune to static charge, care must be exercised to avoid damaging any of the components.

The procedure for modifying the interface card is as follows:

* Remove IC 17-This integrated circuit which is mounted in a socket to facilitate easy removal, controls the port address to which the MiniMover-5 responds. Since, in 8-bit parallel mode, the port address is determined by the particular microcomputer system the manipulator will be attached to, the port addressing must be disabled.
* Install jumpers-Without the integrated circuit (IC 17), the decoder (IC 16) will never be enabled. Thus, jumpers must be installed in place of IC 17 to enable the decoder. The jumpers to be installed are (1) IC 17 pin 14 to pin 8, and (2) IC 17 pin 7 to pin 3 and pin 6.

* Remove IC 19-This integrated circuit is a tri-state input buffer. Since, in 8-bit parallel operation, the inputs to the computer are on a separate input port rather than the bi-directional data bus of the TRS-80, this must be removed. This IC is also installed in a socket to facilitate easy removal.

* Install Jumpers-In order to make the input signals available at the ribbon connector, jumpers must be installed in place of IC 19. The jumpers are: pin 3 to 2, pin 5 to 4, pin 7 to 6, pin 9 to 10, pin 11 to 12, and pin 13 to 14.

* Check jumper J1-Connect E1 and E2 with J1.

The completed modifications are shown in Figure 46. This procedure allows A0-A2 to select the correct output latch, and makes the input signals available on the ribbon connector.

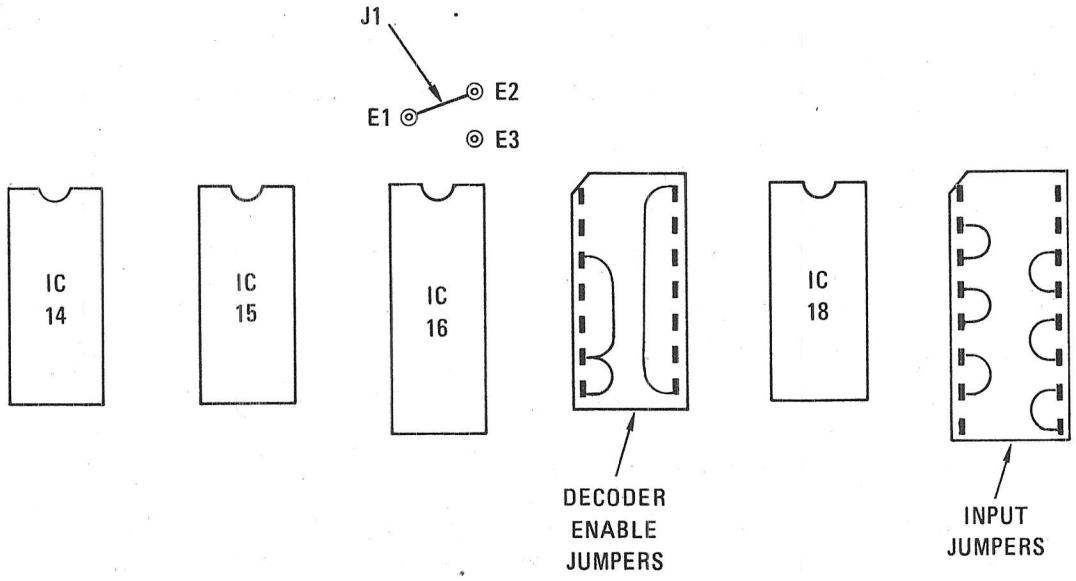



Figure 46 Jumpers Configured for Parallel Port

B. Attachment and Operation from a Parallel I/O Port

Proper connection of the MiniMover-5 to an 8-bit parallel I/O port is shown in Figure 47. Figure 48 shows the logical mapping of the bits of a typical 8-bit parallel output register to those of the MiniMover-5 interface. Figure 49 shows the MiniMover-5 ribbon connector and pin assignments.

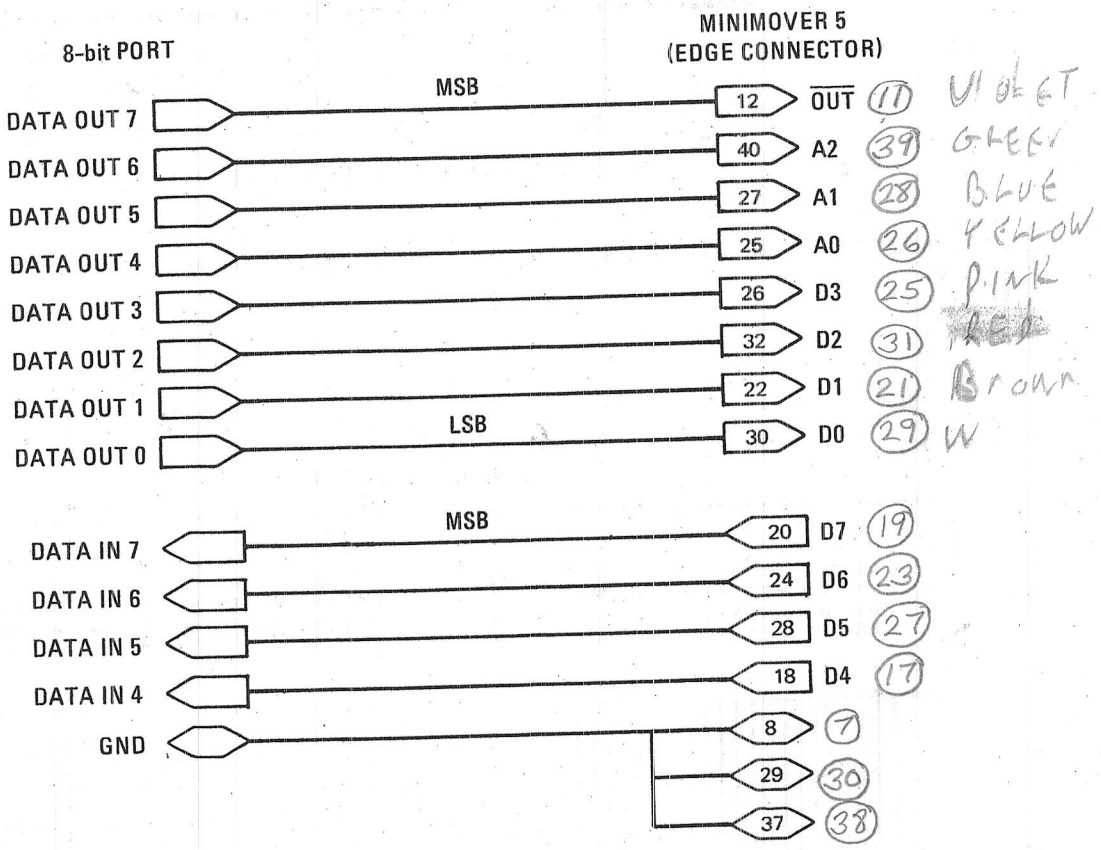



Figure 47 Connections for 8-bit Parallel I/O Port

The procedure for outputing data to a motor or port is as follows (see Figure 48):

* The driver software determines the particular 4-bit pattern which is to be output to a specific motor.
* The driver software places this pattern in bits 0, 1, 2, and 3 of a register which will eventually be output to the parallel port as shown in Figure 48.

