CALIFORNIA STATE UNIVERSITY, NORTHRIDGE

ROBOTIC SYSTEM

WITH COMPUTER VISION

A thesis submitted in partial satisfaction of the requirement for the degree of Master of Science in

Electrical and Computer Engineering

by

Soo-Man Lee

May 1985

The Thesis of Soo-Man Lee is approved:

____
Professor Yuh Sun

____
Professor V. Anderson

____
Professor Robert Y. Wong, Chairman

California State University, Northridge

To my Eunjin and parents

for their love and patience

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

ROBOTIC SYSTEM
WITH COMPUTER VISION

by

Soo-Man Lee

Master of Science in Electrical
and Computer Engineering

This thesis is intended to study the techniques and to solve the problems in the application of video processing to an automatic robotic system. The contents of this thesis are:

a. Description and explanation of the Video Processign Unit, how it is integrated with a television camera, television monitor, and a system controller, Apple IIe.

b. Description and explanation of the Five-Axis Robot, TeachMover, and how it is operated by the system controller, the Apple IIe.

c. Connection and operation of the video camera, video monitor, Video Processing Unit, robotic arm, and the controller, the Apple IIe.

d. Flow chart of controlling program for Apple IIe.

e. Presentation of experimental program.

The basic components of the system are an Apple IIe personal computer with two serial ports, a Video Processing Unit, television camera, television monitor, lighting system, and the TeachMover Robotic Arm.

The system was designed and developed and software programs written. Test results indicated the system operated properly and its performance satisfied the design objectives.

Chapter 1

INTRODUCTION

Pattern recognition and image processing techniques have been developed and applied to automatic visual measurement and inspection. These techniques are used to detect an object and to determine the object's location, size and shape. Industrial automation for assembly, automatic alignment of the assembly for testing and component recognition have demanded the development of a human-like robot. Including the benefit for increasing productivity or improving the quality of life, the life of workers now doing repetitive and sometimes hazardous tasks would be taken care of by using sophisticated robots. Furthermore, using computer vision techniques, the positions and orientations of an object within the field-of-view of the system can be determined.

This thesis involves the design and development of an automated robotic system with computer vision using a Video Processing Unit. The system uses an Apple IIe microcomputer to process data transmitted from the Video Processing Unit (VPU) and sends the command to the robot arm, TeachMover.

First, the data sent from the VPU is processed to analyze where the object is located, its size and orientation.

Second, a set of coordinates and signals were generated and sent to the TeachMover to reach the object.

Finally, the robot arm, moving to that object grasps it from the proper direction and places the object as required by the operations.

Algorithms were developed to program the computer to process the data to recognize the object and to command the robot-arm to perform the proper operations. Cubic and cylindrical objects were used for testing.

Chapter 2

ROBOTIC ARM WITH COMPUTER VISION SYSTEM

A functional block diagram of the system is shown in Figure 1. The basic components of the system are:

a) Apple IIe personal computer with two serial ports

b) DAGE-MTI, INC. MK 11 series vidicon type video camera with 10 mm lens

c) The Rank Videometrix Video Processing Unit

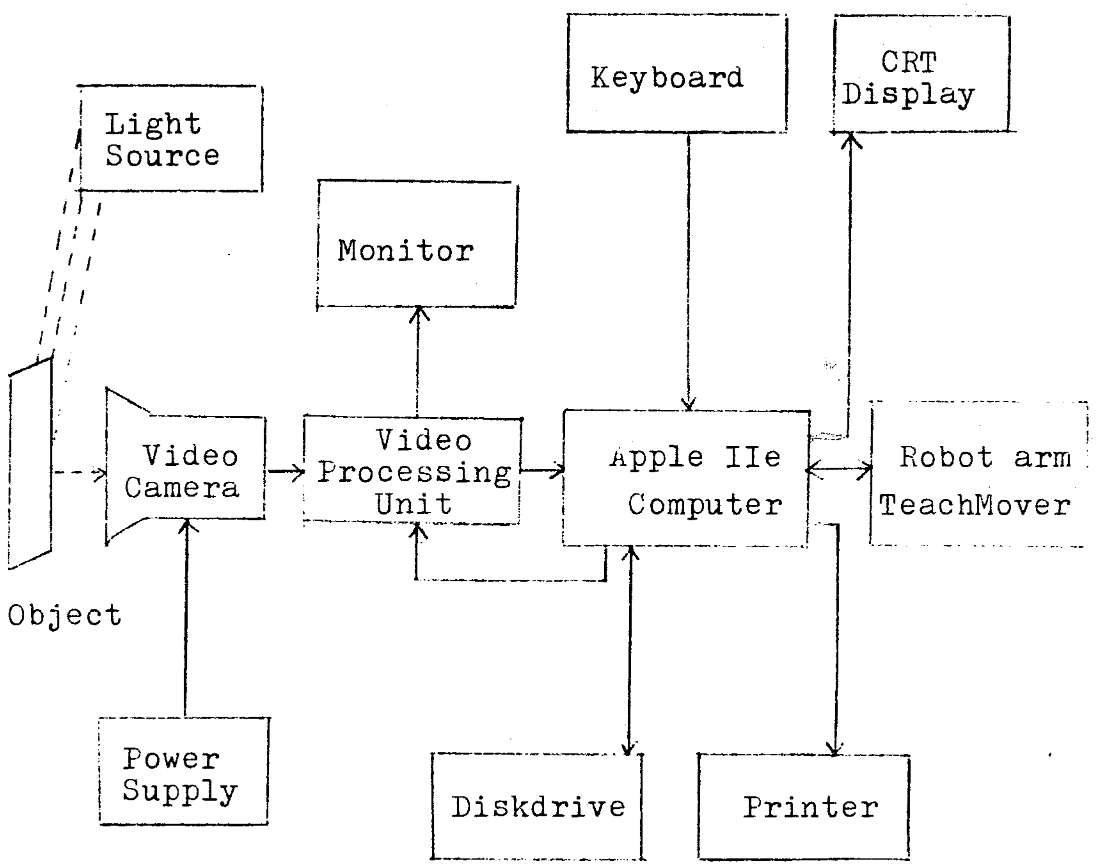
d) Video monitor

e) Robot-arm; Five-Axis Robot Model TCM (TeachMover)

f) 5-1/4 in. single sided, double density floppy disk storage system

g) Okidata printer

Figure 2 shows the physical arrangement of the system.



System Structure Configuration

Figure 1

Experimental Equipment

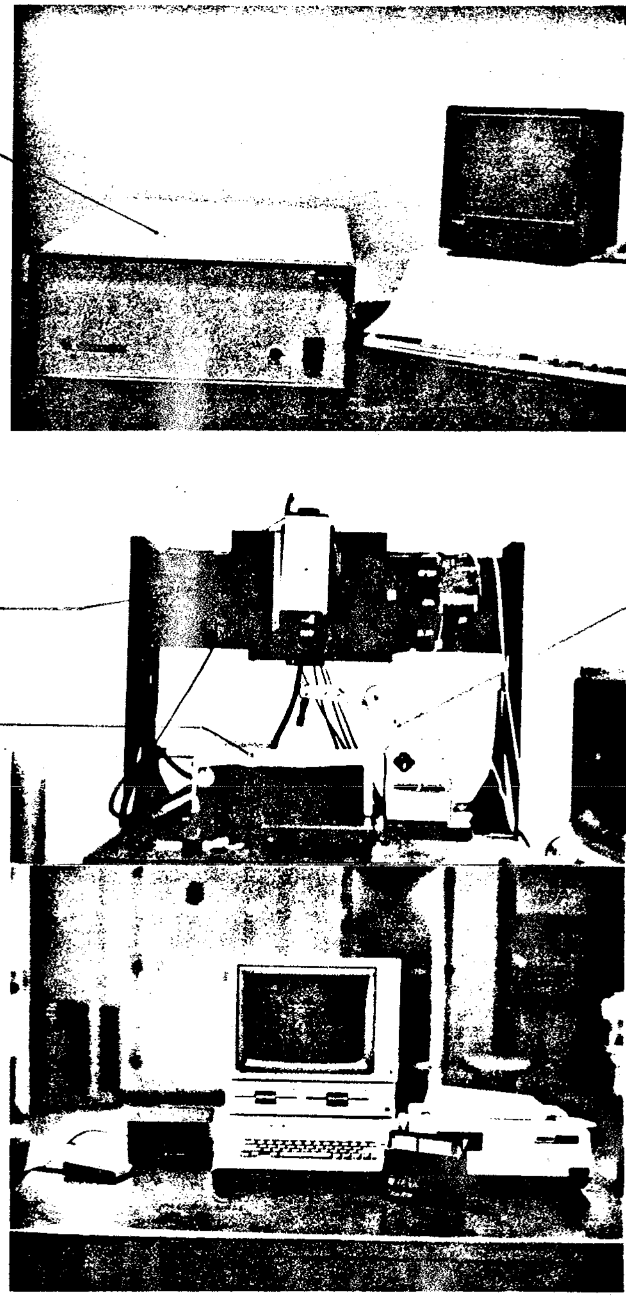


Figure 2

System Arrangement

Chapter 3

VIDEO PROCESSING SYSTEM

3.1 Video Camera with Lens

The optical information is reflected by a surface of an object and its surrounding top into a video camera under a light source. The video camera converts the optical information into an electrical signal. The 10 mm lens was used to cover a 10 x 10 inch field-of-view.

3.2 Video Processing Unit (VPU)

VPU was used to convert analog signal into digital form and analyze the digitized data. The Rank Videometrix Video Processing Unit (VPU) is a general purpose device which was integrated with a TV camera, TV monitor and a system controller-Apple IIe to provide an automatic dimensional measuring system. Its basic function is to process the video signal generated by the camera and extract various edge data which can be used to determine dimensions of the object being viewed. The unit responds to commands received from the system controller and returns various status and measurement data.

3.2.1 Functional characteristics

Measurement Window and Crosshairs -

The VPU superimposes a rectangular "measurement window" and a set of crosshairs on the TV monitor. The location can be changed under software control. The window surrounds that portion of the video scene which the VPU actually "sees." That is, it ignores any part of the scene outside the window. Thus the window can be used to isolate specific areas of the scene for analysis. The horizontal and vertical crosshairs are independently controllable. They enable gathering data along a particular scan line, a feature which is useful in many measurement applications.

The crosshairs are always confined to be within the window and are never allowed to be closer than 8 pixels or lines from a window side. The left and right window slides cannot be closer than 12 pixels. The same is true for the top and bottom. The VPU will override system controller commands that violate these constraints. At full size the window sides are located as follows:

TOP	=	0 lines
BOTTOM	=	400 lines
LEFT	=	8 pixels
RIGHT	=	503 pixels

Edge Detection

In a sense, the VPU is an analog-to-digital converter. The continuous analog video signal generated by the camera is sampled at a specific time interval and compared with a pre-set threshold. Sampled voltages above the threshold are declared to be "1" and those below are "0". The result is "digitized" bi-level video,, that is, pure black and white with no gray. By saving all the 0's and 1's in memory it would be possible to digitize the entire scene. For most measurement applications, however, this is neither required or desirable. Typically, edge transitions form the basis for measurement. Therefore, the VPU was designed to "remember" only the location of the first edge transition (or alternately the last) that it encounters on each horizontal TV scan line. Similarly, it remembers edge transitions in the vertical direction yielding, in effect, a two-dimensional outline of the image. There are 400 usable scan lines and each is divided into 500 elements by the VPU. The data is stored in a table in its computer memory. Upon command the VPU writes to the table one of four types of data:

400 X (horizontal) leading edges (first transitions)
400 X trailing edges (last transitions)
500 Y (vertical) leading edges
500 Y trailing edges

The terms "X", "Y", "leading" and "trailing" are used frequently throughout the remainder of this manual. The VPU can transfer all this data to the system controller upon demand, which is useful in some cases. More typically, however, the system controller would request only the minimum of all the edge values, or the maximum, or the one coinciding with the current crosshair location, etc. Appendix A describes all the various possibilities under "Data Gathering Commands."

Centroids and Areas

Besides defining edge locations the VPU can compute the centroid and area of the image in the window. This feature is useful in finding the area or centroid (area moment) of an object that is entirely within the measurement window. The process is performed entirely in hardware and runs at the video scan rate, that is, a centroid/ area can be computer thirty times a second. Unlike the portion of the VPU which does edge detection, the centroid/area hardware uses all edge transitions, not just leading and trailing. The result is a true area/area moment. The centroid is referenced to the upper left hand corner of the window (when at its maximum size). This corner always represents 0,0.

Thresholding

The analog video signal voltage for a given scene covers a range representing the blackest black to the whitest white. The VPU contains peak detectors which in effect remember these extremes over the entire frame. It is then able to compare the intensity of every other point in the scene relative to these peaks in making its "0" or "1" determination as previously described under "EdgeDetection." This process is called thresholding.

The comparator setting can be anywhere from 0 to 100% of the range defined by the peak detectors and is under software control. A typical setting is 50% but sometimes various lighting and surface conditions require some experimentation to find the proper setting. Appendix A describes the threshold setting command.

3.2.2 System interconnects

Hardware setup for vpu system

Figure 3 shows cable interconnects for a the system consisting of the Video Processing Unit, TV camera, monitor, and a system controller. Video cables were the coaxial, shielded type.

Table 1 shows the pin assignments for the RS-232 connector (DB-25S) on the rear panel, which were connected to Apple IIe with serial port. And, Table 2 shows the switch settings on the first pc board inside the VPU.

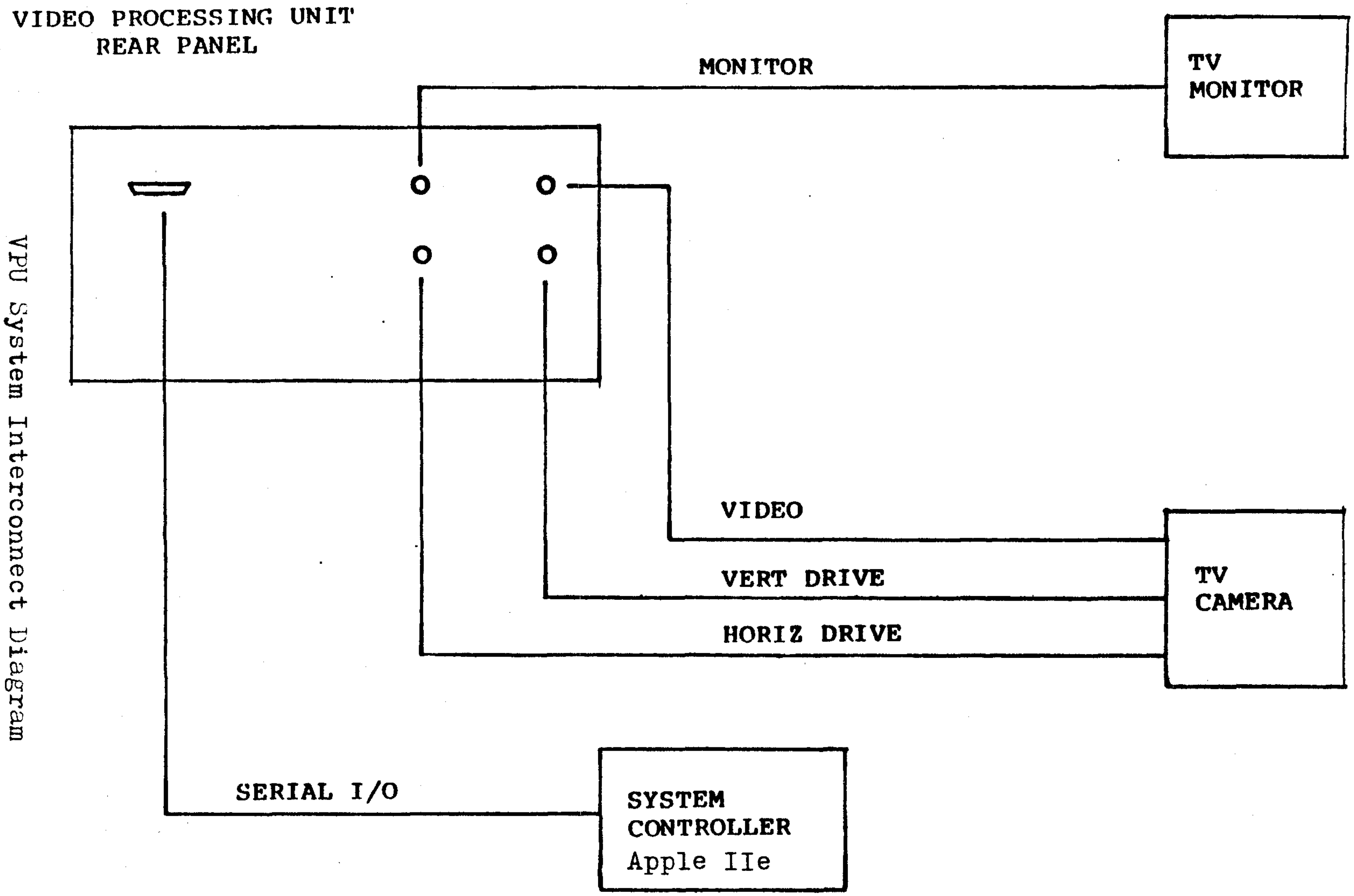
3.2.3 RS-232 specifications

The VPU configuration used with the Apple IIe was as follows:

2400 Baud
7 Bit Characters (ASCII Standard)
Even Parity
1 Stop Bit

The VPU software does not support the following RS-232 functions:

Clear to Send
Request to Send
Data Terminal Ready
Data Set Ready



VPU System Interconnect Diagram

Figure 3

SERIAL I/O PIN ASSIGNMENTS

PIN NO.	SIGNAL NAME
1	Protective Ground
2	Received Data
3	Transmitted Data
4	Clear to Send
5	Request to Send
6	Data Terminal Ready
7	Signal Ground
8-19	Unused
20	Data Set Ready
21-25	Unused

Connector - DB-25S

Table 1

SWITCH SETTINGS

Bit No.	1	2	3	4	5	6	7	8
VPU EOL Sequence				X				
Daisy/Non-daisy					X			
Baud Rate						X	X	X

EOL Sequence (bit 4)

0 = CR on VPU output

* 1 = CR/LF

Daisy/Non-daisy (bit 5)

* 0 = Standard I/O

1 = Daisy-chained I/O

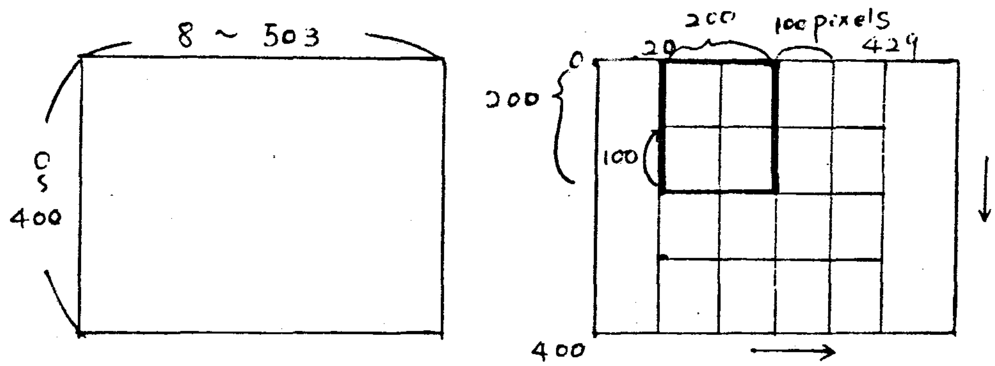
Baud Rate (bits 6-8)

000	19200
001	9600
010	4800
* 011	2400
100	1200
101	600
110	300
111	150

* : setting for experimental program

Table 2

3.3 Procedure for Detection of Object



1. If the width of the object is less than 100 pixels, it is less than two block lengths. Therefore, it never exceeds the two blocks (one block is 100 pixel-wide).
2. The object is adjusted to be wider than 30 pixels.
3. Then, the step of Y should be less than 30 pixels. Let Y step be 20.
4. a) Set y(I) at first 20 of a selected 200 x 200 square window.

   For I=20 to 200 step 20

   Find IE/X/L and IE/X/T

   b) If (XT-XL) << 30

   That is, if (XT-XL) < 5 then it's just one portion of arc of an object. Then, save it and continue to measure the remaining arc or line.

   20 < width < 100

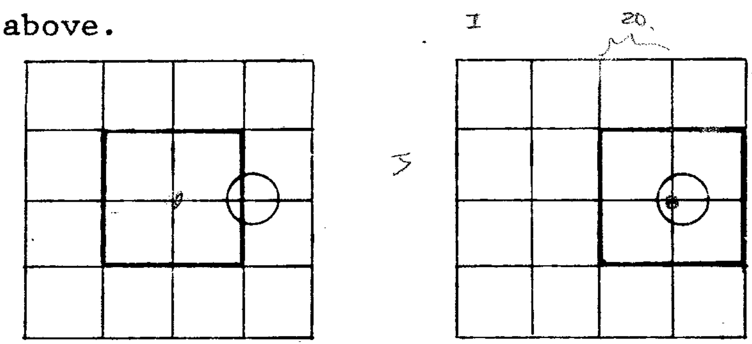
c) If (XT-XL) > 5 then, this block might hold the object then treat it as the object and go to step d)

If (XT-XL) > 1000 then, that point may contain noise and go to e.

d) Accumulate the number

$ps(I,J) = p(I,J) + 1$

e) Go to next block (move the window) and check as above.



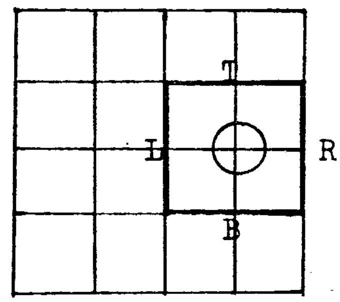
Window ps(2,2) Window ps(2,3)

$ps(2,2) < ps(2,3)$

5. The maximum number of ps(I,J) is implemented as a holding block. Therefore, n^{th} block is holding the object.

Save I,J number into EX and EY and set up that window, and analyze the data using commands to get the data for the object and compute where it is placed and how much it is rotated if it is cubic.

6. In order to detect where it is located, locate the window around the object:



AG/T/B/L/R

a) Now with EX, EY, compute T, B, L, R to locate the window; AG/T/B/L/R.

b) To get the top point of the object compute MI/Y/L.

To get the bottom point of that object, compute MX/Y/T.

To get the left-most point of the object, compute MI/X/L.

To get the right-most point of the object, compute MX/X/T

c) From I=T to B step 5

Compute IE/X/L and IE/X/T.

Find Max. (XT-XL)

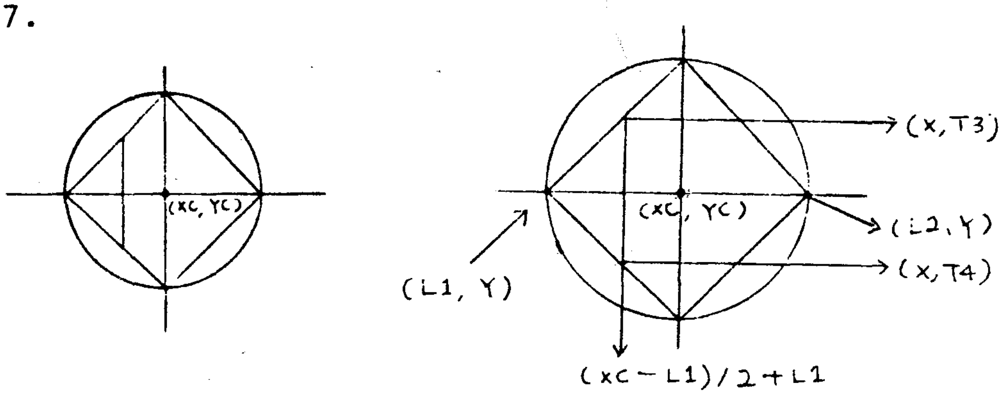
IE/Y/L and IE/Y/T

d) See if these values matche 6(b).

If almost same or less than 10% error then, there is no noise and data is good. And go to e. Else, go to c) and check data or check threshold again.

e) If the difference between X of the top point and X of the bottom point is less than 5, and the difference between Y of the left-most and Y of the right-most point is less than 5, THEN it is 45 degree rotated cubic or cylinder top, and go to next step 7. Else, it is surely cubic go to 7-c to check the degree of rotation.

7.

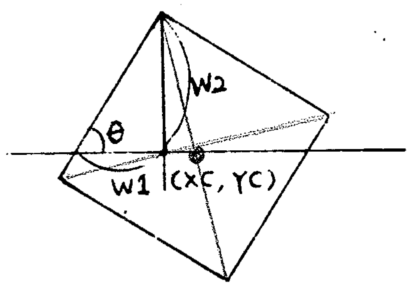


a) Using CC or CN/B, get the center of that top surface of the object. Save it to (XC,YC). After getting the point (L1,Y), and (L2,Y) calculate (XC - L1)/2 + L1 and set up the crosshair using the command CS/V/((XC - L1)/2 + L1). Next, get the point (X,T3) and (X,T4)

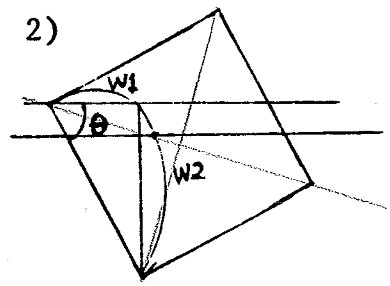
b) If ABS((T4-T3) - (L2-L1)/2) < (L2-L1)/4 THEN the object is cubic and rotated degree is 45 degrees else, it is a cylinder.

c) If it is cubic then, get the degrees the cubic is rotated.

1)



2)



1) First, get the centroid.

   Second, calculate the length W2 from the top to the horizontal crosshair which is set up through (XC,YC).

   Third, calculate W1.

   Finally, using $\theta = \text{Tan}^{-1}(W2/W1)$

2) If the left-most point is higher than the centroid point, then again set the horizontal crosshair via that point and get $\theta = \text{Tan}^{-1}(W2/W1)$. The data needed to command Robot-Arm is compensated and corrected if there is an error and checked again by slightly different method in program.

Chapter 4

ROBOTIC SYSTEM

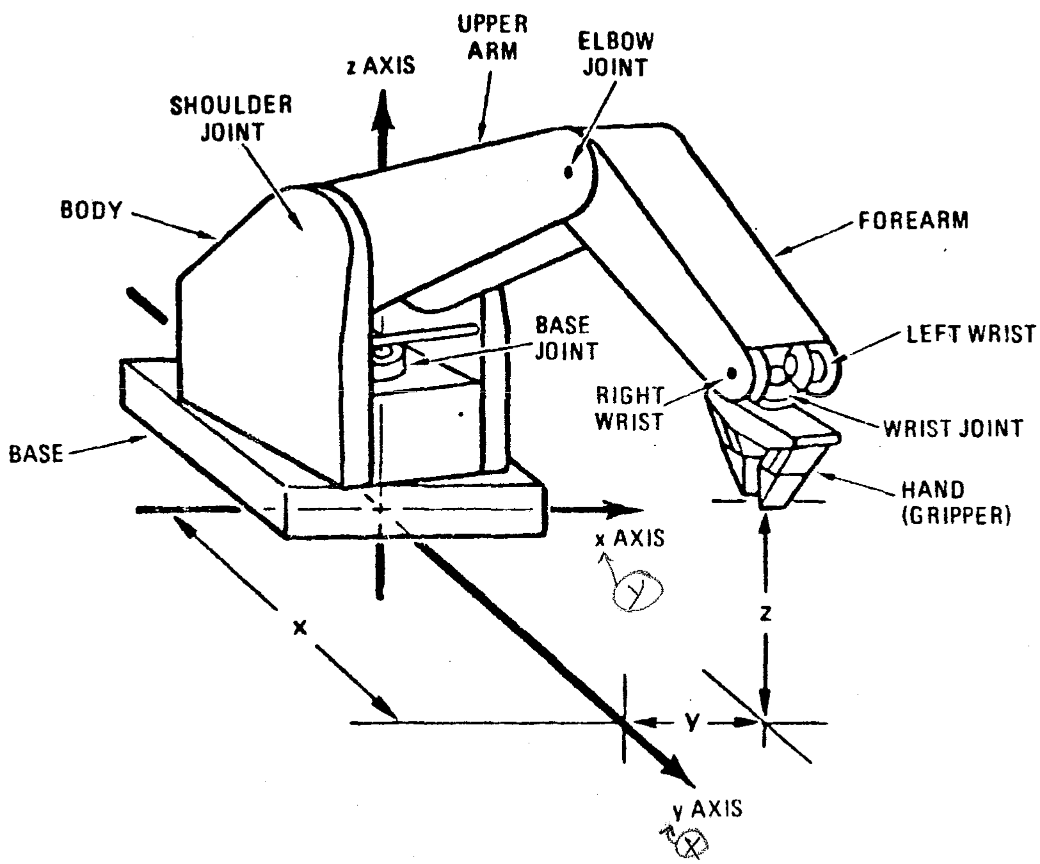
4.1 Introduction

The robotic system used for this project is the TeachMover robot arm which is a microprocessor-controlled, six-jointed mechanical arm designed to provide an unusual combination of dexterity. This project is using Serial Interface Mode, in which the TeachMover arm can be controlled by a host computer, Apple IIe via one of two built-in RS-232C asychronous serial communications lines. Major structural components are shown in Figure 4.

4.2 TeachMover Performance Characteristics

4.2.1 General

Configuration	5 revolution axes and integral hand
Drive	Electric stepper motors - Open loop control
Controller	6502A microprocessor with 4K bytes of EPROM and 1K bytes of RAM located in base of unit
Interface	Deal RS-232C asynchronous serial communications interfaces (baud rates in switch-selectable between 110, 150, 300, 600, 1200, 2400, 4800, and 9600 baud)
Teach Control	14 key - 13 function keyboard, 5 output and 7 input bits under computer control



Major Structural Components

Figure 4.

Power Requirement	12 to 14 volts, 4.5 amps DC

4.2.2 Performance

Resolution	0.011 in. (0.25 mm) maximum on each axis
Load Capacity	16 oz. (445 gm) at full extension
Gripping Force	3 lbs. (13 Nextons) maximum
Reach	17.5 in. (444 mm)
Velocity	0-7 in./sec. (0-178 mm/sec.) with controlled acceleration

4.2.3 Detailed Performance

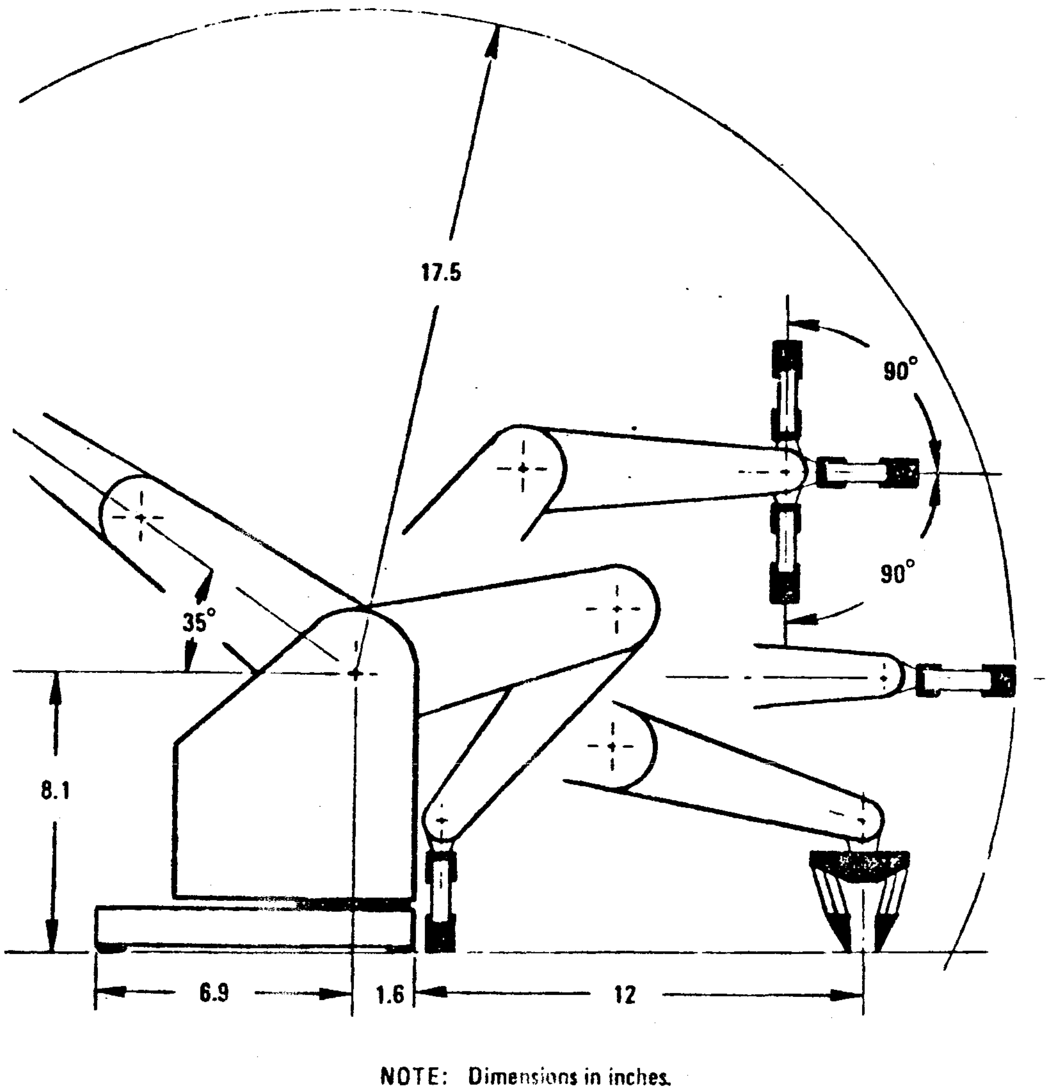
Motion	Range
Base	±90 degrees
Shoulder	+144, -35 degrees
Elbow	+0, -149 degrees
Wrist Roll	±360 degrees
Wrist Pitch	±90 degrees
Hand	0-3 i (0-75 mm)