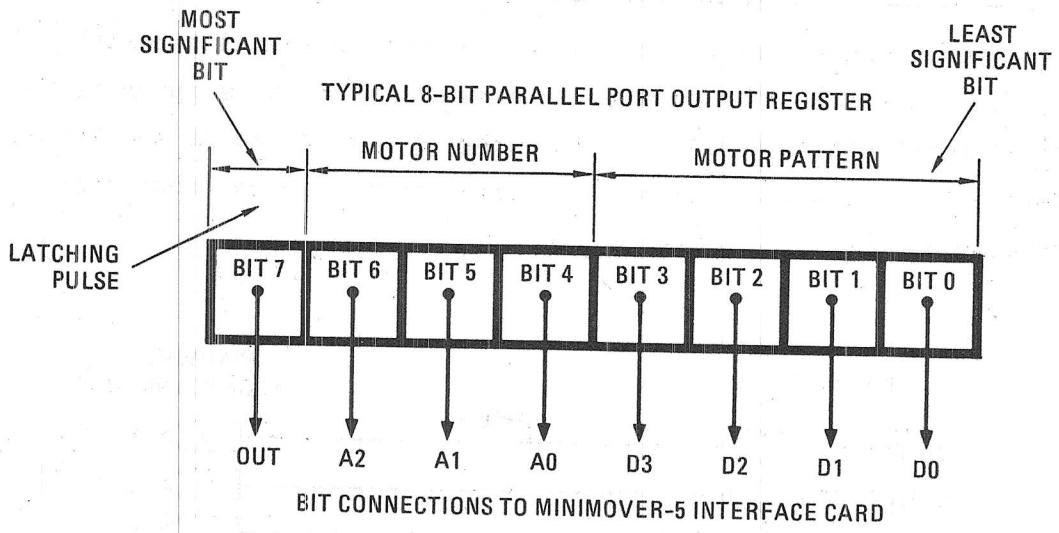
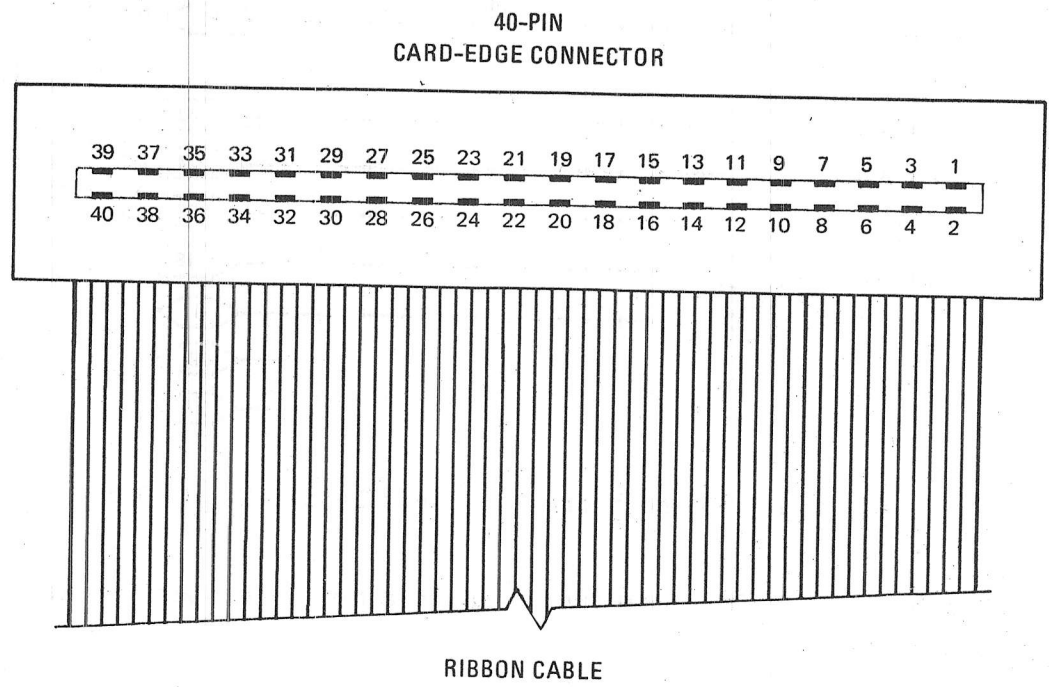



Figure 48 Logical Mapping of a Typical Parallel Output Register




NOTE: Viewed looking into card-edge connector.

Figure 49 Pin Assignments for MiniMover-5

* The driver software clears the most significant bit of the register (latching pulse).

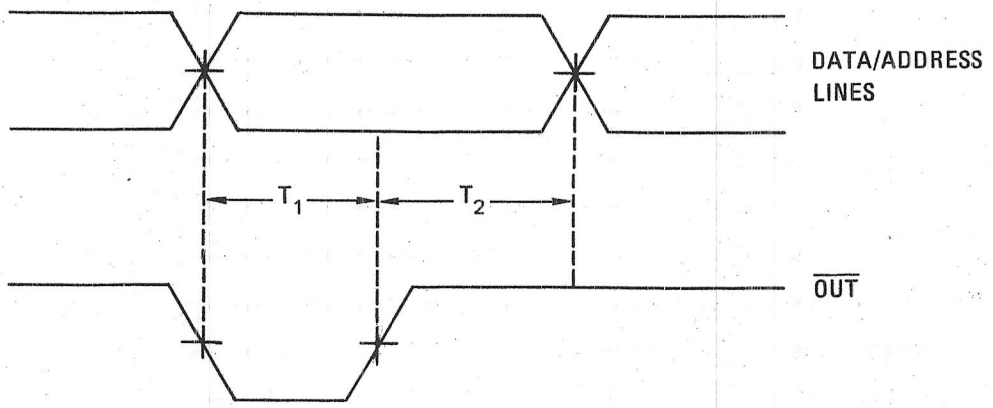
* The driver software places the number of the particular motor* to be driven in bits 4, 5, and 6 of the register.

* The register is output to the parallel output port**. This sets up the data bus and the address select lines.

* The most significant bit of the register should then be set and the register output again. This provides a positive going signal on the OUT line of the MiniMover-5. This will strobe the four bits of motor pattern into the latch addressed by motor number.

The procedure discussed here, and the description of the TRS-80 driver program in Section XI, are intended to aid the user in writing an assembly language motor driver program on their own computer. The driver should be written in assembly language to obtain acceptable speed. If slow speed operation is acceptable, then the driver can be written in a high level language such as BASIC.

The timing of the output signals is shown in Figure 50. Information on the data input lines may be read at any time.




Minimum Times: $T_1 > 90$ ns, $T_2 > 100$ ns

Figure 50 Output Timing Diagram

---

* Motor 1 (001) is the base and motor 6 (110) is the hand.

** Some parallel I/O ports are inverting. If this is the case, then the value of the register should always be complemented just prior to output.

C. Example--Interfacing to the Cromemco Single Card Computer

The Cromemco* single card computer (CSCC) is an S-100 bus CPU board with ROM, RAM, serial I/O and three parallel I/O ports. It can control one or two MiniMover-5/8P units directly from its parallel I/O port(s).

The diagram for connecting a MiniMover-5/8P manipulator to J1 or J2 of the CSCC is shown in Figure 51. This can be accomplished by cutting the connector off of the flat ribbon cable of the MiniMover-5, and soldering the wires to a connector which mates with J1 or J2 of the CSCC. If J1 is used, the I/O port number is Hex OA; If J2 is used, the port number is Hex OB.

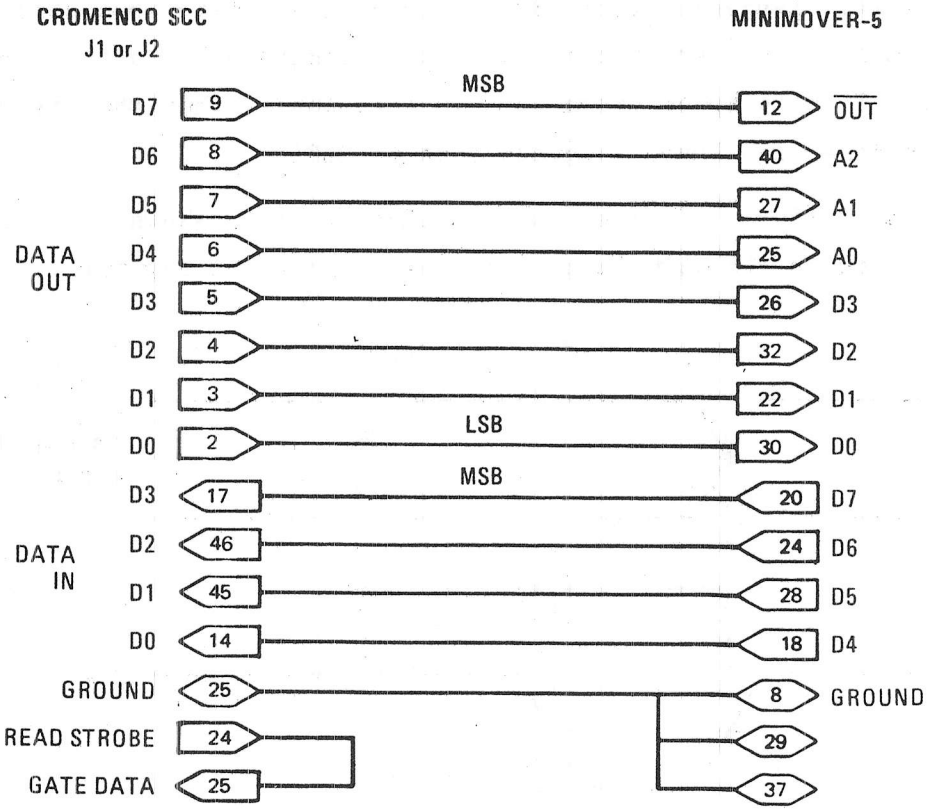



Figure 51 Connection to the Cromemco Single Card Computer

Cromemco has a 4K ROM set (MCB-216) to run control BASIC on the SCC board. The user can have application programs, written in a high level language reside either in the remaining 4K of ROM space or in the 1K of

---

* Information on the SCC can be obtained from Cromemco, Inc., 280 Bernardo Ave., Mountain View, CA. 94040.

on-board RAM. If control BASIC is used, there are two ways to control the MiniMover-5: The OUT command (high level) and the CALL command (low level).

The OUT command of control BASIC can be used to send commands to the MiniMover-5, and the IN command can be used to read the jaw-sense switch. The following sequence will output motor phase pattern one to motor number 2 (the shoulder motor):

```
10 OUT (%0A) = %21
20 OUT (%0A) = %A1
```

The following statement will test the jaw-sense switch and branch to Statement 100 if it is closed:

```
10 IF AND (128,IN(%0A)) GOTO 100
```

The CALL command of control BASIC can be used to access assembly language subroutines to control the MiniMover-5.

If assembly language is used, the following sequence is needed to set/change the stepper motor phase pattern:

```
LD    A,16*(MOTOR NUMBER)+(PHASE PATTERN)
OUT   (0AH),A
OR    A,80H
OUT   (0AH),A
```

For reading the jaw-sense switch, use:

```
IN    A,(0AH)
AND   A,80H
```

# XI MOTOR DRIVER ALGORITHM

One of the more complicated aspects of controlling the MiniMover-5 from a computer is controlling the motion of all six motors simultaneously. This section describes the assembly language subroutine used by ARMBASIC on the TRS-80, that steps the six motors so that all six movements begin and end simultaneously. Background information for this section will be found under stepper motor control (Section V) and the @STEP command (Section VII-A).

## A. Background

The driver accomplishes coordinated motion which is important in any manipulation task. For example, as shown in Figure 52, if the elbow motor is driven first and the shoulder second, a rather awkward and unpredictable path results. However if the steps are timed such that each motor is pulsed regularly during the duration of the move (see Figure 29), then a coordinated motion results. The coordinated motion produces a straighter path from A to B and is also faster.

The multi-axis driver algorithm is general purpose and may be adapted to other manipulators and robotic devices having two or more actuators to be coordinated. Its origin stems from the digital differential analyzer (DDA) used in machine tool control (See Reference [15] and [16]). In fact it is a software simulation of six DDA's.

The user wishing to write their own arm driver for another computer will find it valuable to analyze the driver algorithm developed for the TRS-80 and its detailed implementation. This program could be modified for operation of the MiniMover-5 from a parallel port as discussed in Section X. In this case, each output instruction must be replaced by two as described.

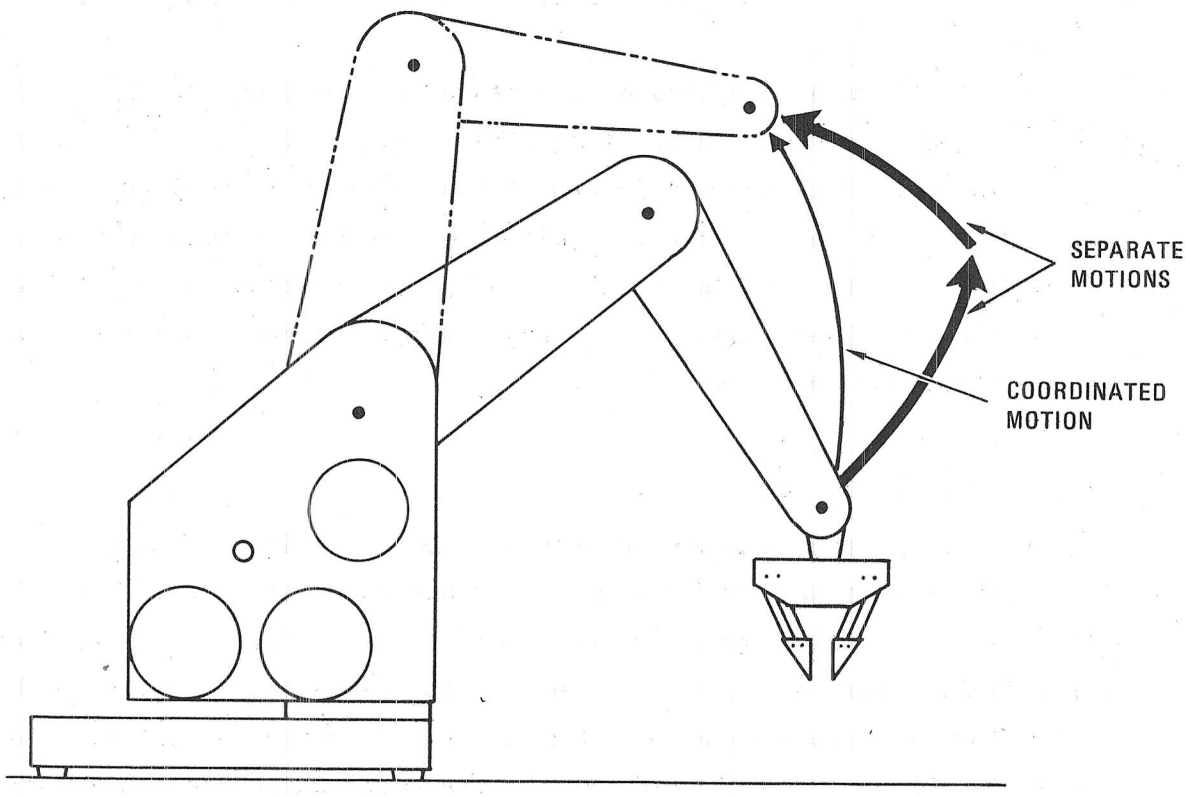



Figure 52 Coordinated versus Uncoordinated Motions

B. Driver Algorithms

The method of generating a coordinated motion is briefly described below. Variables include an integer M and six integer counters.

* Determine the largest number of steps, M, that any of the six motors is to be driven.
* Set all six counters to M/2.
* Repeat the following loop M times.
    - o Subtract the absolute value of the number of steps each motor is to be stepped from its corresponding counter.

- If the contents of any counter is less than 0, step that motor in the requested direction and add M to its counter.
- Enter a Wait routine to produce the required time delay.

Note that the motor with the largest number of steps dominates this procedure. The dominant joint is stepped once, each time through the inner loop, and moves at the speed determined by the wait routine. No motor will go faster.

In order to turn each motor by one step, one of the numbers shown in Table 16 must be output to the desired motor. In order to step the motor in one direction, the numbers in Table 16 must be output to the motors from top to bottom. If the end of the table is reached, it is necessary to wrap around to the other end of the table and continue sequentially. In order to change direction, the direction in which the entries are used must be reversed.