4.2.4 Physical characteristics

Arm Weight	8 lbs. (4 kg)
Teach Control Cable Length	3.75 ft. (1150 mm)

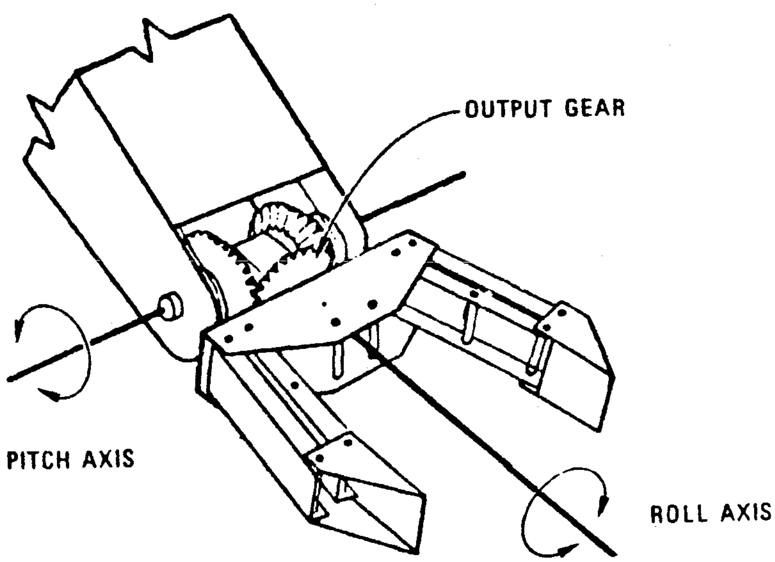
4.3 How the Motors Operate

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



Operating Envelope of the TeachMover Arm

Figure 5.



The wrist Joint

Figure 6.

pattern of currents in the motor windings. By changing the pattern of currents, a rotating magnetic field is obtained inside the motor that causes the motor to rotate in small increments or steps.

Stepper motors are not the only kind of motors used in robot arms. Some arms use servo motors with electronic feedback loops for precise position control. Unlike stepper motors, these servo motors cannot develop slippage. This advantage must be weighed against the servo motor's far greater cost.

Stepper motors are easier to control from a computer than are servo motors.

Now, in order to turn a stepper mtor in the TeachMover, a particular sequence of binary phase patterns is output to the desired motor, one pattern per step. In order to change motor direction, the order in which the phase patterns are output is simply reversed. The particular phase patterns used in the TeachMover generate a sequence known as "half-stepping;" the steps are half the size specified by the motor manufacturer. (The motors used to drive the TeachMover are specified by the manufacturer at 48 steps per revolution, but are actually stepped at 96 steps per revolution.) Compared to full stepping, halfstepping produces smoother slow-speed motions, reduces the power requirement, and improves the arm resolution by a factor of two.

The relationship between motor steps and actual joint rotation is given in Table 3.

4.4 Electronic Circuitry and Interface

4.4.1 On-board computer and memory

A circuit card houses all the internal electronics, including the 6502A Microprocessors. In technical terms, this microprocessor is an 8-bit, 2MHz chip. It is the same chip used in the Apple, Atari, and PET computers; it is used in the TeachMover to coordinate all joint motions and handle all input and output.

TeachMover firmware (permanently built-in software) is contained in another chip housing 4K bytes of read-only memory (ROM); this firmware interprets the commands given to the arm, coverting these to electrical signals the arm can obey.

The circuit card also includes chips containing 1K bytes of random-access memory (RAM). This is enough RAM to store an arm-motion program of up to 53 steps. It is possible to "piggy-back" a second set of RAMs on the first, thereby extending the program capacity to 126 steps.

4.4.2 Serial ports

Two serial interface ports allow the connections to the TeachMover to a host computer, printer, or terminal. Serial transmission speed is selectable with eight

MOTOR STEPS AND JOINT ROTATIONS			
Motor	Joint	Steps per degree	Steps per radian
1	Base	19.64	1125
2	Shoulder	19.64	1125
3	Elbow	11.55	672
4	Right wrist	4.27	241
5	Left wrist	4.27	241

Motor Steps and Joint Rotations

Table 3

standard speeds available from 110 to 9600 Baud. The 9600 Baud speed is used for Apple IIe.

4.4.3 User inputs and outputs

The computer card also contains an auxiliary parallel input/output port. Interfaces from the TeachMover to external equipment is done through a 16-conductor flat ribbon cable. Five TTL compatible user output bits can be set (to 1) or cleared (to 0) under program control to turn other equipment on or off when a given arm motion is complete. Seven TTL compatible input bits can be used to control an arm sequence when a given external condition is met.

A block diagram of the TeachMover's electronic circuitry is shown in Figure 7.

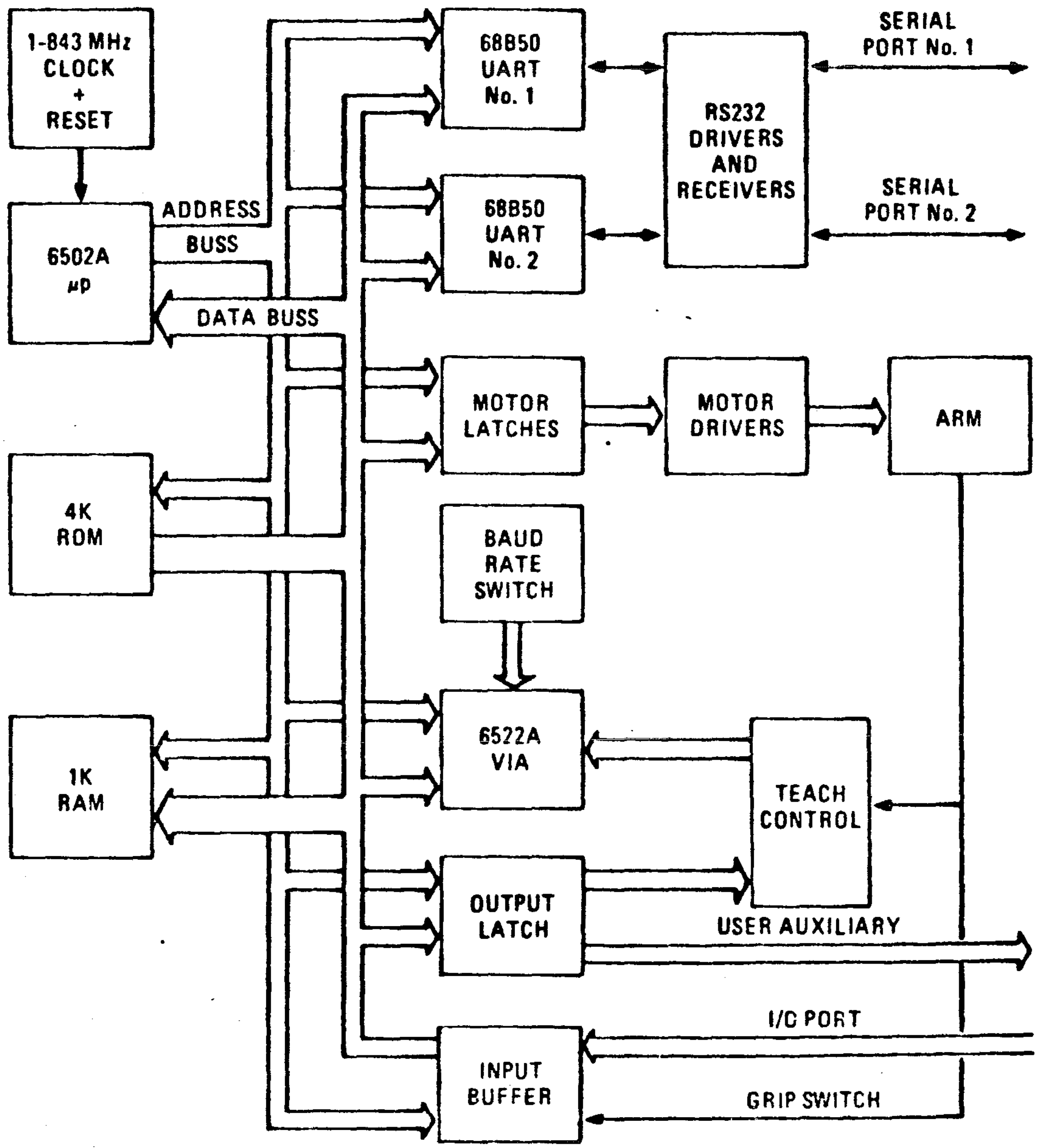
4.5 Operation from a Host Computer, Apple IIe

Connecting the TeachMover arm to a host computer or a terminal greatly extends the unit's capabilities.

4.5.1 Configuring the serial ports

"Configuring the serial ports" refers to making sure that the computer and the TeachMover can "talk" to one another. This requires taking care of the following:

1. electrical connections
2. transmission rate
3. data format
4. settings for standard interface signals



Block Diagram of TeachMover Computer and Electronics

Figure 7.

5. opening the port
6. testing the configuration

4.5.2 Electrical connections

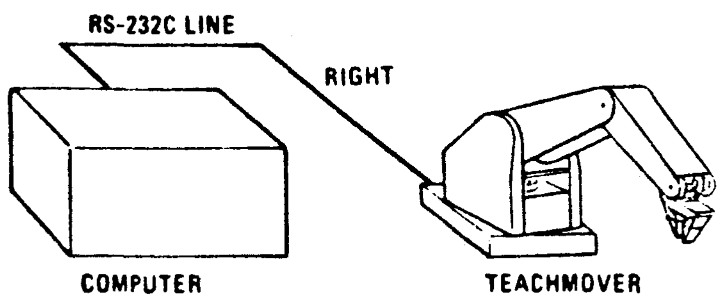
The two serial ports perform the following functions:

- Signals that enter the left port (P2) always pass through to the right port (P1) unchanged.
- Signals that enter the right port pass through to the left port unchanged, unless the signals are a series of characters beginning with an "@" sign and terminating with a <CR> (carriage return); these signals are not passed through, but are interpreted as arm commands.

Thus, to operate the arm from a host computer or a terminal, connect the computer or terminal to the TeachMover's right serial port.

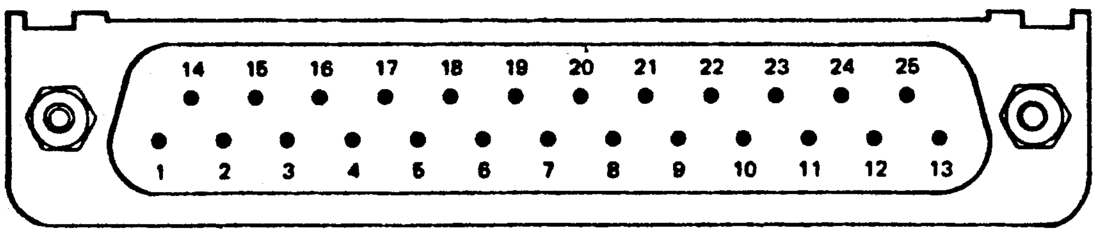
4.5.3 Transmission rate

The TeachMover is configured to operate at a transmission rate of 9600 baud (9600 bits per second), for both send and receive. You can change this rate to any of seven other standard rates by means of three switches located on TeachMover computer card (Figure 11). The available rates and the corresponding switch settings are given in Table 4.



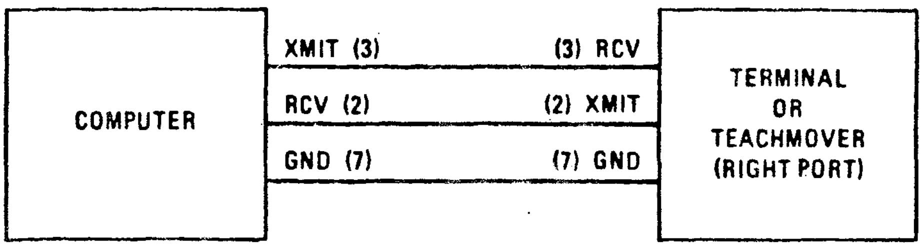
Connecting the TeachMover to Apple IIe

Figure 8.



Pin Numbering For Serial Port Connectors

Figure 9.



Apple IIe-to-Teach Mover Serial Connection

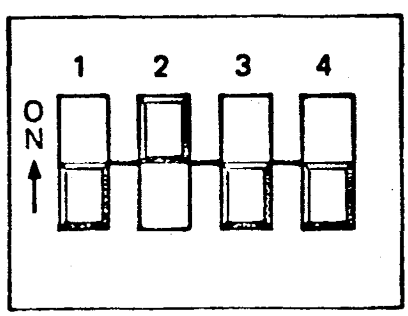
Figure 10.

BAUD RATE SELECTION			
BAUD	SW1	SW2	SW3
110	ON	ON	ON
150	OFF	ON	ON
300	ON	OFF	ON
600	OFF	OFF	ON
1200	ON	ON	OFF
2400	OFF	ON	OFF
4800	ON	OFF	OFF
9600	OFF	OFF	OFF

NOTE: SW4 is not used.

Baud Rate Selection

Table 4



Switches for Selecting Serial Transmission Rate

Figure 11.

These switches should be changed when power is off, since the switch settings are read by TeachMover firmware on power-up only.

4.5.4 Data format

The TeachMover uses the following data format:

word length = 8 bits
1 start bit
1 stop bit
no parity bit
full duplex

Especially, Apple IIe rather than specify a word length of 8, it is necessary to specify a word length of 7 plus a parity bit equal to zero. This is because this computer uses a most significant bit equal to 1 when processing 8-bits words. (It is possible to use the @ARM command to allow the robot to recognize an "@" with the eighth bit = 1, but in order to execute this command the robot must recognize the first "@" in this @ARM command. To do this the robot must receive an "@" character with the most significant bit = 0.)].

4.5.5 Standard interface signals

Some computers and terminals require logic levels on certain pins to indicate the following status conditions:

Data Terminal Ready
Clear to Send
Carrier Detect
Request to Send

The TeachMover does not use these signals, but does pass them through when it is placed in series between a computer and a terminal.

4.5.6 Serial interface commands

Ten different commands can be issued to the Teach-Mover over the serial lines. (A concise summary of all ten commands is given in Appendix D.)

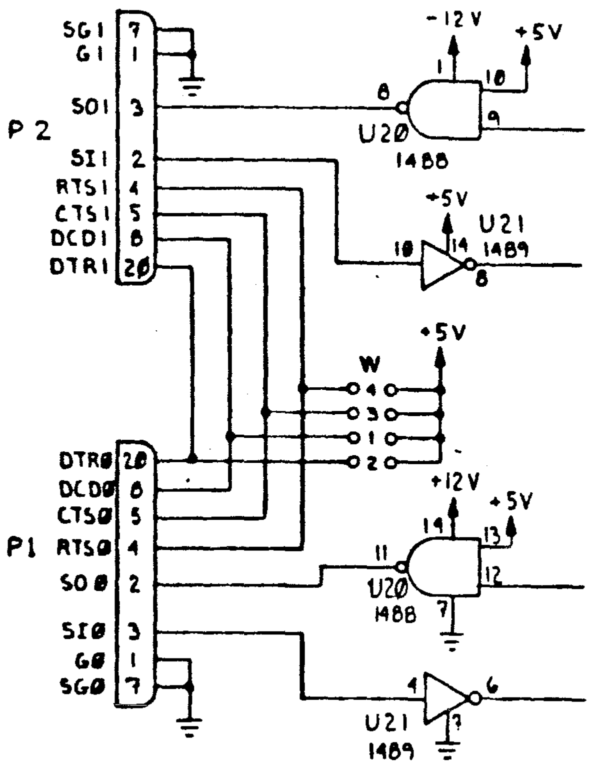
Note: All commands can be abbreviated to an "@" sign plus the first three characters--@CLO for @CLOSE, etc.

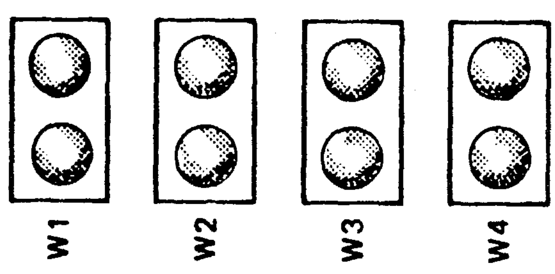
- All characters and numeric values are decimal ASCII (industry-standard character format).
- Once a serial command is executed, the teach control is left in TRAIN mode, with two exceptions:
  - @RESET leaves it in MODE mode.
  - @RUN simply runs the arm until another command stops it.
- However, the indicator lights will remain as they were before the serial command was executed. (Example: If MODE light is on, and then, say, a @CLOSE command is executed, the Teach Control will then be in TRAIN mode but with the MODE light still on.)
- To change the status of the indicator lights, use the @STEP command with all parameters set to zero except the "OUT" value (see below). No other serial command affects the status of the lights (except the closed light which

Left Port Pin No.	Description	Right Port Pin No.	Jumper
8	Data Carrier Detect	8	W1
1,7	Ground	1,7	--
3	Transmit from TeachMover	2	--
2	Receive by TeachMover	3	--
4	Request to Send	4	W4
5	Clear to Send	5	W3
20	Data Terminal Ready	20	W2

Control Lines

Table 5





Location of Jumper Connections for Serial Operations

Figure 12.

always indicates the state of the gripper switch).

The ten commands are as follows:

1.	@STEP	6.	@ARM
2.	@CLOSE	7.	@DELAY
3.	@SET	8.	@QDUMP
4.	@RESET	9.	@QWRITE
5.	@READ	10.	@RUN

4.6 Arm Initialization and Calibration

The computer in the robot keeps track of the arm by using the starting position as a reference. To run a program to operate the robot arm, the starting position was as specified in the program.

With the origin of the coordinate system located at the axis of rotation of the base of the robot arm, the location of the front edge of the base of the arm is defined as x = 1-5/8. The base is centered on the y-axis. The gripper is brought to rest, barely touching, on the spot PO, shown on the grid at X=5 and Y=0.

Because of the proper window of the VPU for the experimental program, two relative cartesian coordinates are used. One is called table top cartesian coordinates and the other called real cartesian coordinates.

The table top cartesian coordinates is placed on the table top which is 8 inch higher than the bottom where the base of the robot arm is placed. And the 0 point of the X axis (X=0) is 4 inches farther placed from the axis of rotation of the base of the robot arm.

The real cartesian coordinates is for robot arm, with origin located at the axis of rotation of the base of the robot arm.

After all the data was calculated using the table top cartesian coordinates from the VPU data, it was converted into the real cartesian coordinates as follows:

Let XR, YR, and ZR be the variables for VPU and X,Y and Z be those for robot arm.

Then, X=XR+4

Y=YR

Z=ZR+8

This conversion is shown in Figures 13 and 26.

Following the above explanation, the initialization point is:

XR=5

YR=0

ZR=0

Therefore,

X=9

Y=0

Z=8

The hand must be perpendicular to the work surface and parallel to the front edge of the base of the robot. The gripper opens as the arm is brought to this position, and then closes the gripper as the last step in setting the initial position. The gripping force is applied by the motor as the point is closing.

The arm can be brought to this starting position by moving it manually with the power off or by using the teach control with the power on.

Specifically, the cartesian coordinates of the initial configuration are:

XR=4, X=9 inches

YR=0, Y=0 inches

ZR=0, Z=8 inches

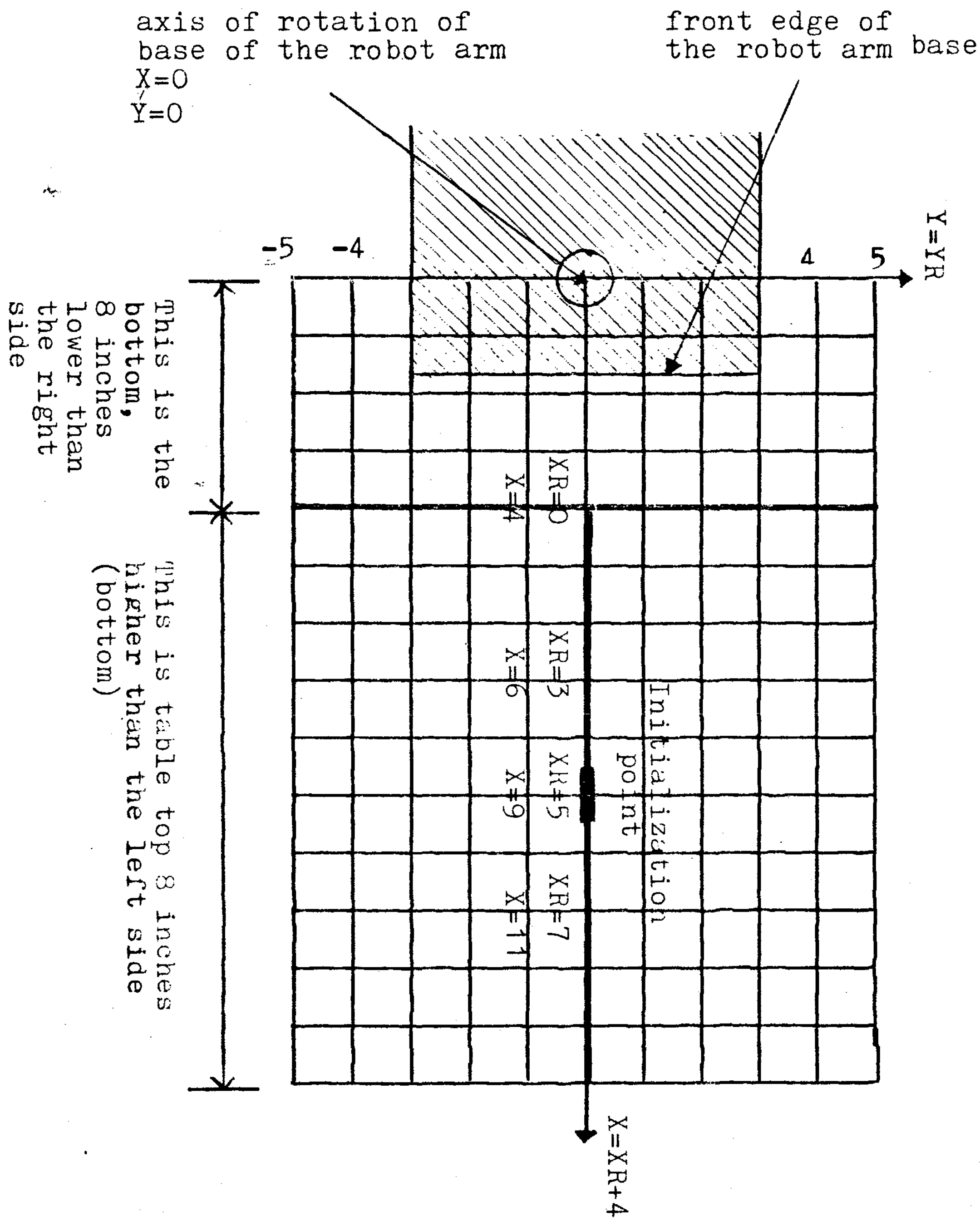
Pitch = -90°

Roll = 0° (see Note 1, below)

Grip = Closed (see Note 2, below)

Note 1: Because the hand can turn through many revolutions of "roll," it is difficult to tell by simply looking at the hand whether the "roll" has been set to 0°. Yet, it is important that the roll initially be 0° in order that the wrist cables be allowed their full range of motion.

Note 2: For experimental program it is important that the initial position be very precise. Initialization point is shown in Figure 13.



1 block is 1x1 inch

Initialization and Cartesian Paper

Figure 13

4.7 Coordinate Conversions

It is often advantageous to be able to describe the configuration of a robot arm in more than one coordinate system. The two most commonly used systems are:*

- Joint Coordinates (the joint angles of the arm). These are most convenient for controlling the arm directly from a computer.
- Cartesian Coordinates (X, Y, Z, pitch, and roll). These are more convenient for describing an assembly task on a flat table top.

For practical work, a set of formulas is needed to mathematically convert from one coordinate system to the other.

- The Forward Solution converts from joint angles to Cartesian coordinates.
- The 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 may be derived and implemented.

4.7.1 Kinematic model of arm

Before formulating 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 D-1. The kinematic model indicates how each joint is articulated, how the joint angles are measured, and the distances between joints.

Θ is used to indicate joint angles in mathematical expressions. The symbols Θ_1, Θ_2, Θ_3, Θ_4, and Θ_5, respectively, are proportional to the joint expressions J_1, J_2, J_3, J_4, and J_5 used in the computer command discussed in Chapter 7. The Θs, measured in degrees or radians, are related to the Js, measured in motor steps, as shown in Table D-1. There are 360 degrees or 2 radians in one complete revolution.

The distances bewteen joints (lengths of arm members) are indicated by the constants, H, L, and LL shown in Figure 14. H is the distance from the table top to the shoulder joint centerline; L is the distance from shoulder joint to elbow joint, which equals the distance from elbow joint to wrist joint; and LL is the distance from the wrist joint to the center point between the two fingertips, with the fingertips separated by 1.5 inches. Values for these distances are given in Table 7.

The pitch angle, P, and the roll angle, R, are given by the following equations.

$$P = .5\ (\Theta_5 + \Theta_4) \qquad (1)$$

$$R = .5\ (\Theta_5 - \Theta_4) \qquad (2)$$

where Θ_4 and Θ_5 are right and left wrist angles. The angles P, Θ_4, and Θ_5 are all measured from the horizontal as shown in Figure 15.

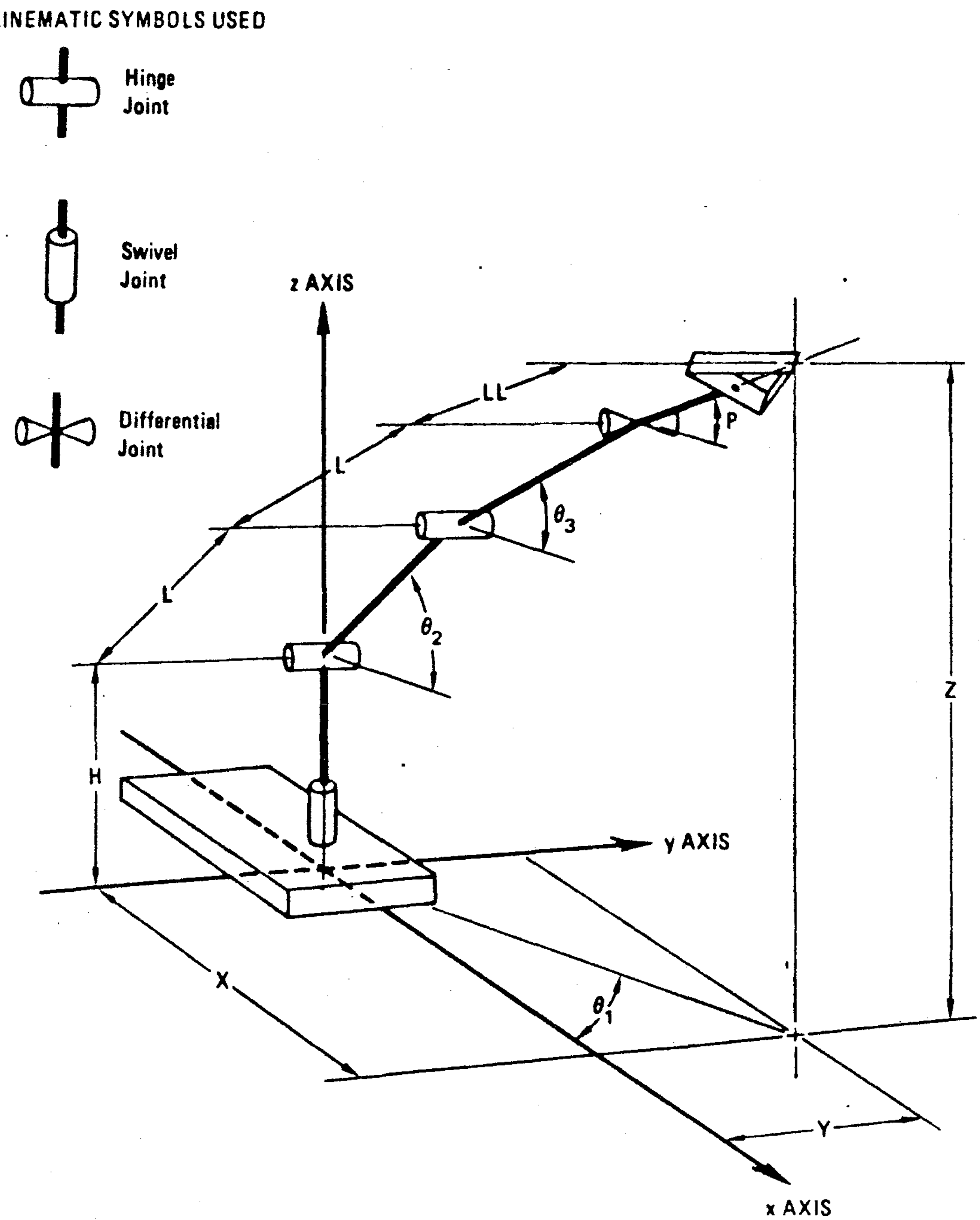
CONVERSION FACTORS BETWEEN MOTOR STEPS

AND REVOLUTE JOINT ANGLES

Motor	Joint	Steps in Revolution	Steps per Radian	Steps per Degree
1	Base	7072	1125	19.64
2	Shoulder	7072	1125	19.64
3	Elbow	4158	672	11.55
4	Right wrist	1536	241	4.27
5	Left wrist	1536	241	4.27

Conversion Factors Between Motor Steps
and Revolute Joint Angles

Table 6



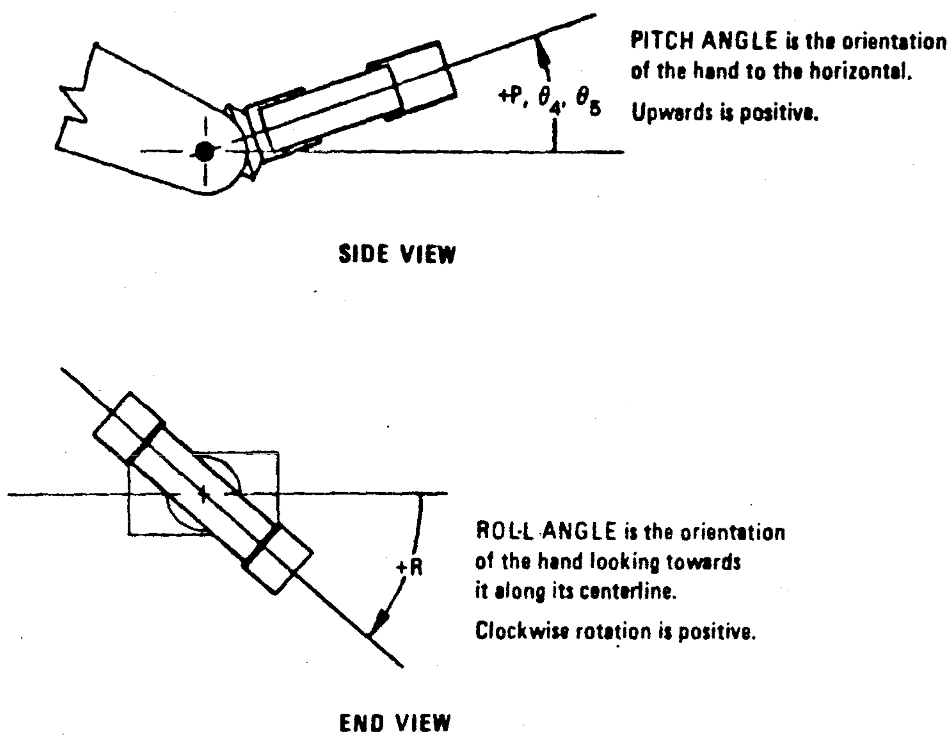
Kinematic Model of the TeachMover Arm

Figure 14.