Table 16

Stepper Motor Output Values

| STEP | HEX VALUE |
|---|---|
| 1 | 1 |
| 2 | 5 |
| 3 | 4 |
| 4 | 6 |
| 5 | 2 |
| 6 | A |
| 7 | 8 |
| 8 | 9 |

C. The TRS-80 Driver

The flow chart of the algorithm for the TRS-80 motor driver program is shown in Figure 53 and the Z-80 assembly listing is given in Table 17. This driver accepts as inputs, the six signed values representing the number of steps to be output to each motor, and an integer delay value. The sign of each motor value indicates the direction in which the motor is to be driven and the magnitude indicates the number of steps in that direction. Motors are controlled by the electronic interface described in Section VI.

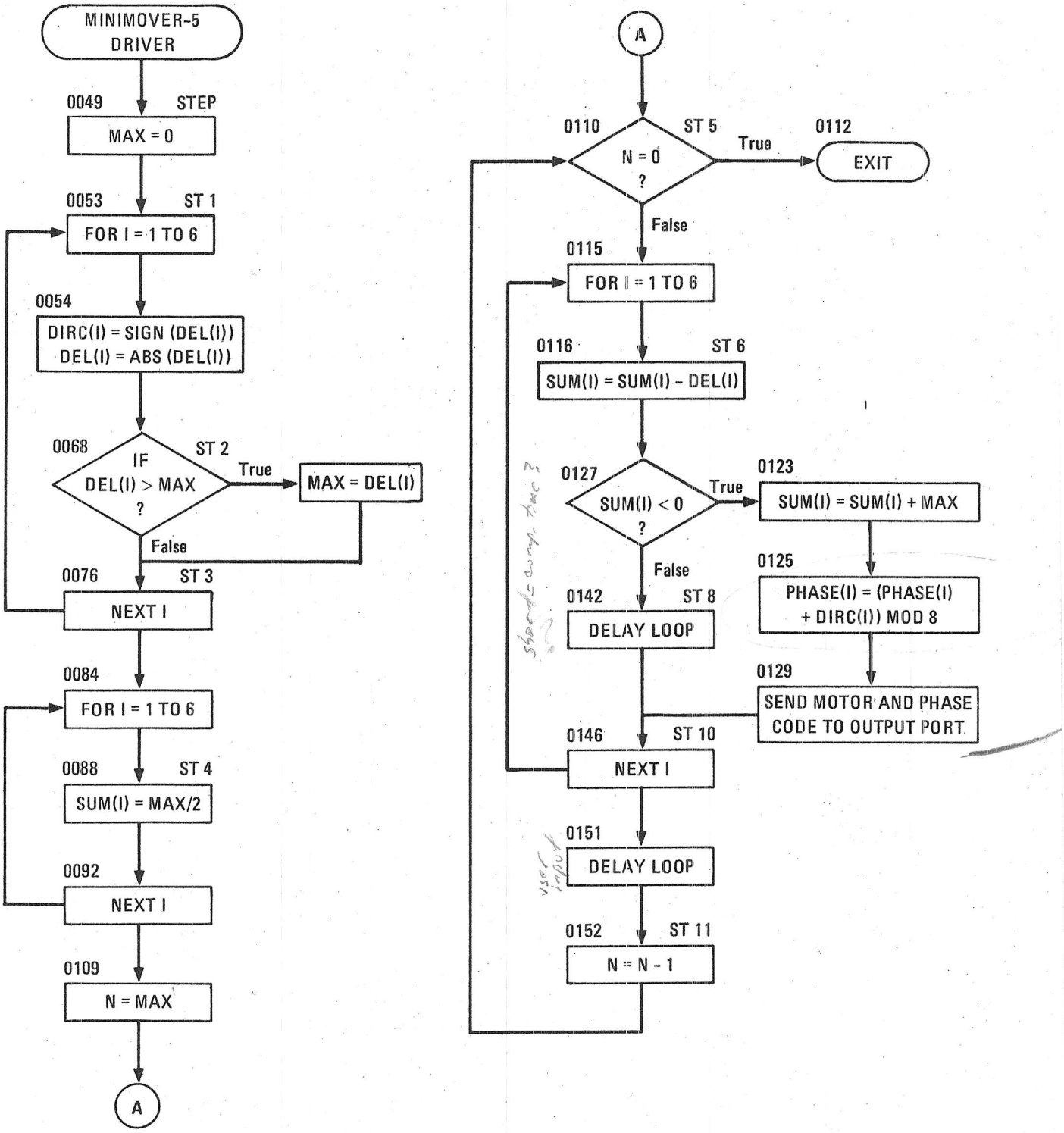



Figure 53 Flow Chart of Z-80 Motor Driver Program

Table 17

Assembly Language Listing of Z-80 Motor Driver Program

```
Z80 ASSEMBLER                                                    PAGE 0001

                        0001 ;-----------------------------------------------;
                        0002 ;                                               ;
                        0003 ;            DRIVER FOR MINIMOVER-5             ;
                        0004 ;                                               ;
                        0005 ;                COPYRIGHT 1980                 ;
                        0006 ;                   MICROBOT                    ;
                        0007 ;                                               ;
                        0008 ;-----------------------------------------------;
                        0009            ENTRY   STEP
(00C8)                  0010 PORT       EQU     0C8H            ;6 OUTPUT PORTS
                        0011 ;
                        0012 ;    INPUT PARAMETERS:
                        0013 ;       CONT = INTEGER
                        0014 ;              DELAY LOOP COUNT BETEEN STEPS
                        0015 ;       DEL = ARRAY[0..5] OF INTEGER
                        0016 ;             # OF STEPS FOR JOINT #0..#5
                        0017 ;             SIGN IS DIRCTION OF STEP
                        0018 ;    OUTPUT PARAMETERS:
                        0019 ;       NONE
                        0020 ;    INTERNAL VARIABLES:
                        0021 ;       SUM = ARRAY[0..5] OF INTEGER
                        0022 ;       DIRC,PHASE = ARRAY[0:5] OF BYTE,BYTE
                        0023 ;       MAX = INTEGER
                        0024 ;
                        0025            DATA
0000"  (0002)           0026 CONT       DEFS    2
       (0002")          0027 ARRAY      EQU     $               ;REFERENCE POINT
       (0000)           0028 DEL        EQU     $-ARRAY
0002"  (000C)           0029            DEFS    2*6
       (000C)           0030 SUM        EQU     $-ARRAY
000E"  (000C)           0031            DEFS    2*6
       (0018)           0032 DIRC       EQU     $-ARRAY
       (0019)           0033 PHASE      EQU     $-ARRAY+1
001A"  (000C)           0034            DEFS    2*6
0026"  (0002)           0035 MAX        DEFS    2*1
                        0036 ;
                        0037 ;    CODE IS PURE BUT NOT RECURSIVE NOR REENTRANT
                        0038 ;    AF',BC',DE',HL', AND IY REGISTER NOT USED
                        0039 ;    OTHER REGISTERS AND INPUT PARAMETERS ALTERED
                        0040 ;    STACK IS THREE DEEP (INCLUDING RETURN ADDRESS
                        0041 ;    OF THE CALL TO THIS SUBROUTINE)
                        0042 ;
                        0043 ; 1. PREPARE PARAMETERS
                        0044 ;    DIRC[I] ::= SIGN(DEL[I])
                        0045 ;    DEL[I] ::= ABS(DEL[I])
                        0046 ;    MAX ::= MAX(DEL[0]..DEL[5])
                        0047 ;
                        0048            REL
0000'  0606             0049 STEP       LD      B,6
0002'  210000           0050            LD      HL,0            ;MAX ::= 0
0005'  222600"          0051            LD      (MAX),HL
0008'  DD210200"        0052            LD      IX,ARRAY
000C'  C5               0053 ST1        PUSH    BC              ;FOR I ::= 0 TO 5
000D'  DD5601           0054            LD      D,(IX+DEL+1)    ;  DE ::= DEL[I]
0010'  DD5E00           0055            LD      E,(IX+DEL)
0013'  7A               0056            LD      A,D
0014'  07               0057            RLCA                    ;  CF ::= SIGN
```