Lengths of TeachMover Arm Members		
Segments	Length (inches)	Length (mm)
H	7.68	195.0
L	7.00	177.8
LL	3.80	96.5

Lengths of TeachMover Arm Members

Table 7



Definition of Roll and Pitch Angles

Figure 15.

4.7.2 Forward arm solution

This section shows how to determine P, R, and the X, Y, and Z coordinates of the end point from the joint angles Θ_1, Θ_2, Θ_3, Θ_4, and Θ_5. The coordinates and joint angles are defined in Figure 14. This solution relies on the trigonometric relationships given in Figure 16 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 from the base pivot to the end point. The situation is summarized in Figure 17. Summing the vertical contributions from each link gives the following expression for Z:

$$Z = H + L \sin \Theta_2 + L \sin \Theta_3 + LL \sin P \tag{3}$$

Summing the horizontal contributions gives:

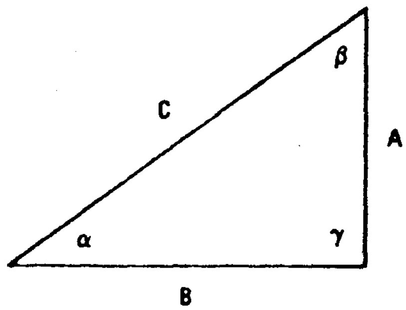
$$RR = L \cos \Theta_2 + L \cos \Theta_3 + LL \cos P, \tag{4}$$

where pitch angle P is given by

$$P = .5(\Theta_5 + \Theta_4). \tag{5}$$

The second step is to determine the X and Y coordinates of the end point from the intermediate variable, RR, as shown in Figure 18. By inspection, the coordinates are:

$$X = RR \cos \Theta_1 \tag{6}$$



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 $B = C \cos \alpha$

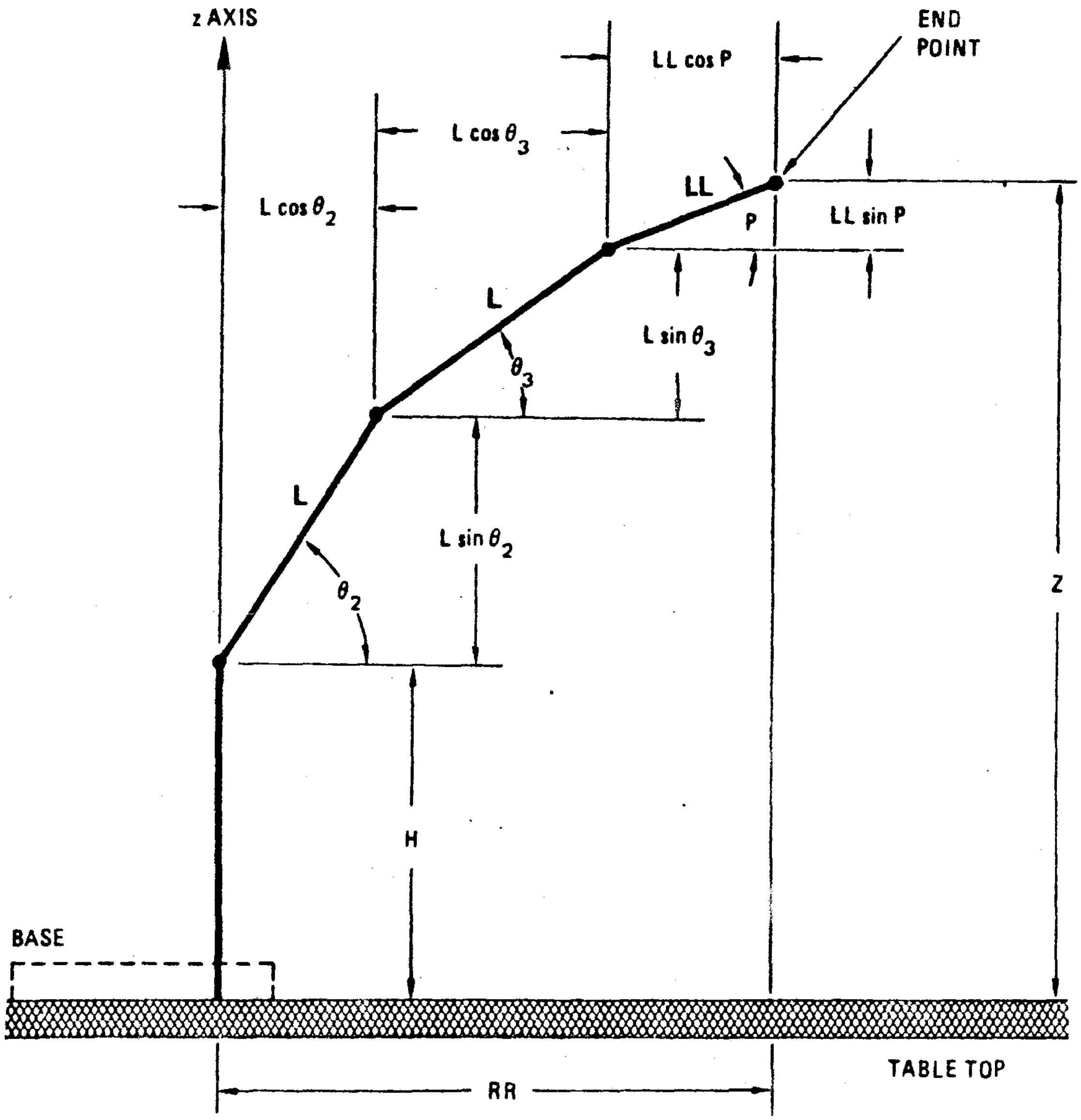
$\tan \alpha = \frac{A}{B}$ or $A = B \tan \alpha$

ANGLE DEFINED BY INVERSE FUNCTION:

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

Basic Trigonomatric Relationships

Figure 16.



Side View of Kinematic Model

Figure 17.

$$Y = RR \sin \Theta_1 \quad (7)$$

A summary of this forward solution is given in Table D-3. A BASIC program implementing this solution is given in Figure D-12 (Statements 460 to 510). The program's variables T_1, T_2, . . ., T_5 correspond to the angles Θ_1, Θ_2, . . ., Θ_5.

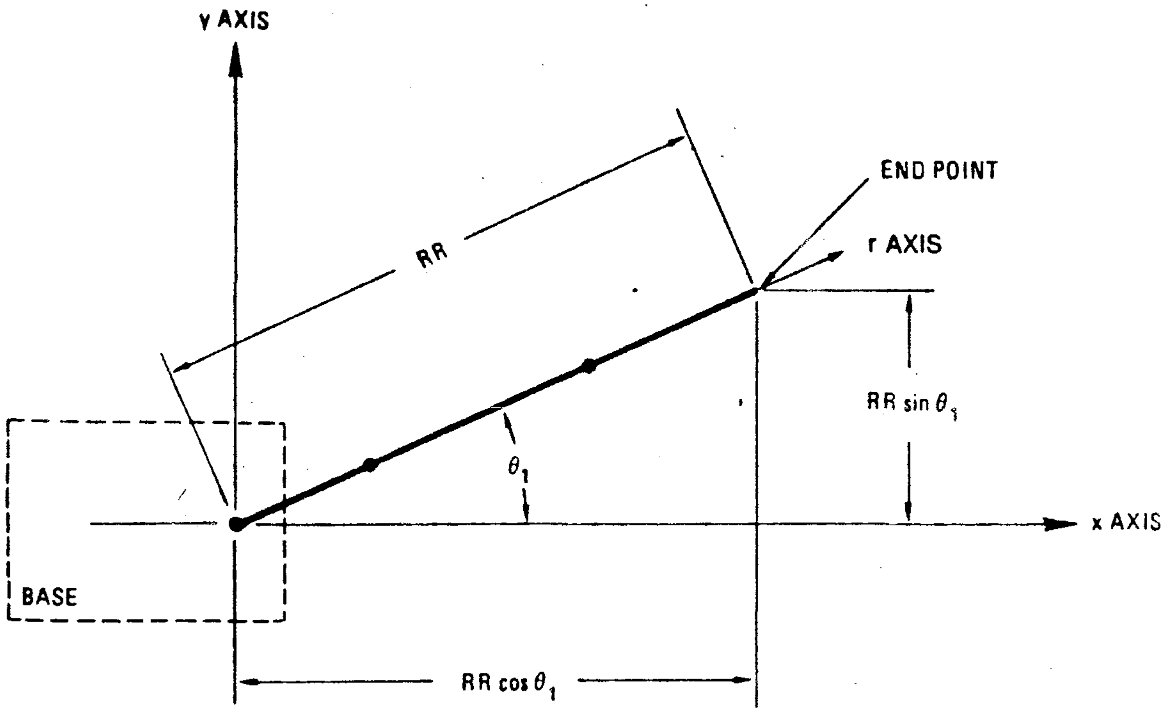
4.7.3 Backward arm solution

This section shows how to determine the joint angles Θ_1, Θ_2, Θ_3, Θ_4, and Θ_5 required to position the end point at a desired X, Y, Z position and with desired values of pitch and roll. The coordinates referred to are shown in Figure 14. A review of the formulas used is given in Figure 16.

A. 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:

- 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.



Top View of Kinematic Model

Figure 18.

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	$Z = H + L \sin \Theta_2 + L \sin \Theta_3 + LL \sin P$

Summary of Forward Solution

Table 8

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

The units of these distances (inches or millimeters) are 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 19):

P: The desired pitch angle, measured in degrees

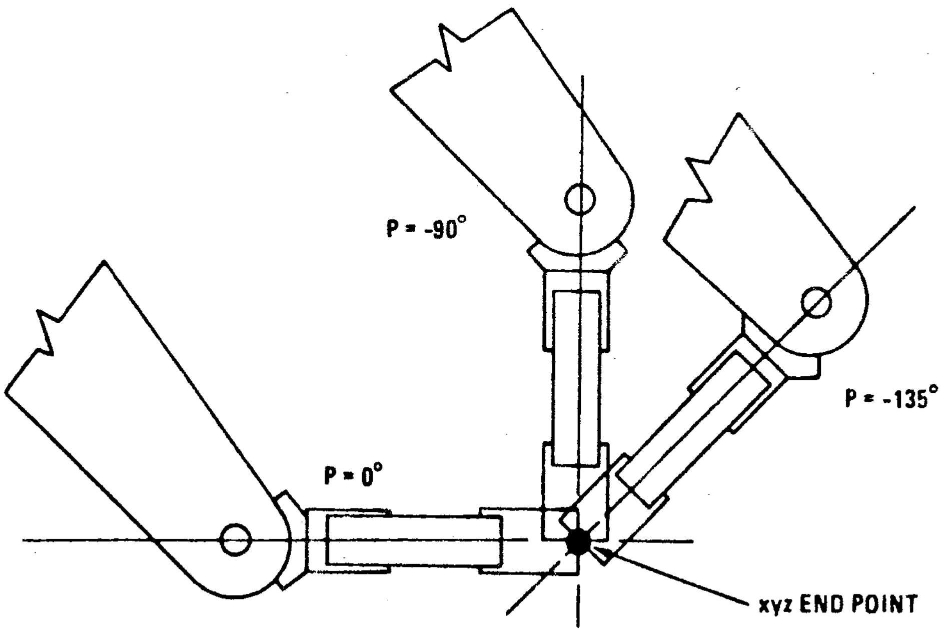
R: The desired roll angle, measured in degrees

In practice it is difficult to distinguish between positive and negative roll angles (as $+90^\circ$ and -90°, or $+45^\circ$ and -135°) by looking at the hand. It is helpful to mark the top of the hand when it is at 0° to eliminate this ambiguity. The 0° position corresponds to the orientation when the wrist cable turnbuckles are aligned.

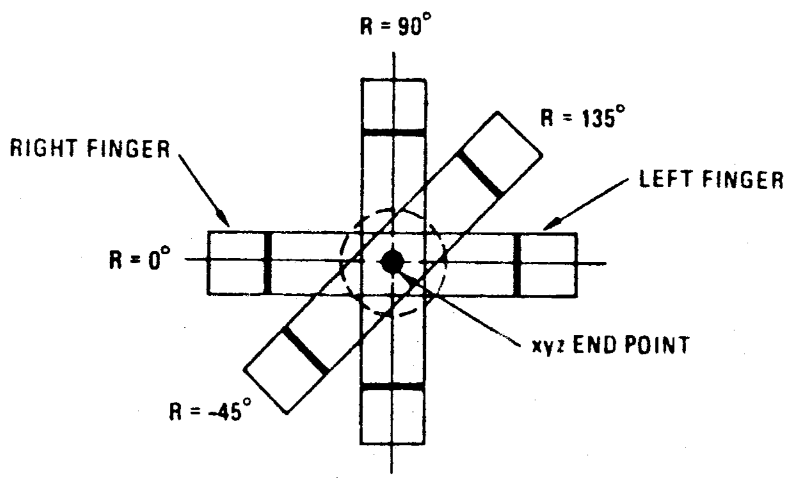
B. Specifying roll in cartesian frame, R'

Sometimes it is useful to express "roll" with respect to a Cartesian frame rather than with respect to the arm. One way to do this is to use $P = -90^\circ$ (hand point down) as a reference orientation, and measure the "Cartesian roll" with respect to the x-axis, as indicated in Figure 20. The formula relating to roll measured with respect to the arm (R) and the roll measured with respect to the Cartesian frame (R') is then simply:

$$R' = R - \Theta_1$$



(a) DIFFERENT PITCH ANGLES AT SAME ENDPOINT



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

Different Hand Orientations

Figure 19.

In the backward solution, we introduce a special variable, R1, that enables us to write equations that are valid regardless of whether roll is measured with respect to the arm or with respect to the Cartesian frame.

R1 = 1 if roll is with respect to Cartesian frame.

R1 = 0 if roll is with respect to arm frame.