Table 17 (Continued)

```
CROMEMCO CDOS Z80 ASSEMBLER version 02.15                    PAGE 0002

0015' 3E01          0058           LD      A,1             ;  A ::= +1
0017' 3009          0059           JR      NC,ST2          ;  IF CF = 0 THEN
0019' 7A            0060           LD      A,D             ;    DE ::= -DE
001A' 2F            0061           CPL
001B' 57            0062           LD      D,A
001C' 7B            0063           LD      A,E
001D' 2F            0064           CPL
001E' 5F            0065           LD      E,A
001F' 13            0066           INC     DE              ;    A ::= -1
0020' 3EFF          0067           LD      A,-1            ;  END IF
0022' DD7718        0068 ST2       LD      (IX+DIRC),A     ;  DIRC[I] ::= A
0025' DD7201        0069           LD      (IX+DEL+1),D    ;  DEL[I] ::= DE
0028' DD7300        0070           LD      (IX+DEL),E
002B' 2A2600"       0071           LD      HL,(MAX)        ;  IF DE > MAX THEN
002E' AF            0072           XOR     A
002F' ED52          0073           SBC     HL,DE
0031' 3004          0074           JR      NC,ST3
0033' ED532600"     0075           LD      (MAX),DE        ;    MAX ::= DE
0037' DD23          0076 ST3       INC     IX              ;  END IF
0039' DD23          0077           INC     IX
003B' C1            0078           POP     BC
003C' 10CE          0079           DJNZ    ST1             ;END FOR
                    0080 ;
                    0081 ; 2. GET READY FOR THE MAIN LOOP
                    0082 ;    SUM[I] ::= MAX / 2
                    0083 ;
003E' 0606          0084           LD      B,6
0040' 2A2600"       0085           LD      HL,(MAX)        ;HL ::= MAX / 2
0043' CB2C          0086           SRA     H
0045' CB1D          0087           RR      L
0047' DD2B          0088 ST4       DEC     IX              ;FOR I ::= 5 TO 0
0049' DD2B          0089           DEC     IX
004B' DD740D        0090           LD      (IX+SUM+1),H    ;  SUM[I] ::= HL
004E' DD750C        0091           LD      (IX+SUM),L
0051' 10F4          0092           DJNZ    ST4             ;END FOR
                    0093 ;
                    0094 ; 3. THE MAIN LOOP TO STEP ALL MOTORS
                    0095 ;    FOR N ::= 1 TO MAX
                    0096 ;      FOR I ::= 0 TO 5
                    0097 ;        SUM[I] ::= SUM[I] - DEL[I]
                    0098 ;        IF SUM[I] < 0 THEN
                    0099 ;           SUM[I] ::= SUM[I] + MAX
                    0100 ;           PHASE[I]
                    0101 ;             ::= (PHASE[I] + DIRC[I]) MOD 8
                    0102 ;           SEND MOTOR & PHASE CODE TO OUTPUT
                    0103 ;        ELSE IDEL FOR SAME AMOUNT OF TIME
                    0104 ;        END IF
                    0105 ;      END FOR
                    0106 ;      DELAY LOOP; COUNT DOWN "CONT"
                    0107 ;    END FOR
                    0108 ;
0053' ED4B2600"     0109           LD      BC,(MAX)
0057' 78            0110 ST5       LD      A,B             ;FOR N ::= 1 TO MAX
0058' B1            0111           OR      C
0059' C8            0112           RET     Z               ;** THE ONLY EXIT **
005A' C5            0113           PUSH    BC
```

Table 17 (Continued)

```
CROMEMCO CDOS Z80 ASSEMBLER version 02.15                    PAGE 0003

005B' 0606          0114          LD      B,6
005D' DD210200"     0115          LD      IX,ARRAY
0061' DD660D        0116 ST6      LD      H,(IX+SUM+1)    ; FOR I ::= 0 TO 5
0064' DD6E0C        0117          LD      L,(IX+SUM)      ;  HL ::= SUM[I]
0067' DD5601        0118          LD      D,(IX+DEL+1)    ;          - DEL[I]
006A' DD5E00        0119          LD      E,(IX+DEL)
006D' AF            0120          XOR     A
006E' ED52          0121          SBC     HL,DE
0070' 3023          0122          JR      NC,ST8          ;  IF HL < 0 THEN
0072' ED5B2600"     0123          LD      DE,(MAX)        ;   HL ::= HL + MAX
0076' 19            0124          ADD     HL,DE
0077' DD7E19        0125          LD      A,(IX+PHASE)    ;   PHASE[I]
007A' DD8618        0126          ADD     A,(IX+DIRC)     ;    ::= (PHASE[I]
007D' E607          0127          AND     7               ;         + DIRC[I])
007F' DD7719        0128          LD      (IX+PHASE),A    ;         MOD 8
0082' E5            0129          PUSH    HL
0083' 21B400'       0130          LD      HL,TABP
0086' 85            0131          ADD     A,L
0087' 6F            0132          LD      L,A
0088' 3001          0133          JR      NC,ST7
008A' 24            0134          INC     H
008B' 56            0135 ST7      LD      D,(HL)          ;   TABP<PHASE[I]>
008C' E1            0136          POP     HL
008D' 3ECE          0137          LD      A,PORT+6
008F' 90            0138          SUB     B
0090' 4F            0139          LD      C,A             ;   PORT<MOTOR#I>
0091' ED51          0140          OUT     (C),D           ;   SEND IT OUT
0093' 1806          0141          JR      ST10            ;  ELSE
0095' C5            0142 ST8      PUSH    BC              ;   IDEL FOR
0096' 0601          0143          LD      B,1             ;    EQUAL TIME
0098' 10FE          0144 ST9      DJNZ    ST9
009A' C1            0145          POP     BC              ;  END IF
009B' DD740D        0146 ST10     LD      (IX+SUM+1),H    ;  SUM[I] ::= HL
009E' DD750C        0147          LD      (IX+SUM),L
00A1' DD23          0148          INC     IX
00A3' DD23          0149          INC     IX
00A5' 10BA          0150          DJNZ    ST6             ; END FOR
00A7' ED4B0000"     0151          LD      BC,(CONT)       ; DELAY LOOP
00AB' 0B            0152 ST11     DEC     BC
00AC' 78            0153          LD      A,B
00AD' B1            0154          OR      C
00AE' 20FB          0155          JR      NZ,ST11
00B0' C1            0156          POP     BC
00B1' 0B            0157          DEC     BC
00B2' 18A3          0158          JR      ST5             ;END FOR
                    0159 ;
                    0160 ;    TABLE OF PHASE CODE
                    0161 ;
00B4' 01050406      0162 TABP     DEFB    1,5,4,6,2,0AH,8,9
      020A0809
00BC' (0000)        0163          END

Errors              0

Program Length  00BC (188)
Data length     0028 (40)
```

# Appendix A

# MINIMOVER-5 ILLUSTRATED PARTS CATALOG

## Appendix A

## MINIMOVER-5 ILLUSTRATED PARTS CATALOG

This appendix provides the reader with detailed drawings of the MiniMover-5 mechanical arm. By using these drawings together with the parts list in Table A-1, the user should be able to perform any required maintenance or repair to the MiniMover-5. The part numbers listed in Table A-1 correspond to the part numbers on the drawings except that 5000 has been added. Several drawings are provided. They are:

* Figure A-1--This figure shows an exploded view of the major structural components of the arm.
* Figure A-2--This figure provides details of the entire, assembled manipulator and shows the details of the gears and cabling.
* Figure A-3--This figure provides a different view of the cabling system and should be of great assistance when replacing worn or broken cabling.
* Figure A-4--This figure provides a detailed view of the differential wrist joint.
* Figure A-5--This figure provides a detailed, exploded view of the hand.