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

$$R' = R - \Theta_1 R1 \qquad (9)$$

Solving for R gives:

$$R = R' + \Theta_1 R1 \qquad (10)$$

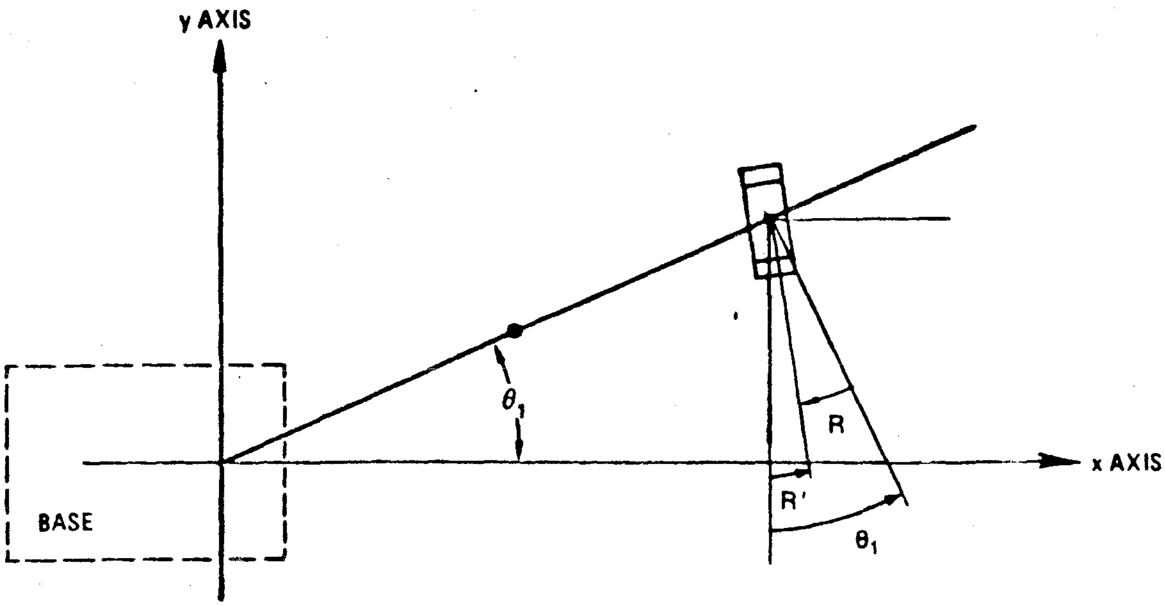
C. <u>Backward solution, step-by-step</u>

The first step of the backward solution is to determine the base angle, Θ_1, and the radius vector, RR, from the base to the end pont as shown in Figure 21. Using the Pythagorean Theorem:

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

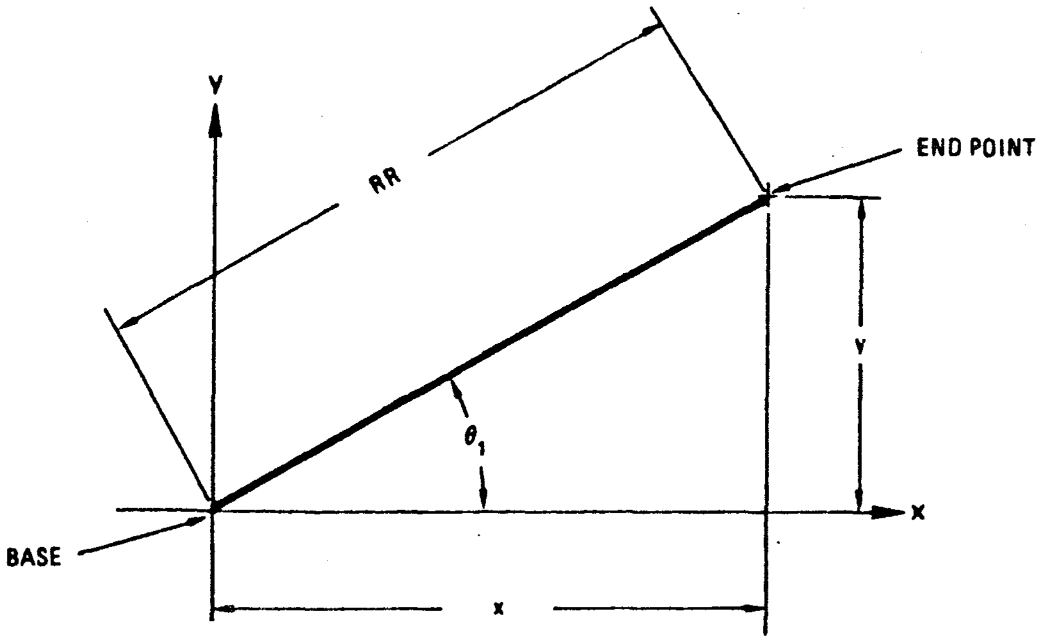
$$\Theta_1 = \tan^{-1}(Y/X). \qquad (12)$$

The second step is to find Θ_4 and Θ_5 from P and R. Using Equation (1) and Equation (2) of the wrist



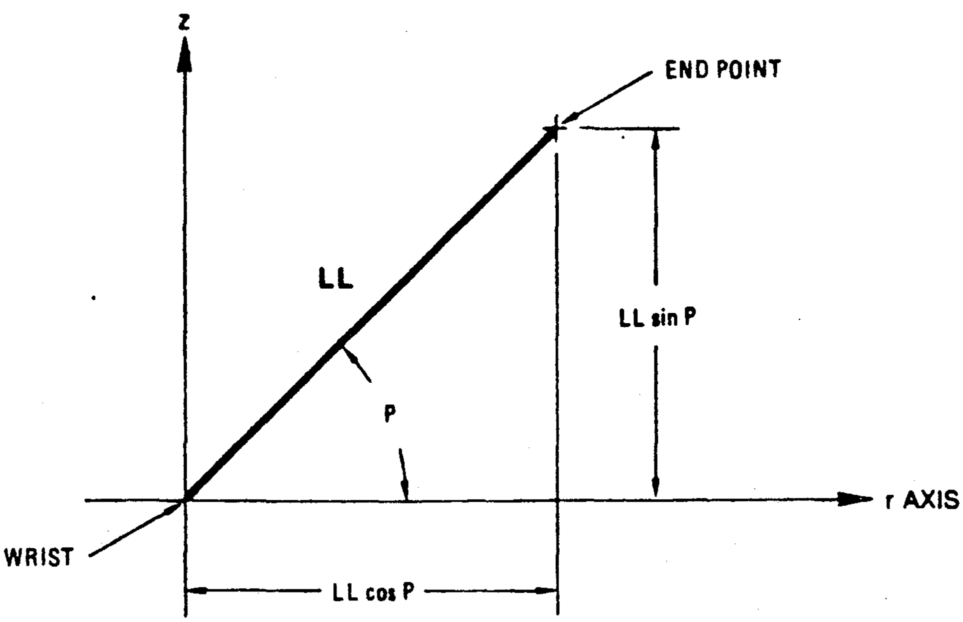
Top view of arm with Pitch = 90 degrees
showing roll in Cartesian frame (R') and
roll with respect to the arm (R).

Figure 20.



Top View of Arm

Figure 21.



Side View of Hand Triangle in Kinematic Model

Figure 22.

differential previously described, and substituting $(R' + {}_1R1)$ for R using Equation (10) gives:

$$_5 = P + R' + \Theta_1 R1 \qquad (13)$$

$$_4 = P - R' - \Theta_1 R1. \qquad (14)$$

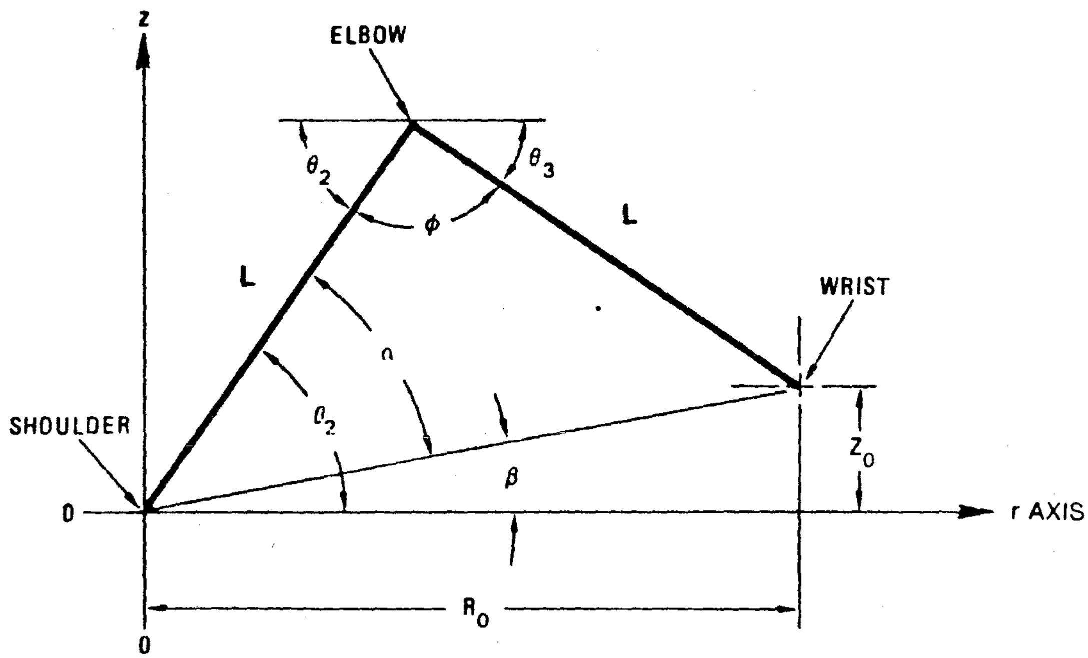
[Note: From here on, the prime will be dropped and use R for roll in all cases, remembering to set R1 = 0 when roll is measured with respect to the arm, and R1 = 1 when roll is measured with respect to the Cartesian frame.]

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 of the kinematic model shown in Figure 17. 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 D-9. From this triangle the coordinates of the wrist are:

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

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

The fourth step is to define the shoulder-elbow-wrist triangle so that Θ_2 and Θ_3 can be determined. For this purpose, the translated coordinate system introduced in Figure 23 is used. The origin (0,0) is at the shoulder and the coordinates of the wrist are now (R_0, Z_0).



Shoulder-Elbow-wrist Triangle

Figure 23.

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:

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

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,

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

substituting for Z_w using Equation (16) gives

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

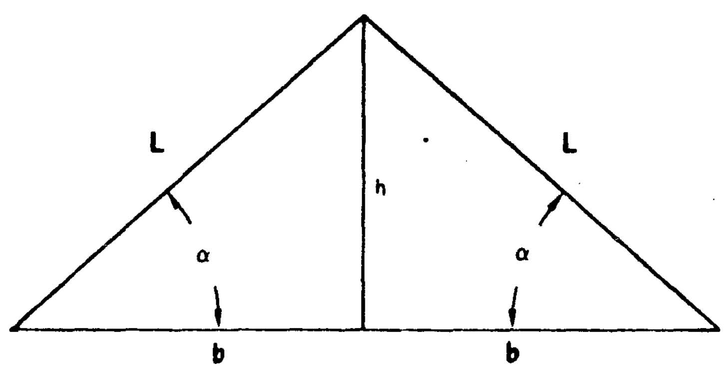
The fifth step is to solve the shoulder-elbow-wrist triangle for Θ_2 and Θ_3. Three new angles: α, β, and ϕ, are introduced to simplify this solution. First solve for α, β, and ϕ.

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

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

Pivoting the shoulder-elbow-wrist triangle about the shoulder by β gives the simplified triangle shown in Figure 24. The length of the base of the simplified triangle is given by $\sqrt{Z_0^2 + R_0^2}$ (Phythagorean Theorem, using the right triangle at the bottom of Figure 23). As shown in Figure 24, the simplified triangle can be partitioned into two congruent right triangles. The base, b, of each of these smaller triangles is then given by:

$$b = .5\sqrt{Z_0^2 + R_0^2} \tag{21}$$



Simplified Triangle

Figure 24.

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

$$h = L^2 - b^2 \quad . \qquad (22)$$

Since the tangent of α is h/b,

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

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

$$\alpha = \tan^{-1} \frac{\sqrt{L^2 - b^2}}{b} \qquad (24)$$

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

$$\alpha = \tan^{-1} \sqrt{\frac{4 \quad L^2}{R_o^2 + Z_o^2} -1.} \qquad (25)$$

The sixth step is to use and to determine Θ_2 and Θ_3. The following three relations are first set up and then solved. At the shoulder (see Figure D-10),

$$\Theta_2 + \Phi + \Theta_3 = 180^\circ . \qquad (27)$$

Summing the internal angles of the simplified triangle (Figure 14) gives $\Phi + \alpha + \alpha = 180$, or

$$\Phi = 180^\circ \alpha - 2 \quad . \qquad (28)$$

Substituting the value of Θ_2 from Equation (26) and the value of Φ from Equation (28) into Equation (27) gives

$$\Theta_3 = \alpha - \beta \quad . \qquad (29)$$

Note however, that the elbow angle, Θ_3, is defined as the angle above the horizontal and hence we must change the sign of Θ_3.

In summary, the results of the sixth step are:

$$\Theta_2 = \alpha + \beta \ . \tag{30}$$

$$\Theta_3 = \beta - \alpha \ . \tag{31}$$

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

4.7.4 Variation of hand length with hand opening.

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

$$S_6 = 371 \text{ steps/inch } (14.6 \text{ steps/mm}).$$

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 25. The effect is small, ±0.10 in. (±2.5 mm), but for more precise work it may be necessary to take this into account.

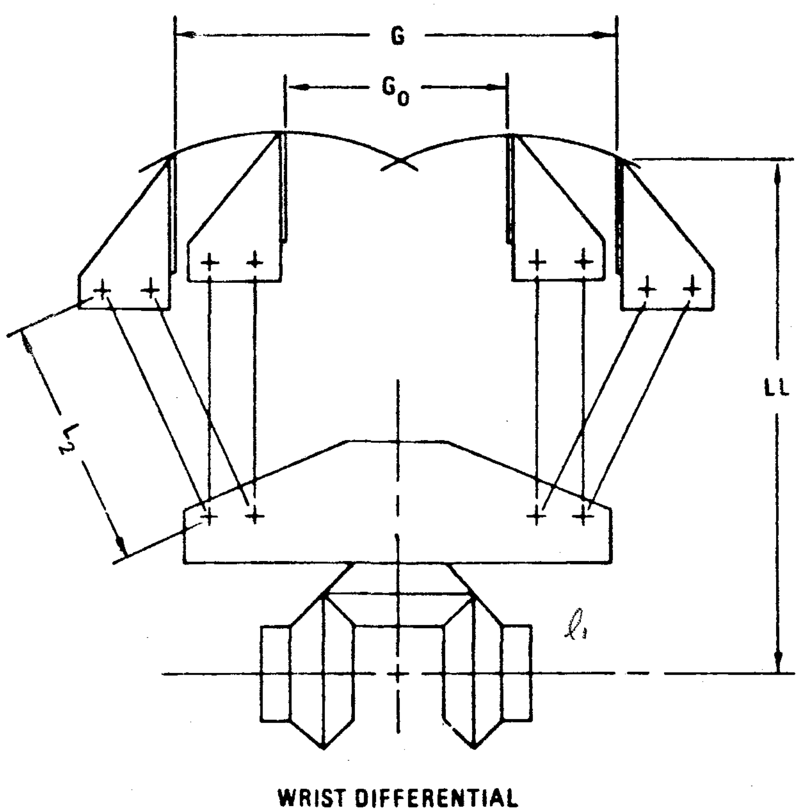
The hand length, LL, may be expressed as the sum of a fixed length, L_1, and a varying length that depends on hand opening, G, by the following formula:

$$LL = L_1 + \sqrt{L_2^{\,2} - \frac{(G - G_0)^2}{2}} \tag{32}$$

Summary of Backward Solution	
Step	Operation
1	Determine arm constants H, L, LL
2	Determine the desired X, Y, Z, R, P, and R1 coordinates of the endpoint
3	$_1 = \tan^{-1}(Y/X)$
4	$RR = X^2 + Y^2$
5	$_5 = P + R + R1 \quad _1$
6	$_4 = P - R - R1 \quad _1$
7	$R_0 = RR - LL \cos P$
8	$Z_0 = Z - LL \sin P - H$
9	$= \tan^{-1}(Z_0/R_0)$
10	$= \tan^{-1} \quad 4L^2/(R_0{}^2 + Z_0{}^2) - 1$
11	$_2 = \quad +$
12	$_3 = \quad -$

Summary of Backward Solution

Table 9



Variation of Hand Length with Hand Opening

Figure 25.

where:

L_1 = 1.884 in (47.9 mm)

L_2 = 1.700 in (43.2 mm)

G_0 = 1.520 in (38.6 mm)

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 computer from the hand opening using Equation (32).

4.8 Procedure for Robot Arm Movement

In order to grab the object correctly and place it where we want, those data that were determined and calculated in procedure for detection of object should be converted into cartesin coordinates for robot arm movement as below.