The electrical schematic is not provided here, as this has been throughly discussed in Section VI. Finally, the explanation of the mechanical operation has been discussed in Section IV. Should the user desire to disassemble the arm or make repairs, it is strongly suggested that Section IV be throughly studied once again.

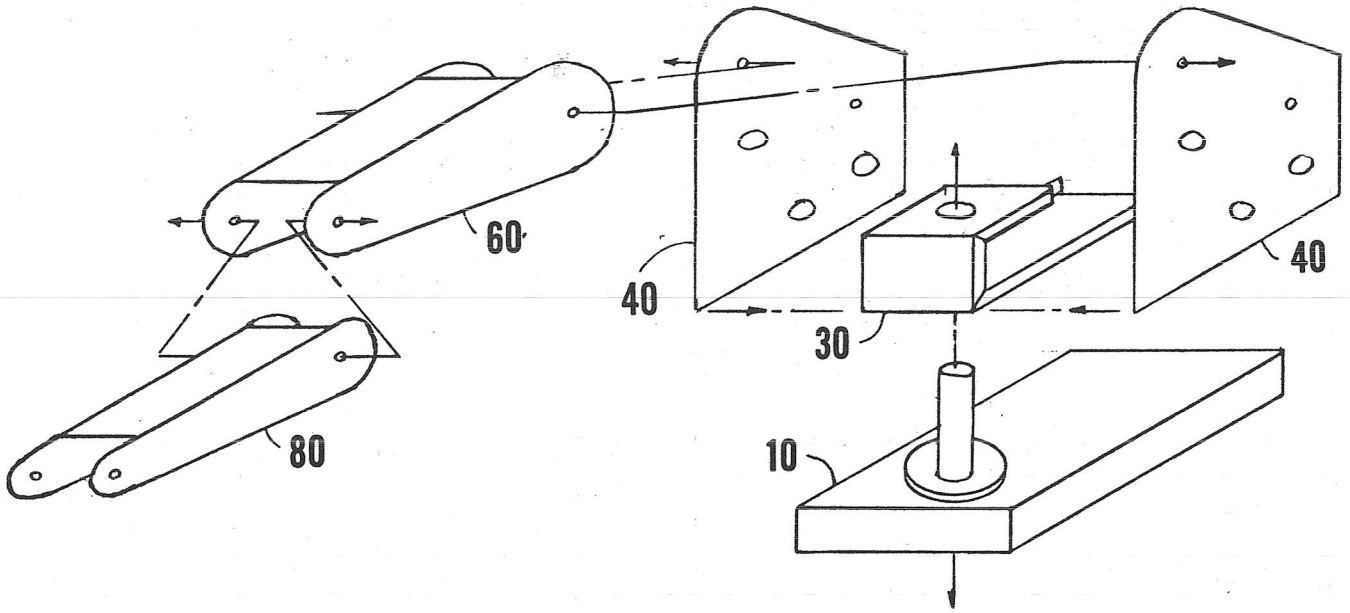



Figure A-1 Exploded View of Major Structural Components

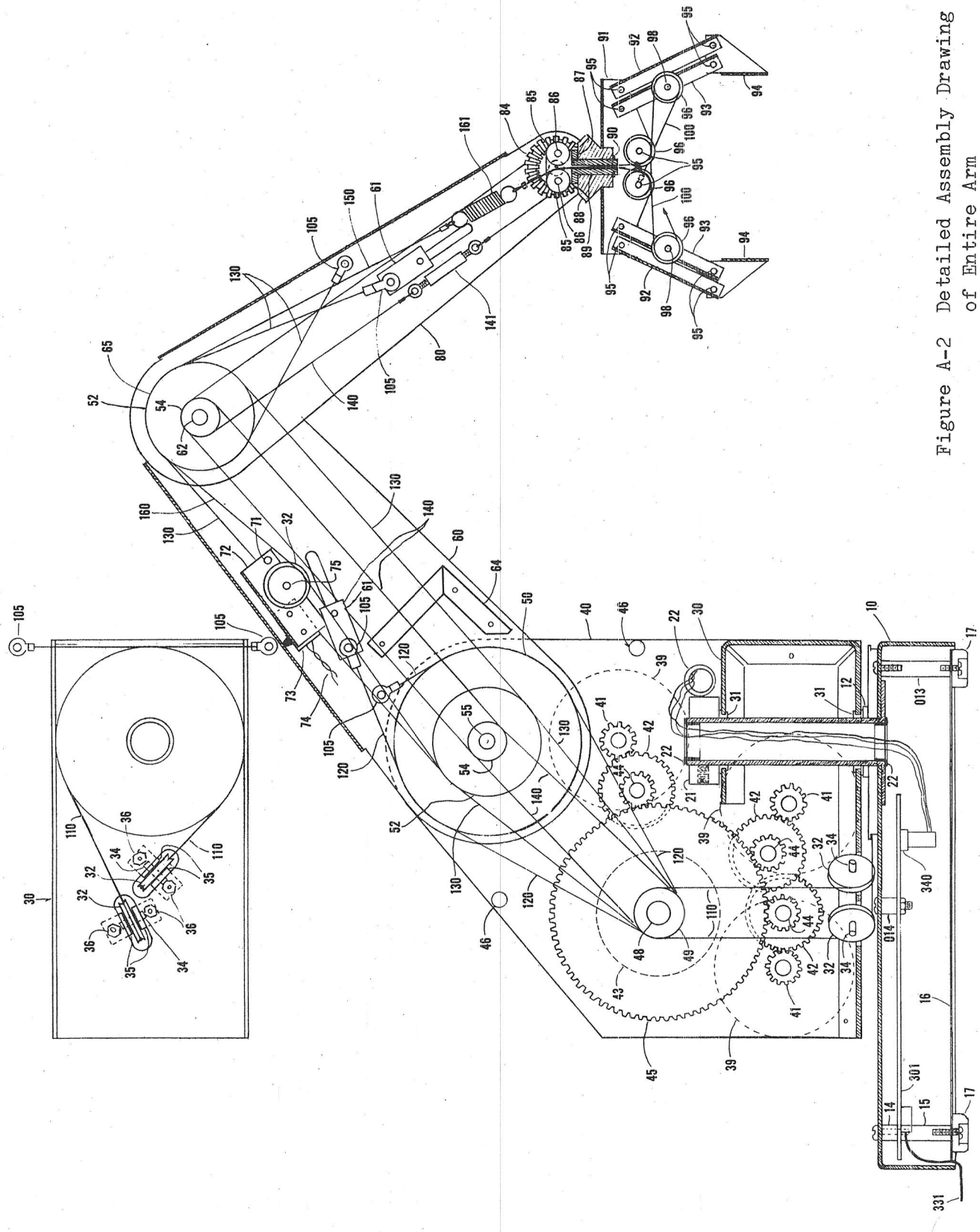


Figure A-2 Detailed Assembly Drawing of Entire Arm

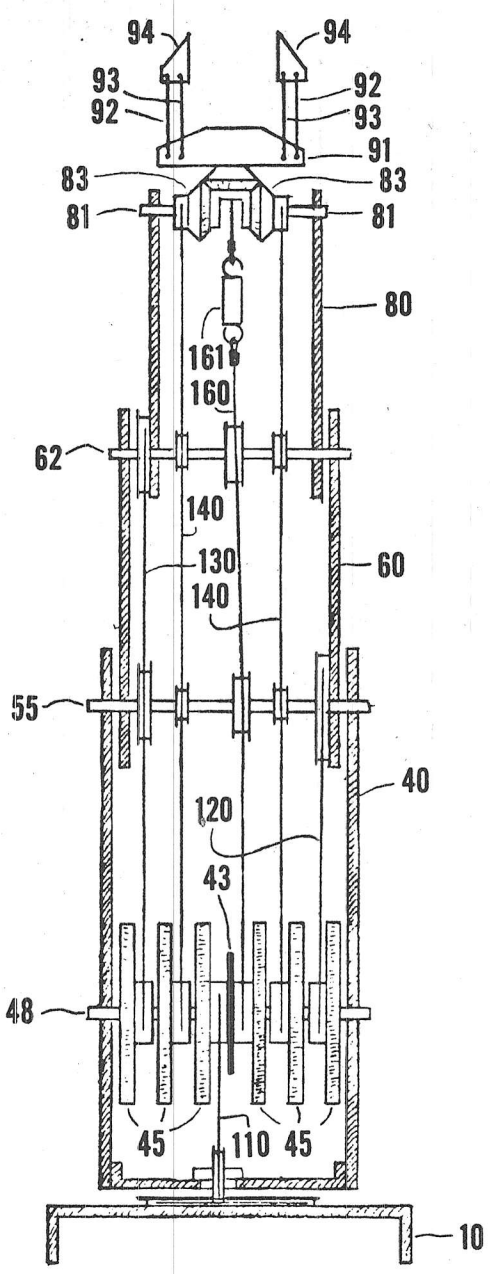



Figure A-3 Detailed view of Cabling Details of MiniMover-5

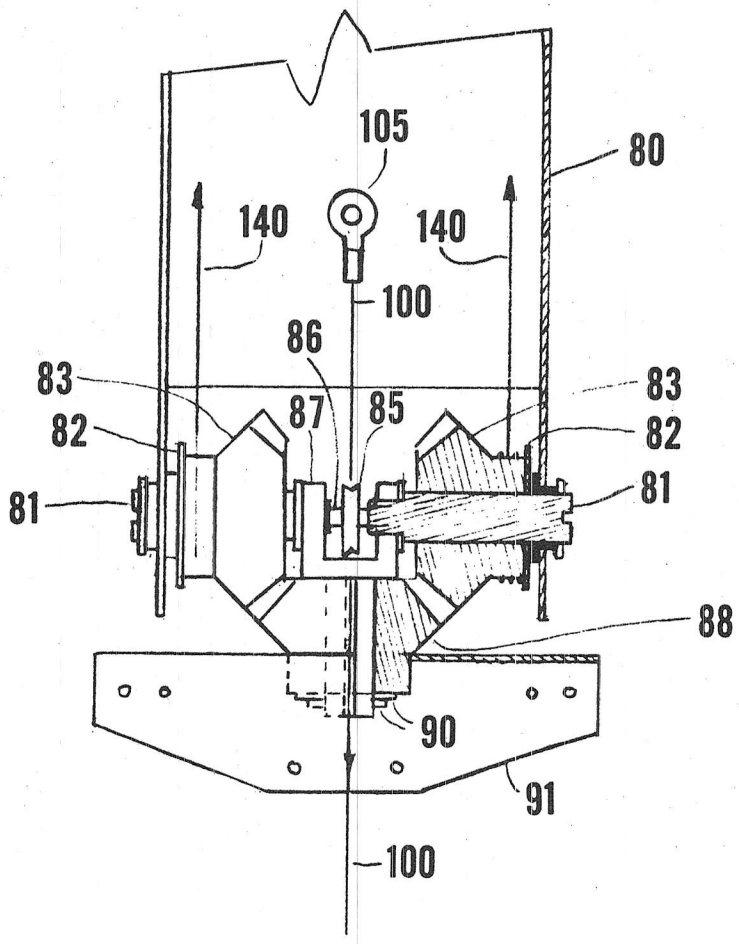



Figure A-4 Detailed View of Differential Wrist Drive

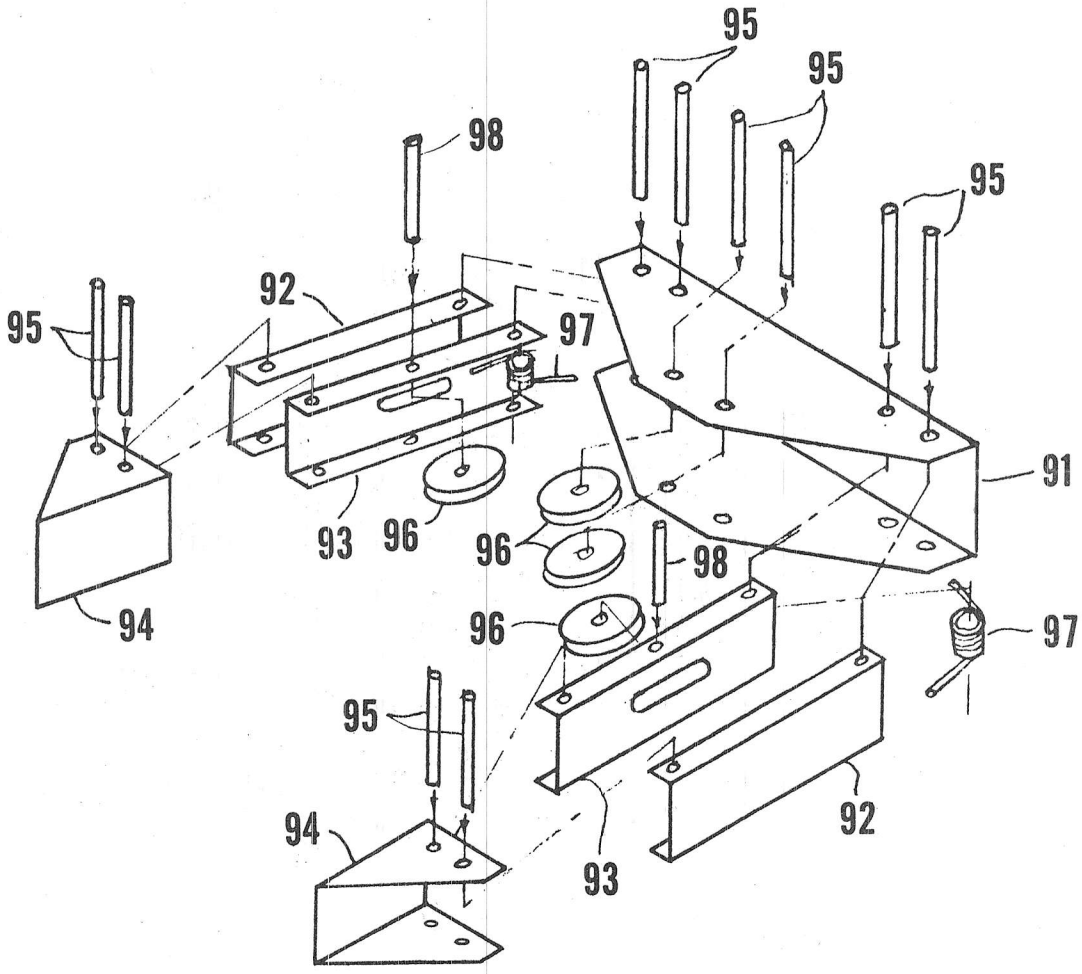



Figure A-5 Exploded View of Hand Assembly

Table A-1

Parts List

| PART NO. | DESCRIPTION | QTY |
|---|---|---|
| 5010 | BASE | 1 |
| 5012 | THRUST WASHER | 1 |
| 5013 | SPACER, LONG | 2 |
| 5014 | SPACER, SHORT | 4 |
| 5015 | SPACER, MEDIUM | 2 |
| 5016 | BASE COVER | 1 |
| 5017 | RUBBER FEET + SCREW | 4 |
| 5021 | COLLAR AND SET SCREW | 1 |
| 5022 | INSULATING BUSHING | 4 |
| 5030 | BODY BOX | 1 |
| 5031 | BASE BEARING | 2 |
| 5032 | BASE IDLER | 2 |
| 5034 | BASE IDLER AXLE | 2 |
| 5035 | BASE IDLER BRACKET | 4 |
| 5036 | BASE IDLER SCREW | 4 |
| 5039 | STEPPER MOTOR WITH CONNECTOR | 6 |
| 5040 | SIDE PLATE | 2 |
| 5041 | MOTOR PINION AND SET SCREW | 6 |
| 5042 | COMBINATION GEAR | 6 |
| 5043 | CENTER DRIVE WASHER | 1 |
| 5044 | COMBINATION SHAFT | 3 |
| 5045 | DRIVE GEAR AND SET SCREW | 6 |
| 5046 | BODY SPACER | 2 |
| 5048 | DRIVE SHAFT AND RETAINERS | 1 |
| 5049 | DRIVE WASHER | 6 |
| 5050 | SHOULDER DRIVE PULLEY | 1 |
| 5052 | LARGE IDLER PULLEY | 3 |
| 5054 | SMALL IDLER PULLEY | 4 |
| 5055 | SHOULDER SHAFT AND RETAINERS | 1 |
| 5060 | UPPER ARM | 1 |
| 5061 | TENSIONING MECHANISM AND THUMB SCREWS | 3 |
| 5062 | ELBOW SHAFT AND RETAINER | 1 |
| 5064 | UPPER ARM STIFFENER | 1 |
| 5065 | ELBOW DRIVE PULLEY | 1 |
| 5071 | SENSE BRACKET SHAFT AND RETAINERS | 1 |
| 5072 | SENSE BRACKET | 1 |
| 5073 | MICRO-SWITCH | 1 |
| 5074 | SWITCH LEADS | 2 |
| 5075 | SENSE IDLER SHAFT AND RETAINERS | 1 |
| 5080 | FOREARM | 1 |
| 5081 | WRIST SHAFT, RETAINER AND WASHER | 2 |
| 5082 | WRIST WASHER | 2 |
| 5083 | WRIST MITER GEAR AND SET SCREW | 2 |

| | | |
|---|---|---|
| 5085 | WRIST IDLER PULLEY | 2 |
| 5086 | WRIST IDLER AXLE | 2 |
| 5087 | WRIST YOKE | 1 |
| 5088 | HAND MITER GEAR | 1 |
| 5089 | HAND SHAFT | 1 |
| 5090 | HAND WASHER AND RETAINER | 1 |
| 5091 | HAND HOUSING | 1 |
| 5092 | OUTER LINK | 2 |
| 5093 | INNER LINK | 2 |
| 5094 | GRIP | 2 |
| 5095 | LINK PIN | 10 |
| 5096 | HAND IDLER PULLEY | 4 |
| 5097 | TORSION SPRING | 4 |
| 5098 | INNER LINK PIN | 2 |
| 5100 | HAND INTERIOR CABLE | 2 |
| 5105 | CABLE TERMINATION | 9 |
| 5110 | BASE CABLE | 1 |
| 5120 | SHOULDER CABLE | 1 |
| 5130 | ELBOW CABLE | 1 |
| 5140 | WRIST CABLE | 2 |
| 5141 | WRIST TENSIONING SPRING | 2 |
| 5142 | WRIST TENSIONING TURNBUCKLE | 2 |
| 5160 | HAND DRIVE CABLE | 1 |
| 5161 | HAND GRIP SPRING | 1 |
| 5301 | PCB, COMPLETE | 1 |
| 5310 | IC17 74LS00 | 1 |
| 5311 | IC14, IC18 74LS02 | 2 |
| 5312 | IC15 74LS04 | 1 |
| 5313 | IC7-13 74LS75 | 7 |
| 5314 | IC16 74LS138 | 1 |
| 5315 | IC19 74LS366 | 1 |
| 5316 | IC1-6 UDN5707A | 6 |
| 5330 | DC POWER CORD AND STRAIN RELIEF | 1 |
| 5331 | C1,40 CONDUCTOR RIBBON CABLE ASSEMBLY | 1 |
| 5340 | C2-C7,8-PIN MOTOR CONNECTOR | 6 |
| 5345 | JUMPER SET (2) FOR 8-BIT INTERFACE | 1 |
| 5350 | CR1, ZENER DIODE 1N4748 | 1 |
| 5351 | CR2, POWER DIODE MR750 | 1 |
| 5360 | R1, 1.5K RESISTOR NETWORK | 1 |
| 5370 | X1, VOLTAGE REGULATOR 7805 | 1 |
| 5371 | HEAT SINK | 1 |
| 5380 | C1-C4, 0.1UF 16V CAPACITOR | 4 |
| 5381 | C5-C6, 10UF 25V CAPACITOR | 2 |
| 5399 | PCB BARE | 1 |

REFERENCES


1. Abraham, R. G., et. al., "State-of-the-Art in Adaptable-Programmable Assembly Systems," Report No. 77-6G1-APAAS-R3, Pittsburgh, Pennsylvania, (NTIS Accession No. PB 270054/AS) (May 1977).

2. Agin, G. J., "An Experimental Vision System for Industrial Applications," Proc. Fifth Int. Symp. on Industrial Robots, pp. 135-148, Chicago, Illinois (September 1975).

3. Bober, R. E., "Taking the First Step," Byte, pp. 35-112 (February 1978).

4. Finkel, R., "AL, A Programming System for Automation," Stanford University Artificial Intelligence Laboratory, Memo AIM-243, STAN-CS-74-456, Stanford, California (November 1974).

5. Guilder, J., "Focus on Stepping Motors," Electronics Design, p. 48 (October 25, 1977).

6. Hill, J. and A. Sword, "Studies to Design and Develop Improved Remote Manipulator Systems," NASA CR-2238, National Technical Information Service, Springfield, Virginia, pp. 41-54, (April 1973).