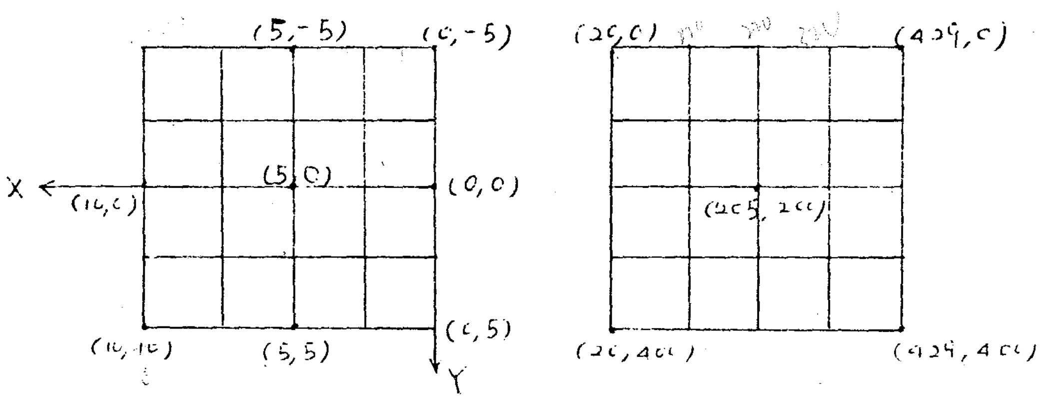
1. Conversion of pixels to cartesian world coordinate for Robot shown in Figure 26.

As compared in Figure 26, the coordinate for X,Y in cartesian paper can be calculated in the following way,

$$X = (430 - XC)/41$$

$$Y = (YC - 200)/41$$

where X and Y are in inch, and XC and YC are in pixels.



Cartesian Paper in Inch for Robot

X,Y coordinates in pixels for VPU

Conversion of pixels to Cartesian Coordinate

Figure 26

2. Save the data of the object into some rooms.

R = transmitted data for rotated degree of the object

GP = the width of the object (=HG)

XR = X coordinate of the center or centroid of the object

YR = Y coordinate of the center or centroid of the object

ZR = height from the table top to the tip of the hand grip

P = pitch in degree

3. Initialize the robot arm using cartesian coordinates. To initialize the robot arm, use the Hand Held Control to place the tip of the hand grip.

XR = 5 inch

YR = 0 inch

ZR = 0

P = -90 degrees

R = 0 degree

GP = 0

Then, calculate the real cartesian coordinates for robot arm. Since the table top is 8 inches higher than TeachMover base and the center of the table top (5,0) is 4 inches farther placed from the real cartesian coordinate for TeachMover:

X = XR + 4 = 9

Y = YR = 0

Z = ZR + 8 = 8

P = -90 degrees

R = 0

GP = 0

This conversion can be easily understood in Figure 27.

4. Move the arm to the stand-by position to avoid the blocking of lens' sight for the 10-in. x 10-in. table top cartesian paper.

XR = 2; X = XR + 4 = 6

YR = 0; Y = YR = 0

ZR = 9; Z = ZR + 8 = 17

P = 45 degrees

R = 0

GP = 0

5. Then, obtain data of the position of the object.

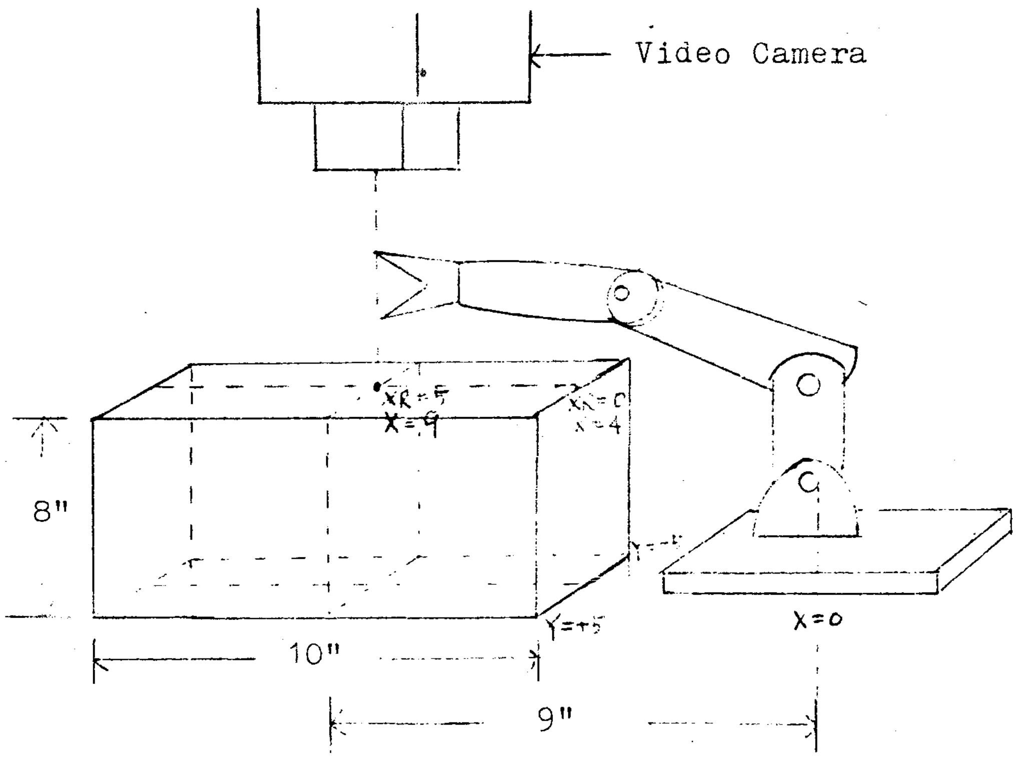
6. Move the arm to the 1-in. higher than the top of that object and ready to be picked up by opening the hand and rotate in proper degrees if it is cubic.

7. Lower the hand to have the object between the hand grips.

8. Hold the object.

9. Lift up the object by one inch.

10. Place it as programmed. If it is cubic, XR = 2, YR = 3, ZR = 0, P = -90 degrees, R = 0 and GP = 0. If



The conversion from the cartesian on the table top to the real cartesian coordinates for robot arm

Figure 27

it is cylinder, XR = 2, YR = -3, ZR = 1, P = -90 degrees, R = 0 and GP = 0.

11. Go back to the origin, XR = 5, YR = 0, ZR = 0, P = -90 degrees, R = 0, GP = 0.

12. Stop the program.

Chapter 5

APPLE IIE COMMUNICATION TO VPU AND TEACHMOVER

5.1 Introduction

For the Apple Computer to communicate with the TeachMover and VPU it is necessary to interconnect the two via a serial card and a cable.

5.2 Hardware Needed

1. Super Serial Card, manufactured by Apple Computer, Inc.

2. RS-232-C cable with a 25-pin male connector at each end.

5.3 Hardware Setup for VPU

The Apple II Super Serial Card (SSC) needs to be prepared first in accordance with Chapter 1 of the SSC manual. Then proceed with the settings below. For reference, the TeachMover uses the SSC in the Communications Mode.

1. The reversible jumper block on the SSC should have the white triangle pointed at "MODEM."

2. Switch No. 1 (SW1) Settings

POSITION	SETTING	NOTES
1	Off	These settings are 2400 BAUD
2	On	transmission rate
3	Off	
4	On	
5	On	
6	On	Communications mode RS-232-C
7	On	signals

3. Switch No. 2 (SW2) Settings

POSITION	SETTING	NOTES
1	On	Sets 1 stop bit
2	Off	Sets 7 data bits
3	Off	
4	Off	Even parity
5	Off	No LF after CR
6	Off	Interrupts off
7	Off	RS-232-C signals

4. The SSC was plugged into slot #2 in the Apple.

5.4 Hardware Setup for TeachMover

The Apple II Super Serial Card (SSC) needs to be prepared first in accordance with Chapter 1 of the SSC manual. Then proceed with the settings below. For reference, the TeachMover uses the SSC in the Communications Mode.

1. The reversible jumper block on the SSC.

2. Switch No. 1 (SW1) Settings

POSITION	SETTING	NOTES
1	Off	These seetings are 9600 BAUD
2	Off	transmission rate, and match
3	Off	the factory setting of the
4	On	TeachMover.
5	On	
6	On	Communications mode
7	On	RS-232-C signals

3. Switch No. 2 (SW2) Settings

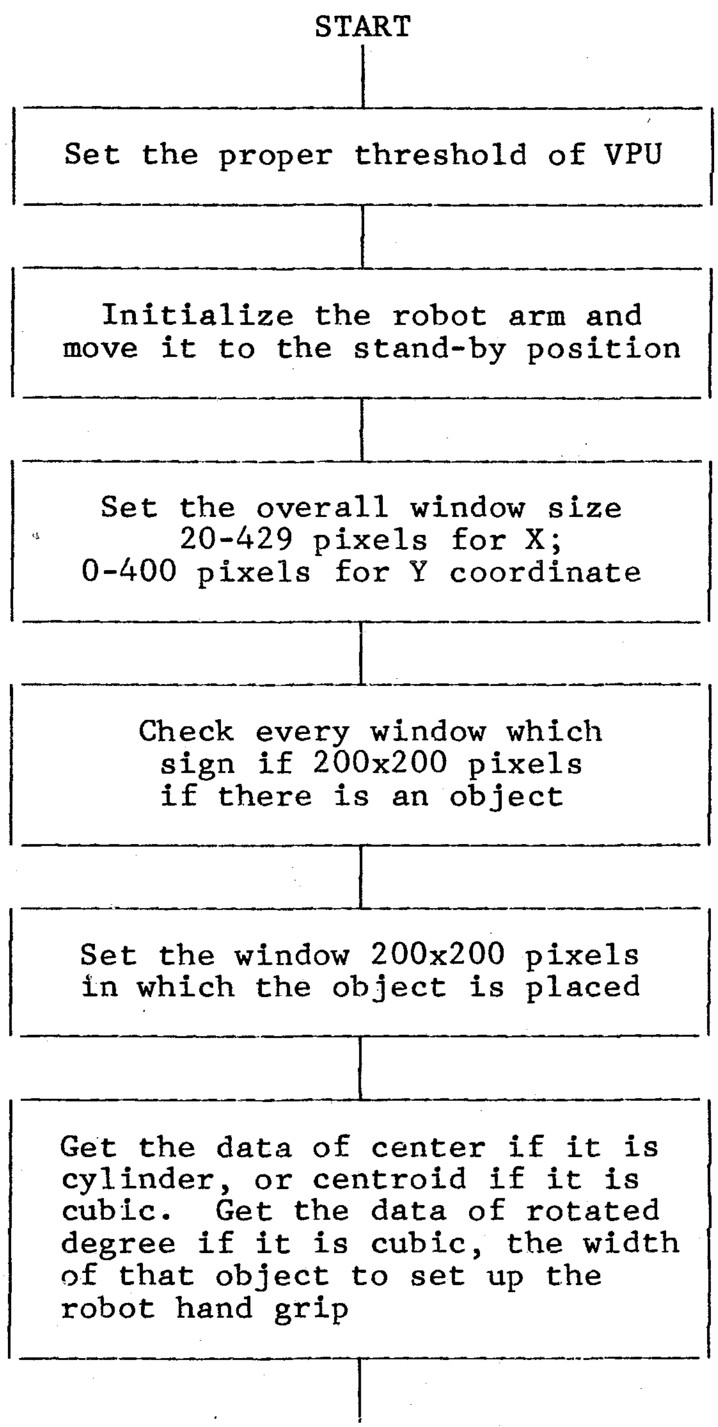
POSITION	SETTING	NOTES
1	On	Sets 1 stop bit
2	Off	Sets 7 data bits
3	On	
4	Off	Odd parity
5	Off	No LF after CR
6	Off	Interrupts off
7	Off	RS-232-C signals

4. The SSC was plugged into slot 4 in the Apple.

5. RS-232-C cable was plugged into the right connector when looking from the rear of the TeachMover.

Chapter 6

FLOW DIAGRAM FOR ROBOTIC SYSTEM WITH COMPUTER VISION



Transfer and calculate the data for robot-arm movement

↓

Move the robot arm 1 inch higher, place the top of the object with 1 inch wider hand grip than the real width of the object and with required rotation and -90 degrees pitch

↓

Lower the hand and hold the object

↓

Place it to the desired place.
If it is cubic X=6, Y=3
If it is cylinder X=6, Y=-3
in the experimental program

↓

Move the robot arm to the center of the table top cartesian paper at X=5, Y=0, Z=0

↓

STOP

Details and algorithms are explained in 3.3 including Procedure for Detection of Object, and in 4.8, the Procedure for Robot Arm Movement. The experimental program is listed in Appendix E.

Chapter 7

CONCLUSION

Image processing was found to be useful in detecting an object. Analyses were made to get the proper information for robot movement using a controller such as the Apple IIe computer. Even though the usage of personal computers was limited in the past, it was found to be useful as a robotic system controller.

The system was found to be capable of distinguishing cubic and cylindrical objects of different sizes. Programs written were capable of retriving these objects and placing them in the proper location.

As a future project, a sensor for object detection can be attached to the robot itself to simplify coordinate transformations. Filtering techniques can be applied to images of the object using video image processing to improve the quality of the images when operating at poor lighting environments.

REFERENCES

1. R. Y. Wong, "Scene Matching with Invariant Movement," Computer Graphics Image Processing, No. 8, August 1978.

2. R. Y. Wong, "Intensity Signal Processing of Images for Optical to Radar Scene Matching," IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-28, No. 2, April 1980.

3. R. Y. Wong, "Computer Pattern Classification and Scene Matching," published by the Department of Electrical and Computer Engineering, California State University, Northridge.

4. "Operating of the Five-Axis Robot Model TCM," by MICROBOT, Inc.

5. "Video Processing Unit Operating Manual" by Rank Videometrix, January 1982.

6. "Apple II Super Serial Card Installation and Operating Manual" by Apple Computer.

APPENDIX A

System Set-Up

APPENDIX A

System Set-up

Power Supply (for the video camera)

ON/OFF switch

This switch turns the camera system power on and off.

Video Camera

1. Camera Connector

   Connect the camera cable connector. Be sure the cable connector if fitted securely to each Vertical, Horizontal, and Video connector which should be properly connected to VPU input. And, the monitor is to be connected to the monitor input.

2. For proper size and picture

   Adjust Lens focus for the sharpest picture. Adjust the aperature for the proper threshold of the VPU. Adjust the distance from the 8-inch high table top to have the entire picture within the 409x400 pixel window of the VPU.

Videometrix

1. Connector coming from camera should be properly connected to camera as explained above in Video Camera section.

2. The connector for monitor to TV monitor

<u>Serial Port</u> (for VPU)

1. 21-pin connector goes to the serial port which resides in slot #2 of the Apple Computer.
2. The switch on the first layer inside the VPU is set up as below. Switch 6 is OFF, 7 is ON, and 8 is ON for 2400 baud rate. Switch 4 is ON for CR/LF and 5 is OFF for standard I/O. The rest of the switch left does not affect any situation.

<u>Robot Arm</u>

1. DC Power Supply should be connected to the robot arm, TeachMover using the two leg connector which red leg should be plugged into red female plug.
2. The DB-24S coming from the Apple IIe connector should be connected to the right female connector attached to the back of the base.
3. The center of the robot-arm should be placed at X=0 and Y=0 point in real cartesian world coordinates.

<u>Apple IIe</u>

1. Super Serial Port for VPU should reside in slot #2 for experimental program.
2. Super Serial Point for TeachMover is in slot #4.
3. Disk Drive has ribbon cable connected to its controller card in slot #6, drive 1 of the Apple IIe computer.

APPENDIX B

System Operating Instruction

APPENDIX B

System Operating Instruction

1. Turn on power supply for the video camera.
2. Turn on the lighting system.
3. Turn on the monitor.
4. Turn on the VPU power switch.
5. Turn on computer power and load "project" program from disk drive 2.
6. Run the program, following the instructions below.

   6.1 Initialize the robot arm with X=5, Y=0, Z=0, P=90 degrees, R=0, GP=0 at the center of the table top cartesian paper, which is actually X=9, Y=0, Z=8, P=90 degrees, R=0, GP=0 in real cartesian coordinates for robot calculation and movement.

   6.2 Set the proper threshold.

   6.3 Return to get the data for the object location and move the robot arm to place the object where we want. If it is cubic, place it at X=6, Y=0 and if it is cylinder, place it at X=6, Y=0 in real cartesian coordinates.