7. Johnsen, E. G. and W. R. Corlis, "Teleoperators and Human Augmentation," NASA SP-5047, Superintendant of Documents, Washington D.C. (1967).

8. Kelly, R. et al., "A Robot System which Feeds Workpieces from Bins into Machines," Proc. Ninth Int. Symp. on Industrial Robots, pp. 339-355, Washington, D.C. (March 13-15, 1979).

9. Konstantinov, M. S., "Structure and Kinematic Analysis of Robots and Manipulators," Proc. Fourth Int. Symp. on Industrial Robots, pp. 331-328, (November 1974).

10. Lieberman, L. I. and M. A. Wesley, "The Design of a Geometric Data Base for Mechanical Assembly," IBM Research Paper No. RC 5489 (June 1975).

11. Nevins, J. L., Sensors for Industrial Automations, McGraw-Hill Yearbook, Science and Technology, McGraw-Hill Book Company (1975).

12. Paul, R. L., "Manipulator Path Control," Proc. Int. Conf. on Cybernetics and Society, pp. 147-152, San Francisco, California (September 1975).

13. Paul, R. L., "Modelling, Trajectory Calculation and Servoing of a Computer Controlled Arm," Memo AIM-177, Stanford Artificial Intelligence Laboratory, Stanford, California (November 1972).

14. Rosen, C. A. and D. Nitzan, "Use of Sensors in Programmable Automation," Computer, pp. 12-23 (December 1977).

15. Seim, T. A., "Applying Microprocessors to Machine Tool Design, Part 1," Computer Design, p. 86 (March 1980).

16. Seim, T. A., "Applying Microprocessors to Machine Tool Design, Part 2," Computer Design, p. 90 (April 1980).

17. Tsuboi, Y. et al., "A Mini-Computer Controlled Industrial Robot with Optical Sensors in Gripper," Proc. Int. Symp. on Industrial Robots, (May 29-31, 1973).

18. Watson, P. C., "A Multi-Dimensional System Analysis of the Assembly Process as Performed by Manipulator," Charles Stark Draper Labs Report P-364, Massachusetts Institute of Technology, Cambridge, Massachusetts (August 1976).

19. Weisel, W. and A. Katoh, "Beachheads for Robotics," Proc. Fifth Int. Symp. on Industrial Robots, pp. 1-10, Society of Manufacturing Engineering, Dearborn, Michigan (September 1975).

20. Whitney, D. E., "Resolved Rate Control of Manipulators and Prosthesis," IEEE Transactions on Man-Machine Systems, Vol. MMS-10, pp. 47-53 (June 1969).

21. Will, P. M., "Computer Controlled Mechanical Assembly," Proc. Fifth Int. Symp. on Industrial Robots, pp. 153-170, Chicago, Illinois (September 1975).

INDEX