7. Check the movement and the result.

APPENDIX C

Functional Difference of VPU Software

APPENDIX C

Functional Difference of VPU Software

1. Required Changes in Host Software

   A. The following commands no longer return output

CS)	
CM)	
GS)	All Gate & Crosshair
GM)	Manipulation commands
WS)	
WM)	
AG)	

TH	Threshold Set
DV	Digitized Video Enable/Disable
CI	Comparator Input Select
RS	Reset
HO	Horizontal Offset
DM	Display Mode
US	Update Aspect Ratio
WP	Write to a Port

   B. Ensure that VPU switch settings are correct for application (settings now control baud rate, daisy/non-daisy, and VPU output terminator characters.

   C. Spaces may no longer be used as delimiters in polled VPU command strings.

2. Enhancements

   A. New Commands

AN	Vertex & Cosine of an angle
RP	Read a parallel port
WP	Write to a parallel port
EL	Equation of a least squares fit line
MF	Median filter enable/disable
CT	Crosshair tracking enable/disable
FN	Surface or edge focus

B. Fixed or Enhanced Commands

RS	Reset now works
CS/W	Set Xhairs to center of current window
CC	Can now use many points (up to 10,000 can be specified)
AG	Now works reliably
WS	Now works reliably
CS	Now center crosshairs to precise center of maximum window

3. Power-Up Conditions

Parameter	Value
Comparator Input	Video
Digitization	Off
Video Threshold	50% (Relative)
Gradient Threshold	50% (Absolute)
Window Setting	Full window
Crosshair Setting	Centered
Aspect Ratio	1.25 (5X:4Y)

APPENDIX D

Serial Interface Commands

APPENDIX D

Serial Interface Commands

Notes:

1. <CR> = Carriage Return
2. Arm returns [0<CR>] if command has a syntax error, [1<CR>] after command is executed (except for @ RUN), [2<CR>] if STOP button was pressed before execution was completed (@STEP and @CLOSE only)

Command	Function	Syntax/Details of Operation
@ARM	Specifies recognition character to use instead of "@" sign	@ARM <CHAR> <CR> where CHAR is any character except a carriage return.
@CLOSE	Close gripper until grip switch is activated	@CLOSE <SP> <CR> where SP = optional speed value (see item 4 in this appendix).
@DELAY	Inserts a delay between transmitted characters	<N> @DELAY <CR> Where N = proper delay value, determined by trial and error.
@QDUMP	Uploads entire current program from TeachMover to host computer	@DUMP <CR> Returns character string comprising 8 two-byte values for each program step. See Table 9 in chapter 7 for details.
@QWRITE	Downloads a program step from host computer to TeachMover	@WRITE<N>,<L1>,<L2>,... <L7><CR> where N = Step number to which program step is to be written. L1-L7 = two-byte values as in @QDUMP command. See chapter 7 for details.

Command	Function	Syntax/Details of Operation
@READ	Reads value of the internal position registers, gives last key pressed on teach control, and tells which input bits are on.	@READ <CR> Arm returns: <K1>,<K2>, ..., <K6>,<I><CR> where K1-K6 = values of internal position registers I = Last key *256 + Input Byte where "Last key" values are defined below:

"Last key" Value	Key Pressed
1	TRAIN
2	PAUSE
3	GRIP
4	OUT
5	FREE
6	MOVE
7	MODE
8	STEP
9	POINT
10	JUMP
11	CLEAR
12	ZERO
13	SPEED
14	REC

and:
"Input Byte" = decimal number whose binary equivalent specifies which of the eight input bits are set to 1 (see "jump condition" numbers under hand-held teach control JUMP command, above).

Command	Function	Syntax/Details of Operation
@RESET	Zeros the internal position registers and turns off motor currents	@RESET<CR>
@SET	Sets subsequent arm speed and activates joint control keys on hand-held teach control	@SET<SP><CR> where SP = optional speed value (see item 4 in this appendix). Control returns to host when REC or MODE key is pressed.

@STEP	Sets arm speed, moves joints, sets output bits	@STEP<SP>,<J1>,<J2>,...,<J6>, <OUT><CR> where SP = speed value (see item 4 in this appendix). J1-J6 = Number of motor half-steps J1 = Base swivel (positive counterclockwise) J2 = Shoulder (positive downwards) J3 = Elbow (positive downwards) J4 = Right wrist (positive downwards) J5 = Left wrist (positive downwards) J6 = Hand (positive open) OUT = Optional decimal number whose binary equivalent specified the value of the output bits (see Appendix F, item G).

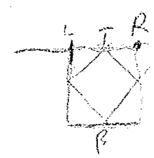
APPENDIX E

Program Listing

```
10  HOME
20 D$ =  CHR$ (4)
40  REM
50  GOSUB 7000
55  PRINT "RS"
60 E = 0
70 T = 0:B = 400:L = 20:R = 429
80  GOSUB 6000
90  PRINT "DV/E"
94  PRINT "CT/E"
95  PRINT "CS/W"
96  PRINT "TH/20"
100  GOSUB 8000
101  GOTO 20000
102  REM  FROM 27082
103  PRINT "***************************"
104  PRINT "  INITIALIZATION "
106  PRINT "---   (5,0) AT CENTER --- (0,0) AT RIGHTMOST CENTER---- (10,0)
      AT LEFTMOST CENTER ---    "
108  GOTO 160
110  PRINT "TYPE IN COMMAND AND, 0 OR, 1"
120  INPUT C$,E
130  GOSUB 7000
140  GOSUB 10000
142  GOSUB 8000
145  PRINT "WANT MORE COMMAND? Y-0,N-1"
146  INPUT ZV
147  IF ZV = 0 GOTO 110
160  PRINT "TYPE IN THRESHOLD VALUE, JUST NUMBER"
170  INPUT TN
175  GOSUB 7000
180  PRINT "TH/";TN
185  GOSUB 8000
190  PRINT " AGAIN? Y-0,N-1"
200  INPUT ZV
210  IF ZV = 0 THEN  GOTO 160
220  GOSUB 7000
230 TD = 0:BD = 200:LD = 20:RD = 223
240 M = (BD - TD) / 5 + 1
245 M =  INT (M)
250  DIM XL(M),XT(M),Y(M)
260  DIM DF(M),MT(M),NL(M)
270  DIM PS(3,3),PE(3,3),PD(3,3),NS(3,3)
300  PRINT "CN/B"
310  INPUT XC,YC,RC
320  PRINT "CN/B"
330  INPUT XB,YB,RB
340  IF  ABS (XB - XC) > 50 OR  ABS (YC - YB) > 50 THEN  PRINT "** FAILED
      **": GOTO 3220
350  PRINT "CN/B"
360  INPUT XC,YC,RC
370  IF XC < 80 THEN L = 20: GOTO 380
375 L =  INT (XC - 60)
380  IF XC > 369 THEN R = 429: GOTO 400
390 R =  INT (XC + 60)
400  IF YC < 60 THEN T = 0: GOTO 420
410 T =  INT (YC - 60)
420  IF YC > 340 THEN B = 400: GOTO 440
430 B =  INT (YC + 60)
440 M =  INT ((B - T) / 5 + 1)
450  GOTO 5060
```

```
3220  FOR NJ = 1 TO 3
3230  FOR NI = 1 TO 3
3240 T = TD + (NJ - 1) * 100
3250 B = BD + (NJ - 1) * 100
3260 L = LD + (NI - 1) * 103
3270 R = RD + (NI - 1) * 103
3280  PRINT "AG/":T:"/":B;"/":L:"/";R
3340 J = 0
3350  FOR I = T TO B STEP 5
3360 J = J + 1
3370  PRINT "CS/H/";I  ——— cross hair set
3380  PRINT "IE/X/L"  ——— intersect edge
3390  INPUT XL(J),Y(J)
3400  PRINT "IE/X/T"
3410  INPUT XT(J),Y(J)
3420  NEXT I
3430  FOR J = 1 TO M
3450  PRINT "AT Y=";Y(J);" XL=";XL(J);" XT=":XT(J)
3457 DF(J) = XT(J) - XL(J)
3458 NT(J) = XT(J) - XT(J - 1)
3459 NL(J) = XL(J) - XL(J - 1)
3460  PRINT "DIFFERENCE XT-XL=":DF(J)
3470  REM  (XT(J) - XL(J)) ) 5 AND          (XT(J) - XL(J)) ( 110 THEN
      P              P S(NI,NJ) = PS(NI,NJ) + 1
3472  IF DF(J) ) 5 AND DF(J) ( 110 AND  ABS (NL(J)) ( 18 AND  ABS (NT(J))
      ( 18 THEN PS(NI,NJ) = PS(NI,NJ) + 1
3476  IF XL(J) ) 0 AND XL(J) ( 503 THEN NS(NI,NJ) = NS(NI,NJ) + 1
3480  IF DF(J) ) 110 THEN PE(NI,NJ) = PE(NI,NJ) + 1
3485  IF  ABS (NT(J)) ) 18 OR  ABS (NL(J)) ) 18 THEN PD(NI,NJ) = PD(NI,NJ)
     ) + 1
3490  REM   TOO BIG
3500  NEXT J
3510  NEXT NI
3520  NEXT NJ
3528  PRINT "-----------------------"
3529  PRINT "  POSSIBLE :::::  TOO BIG"          print result
3530  FOR J = 1 TO 3
3540  FOR I = 1 TO 3
3550  PRINT " PS(";I:",";J;")=";PS(I,J);"   PE(":I:",":J;")=":PE(I,J)
3552  PRINT ' PD(";I:",":J:")=":PD(I,J)
3553  PRINT "NS(";I;",";J;")=";NS(I,J)
3560  NEXT I
3570  NEXT J
4000  REM  FIND MAXIMUM PS(I,J)
4010  REM  IF PS(I,J) IS MAXIMUM THEN SAVE I,J
4020 EX = 0
4022 DX = 0
4030 FX = 0
4040  FOR J = 1 TO 3
4050  FOR I = 1 TO 3
4060  IF NS(I,J) ( 25 AND PS(I,J) ) EX THEN  GOTO 4064
4062  GOTO 4070
4064  IF PE(I,J) ( 3 AND PS(I,J) ( 21 THEN EX = PS(I,J):IX = I:JX = J
4070  IF PE(I,J) ) FX THEN FX = PE(I,J):MX = I:NX = J
4080  NEXT I
4090  NEXT J
4100  REM  MINIMUM=30, 30/5=6. LEAST NO.
4110  IF EX ) 4 THEN  GOTO 5000
4120  REM  5000=FIND CENTER
4130  REM  IF IT IS TOO BIG
4140  PRINT "PLACE OBJECT, IF THERE IS AN OBJECT THEN IT IS TOO SMALL TO
     PICK UP"
4150  PRINT "AFTER PLACE OBJECT, PUSH RETURN"
4160  INPUT SS$
4170  GOTO 100 REM  3200 STARTING POINT
5000  REM  OBJECT CENTER AND RADIUS
```

```
5010 T = TD + (JX - 1) * 100
5020 B = BD + (JX - 1) * 100
5030 L = LD + (IX - 1) * 103
5040 R = RD + (IX - 1) * 103
5050  REM  OBJECT IS IN THIS BLOCK
5060  REM  FROM 450
5062  PRINT "AG/";T;"/";B;"/";L;"/";R
5064  PRINT "TH/":TN
5065  PRINT "TH/";TN
5070  REM  WINDOW EXPANSION
5110  PRINT "****(1)USING COMMAND-CC ****     "
5111 KF = 0
5112  PRINT "TH/":TN
5120  PRINT "CC"
5122  INPUT WX.WY.RC
5123  PRINT "CN/B"
5124  INPUT XC.YC
5126  IF  ABS (XC - WX) ( 10 OR  ABS (YC - WY) ( 10 THEN  GOTO 5132
5127 KF = KF + 1
5128  IF KF ) 5 THEN  PRINT "*** TRY AGAIN WITH DIFFERENT TH ***": GOTO 2
     9000
5129  GOTO 5112
5132  GOSUB 8000          close port#2
5140  PRINT "---------"
5150  PRINT "THE OBJECT IS PLACED AT X=":XC:" Y=":YC;" WITH RADIUS =":RC
5160  PRINT "----------"
5190  PRINT "****(2) USING MI, MX ****"
5191  GOSUB 7000          open port#2
5192  PRINT "TH/":TN
5200  PRINT "MI/X/L"
5210  INPUT LX.LY
5220  PRINT "MX/X/T"
5230  INPUT RX.RY
5240  PRINT "MI/Y/L"
5250  INPUT TX.TY
5260  PRINT "MX/Y/T"
5270  INPUT BX.BY
5275  GOSUB 8000          close
5280  PRINT "----------"
5290  PRINT "LEFTMOST POINT=(":LX:",":LY:")"
5300  PRINT "RIGHTMOST POINT= (";RX:",":RY;")"
5310  PRINT " TOP  POINT=(":TX:",":TY:")"
5320  PRINT " BOTTOM POINT=(";BX;",":BY;")"
5321 YQ =  ABS (RY - LY):XQ =  ABS (BX - TX)
5323  PRINT "RY-LY=":YQ;" :  BX-TX=":XQ
5330  PRINT "    "
5340 X1 = (RX - LX) / 2 + LX
5350 Y1 = (RY - LY) / 2 + LY
5360 X2 = (BX - TX) / 2 + TX
5370 Y2 = (BY - TY) / 2 + TY
5372  PRINT "----------------------"
5380  PRINT "CENTER BETWEEN LEFT AND RIGHT = (":X1:",":Y1:")"
5390  PRINT "CENTER BETWEEN TOP AND BOTTOM = (";X2;",":Y2:")"
5392  PRINT "CENTER XC,YC WAS= (";XC:",":YC:")"
5400  REM  IF (SC.YC) AND (X1,Y1)
5410  REM  AND(X2.Y2) ARE 100 DIFFERENT
5420  REM  THEN TRY--IE/X/L--IE----
5500 PS(IX.JX) = 0
5510 PT = 0:XP = 0:YP = 0:MT = 0:XM = 0:YM = 0
5514  GOSUB 7000
5516  PRINT "TH/":TN
5517  PRINT "TH/":TN
5528 J = 0
5529  FOR I = T TO B STEP 5
5530 J = J + 1
5531  PRINT "CS/H/":I
```



YQ
XQ

```
5532  PRINT "IE/X/L"
5533  INPUT XL(J),Y(J)
5534  PRINT "IE/XT"
5535  INPUT XT(J),Y(J)
5536  NEXT 1
5537  FOR J = 1 TO M
5538 DF(J) = XT(J) - XL(J)
5539 NT(J) = XT(J) - XT(J - 1):NL(J) = XL(J) - XL(J - 1)
5540  IF DF(J) > 5 AND DF(J) < 120 AND  ABS (NL(J)) < 18 AND  ABS (NT(J))
      < 18 THEN PS(IX,JX) = PS(IX,JX) + 1: GOTO 5700
5550  GOTO 5750 REM  GARBAGE GOTO NEXT
5700  IF NL(J) < 0 THEN PT = PT + 1:XP = XP + NL(J):YP = YP + 5: GOTO 573
     0
5710 MT = MT + 1:XM = XM + NL(J):YM = YM + 5
5730  REM  FROM 5700
5750  NEXT J
5755  GOSUB 8000
5760  REM  IF CUBIC IS IN 90 DEGREE
5762  IF PT < 3 AND MT < 3 THEN DG = 0:LL = PS(IX,JX) * 5 + 5: GOTO 5830
5764  PRINT "*** TEST PT=";PT;" MT               =";MT;"  XM=";XM;"  YM=";YM
     :"  DG=":DG
5800  IF PT > MT THEN DG =  ATN ( - YP / XP):LL = (YP + 15) /  SIN (DG): GOTO
     5830
5806  IF XM < 2 THEN DG = 0:LL = PS(IX,JX) * 5 + 5: GOTO 5830
5810 DG =  -  ATN (YM / XM):LL = (YM + 5) /  SIN ( - DG)
5820  REM  FROM 5800
5840  PRINT "--------ANGLE----WIDTH---"
5850 DR = 180 * (DG / 3.14159)
5860  IF LL < 43 THEN HG = LL / 43: GOTO 5880
5870 HG = (LL - 43) / 54 + 1
5880  REM  FROM 5860
5881  REM  IF CIRCLE
5882  IF  ABS (DR) < 41 OR  ABS (DR) > 60 THEN  GOTO 5920 REM  CUBIC  NOT
      CYLINDER
5883  GOSUB 7000: PRINT "TH/";TN
5884 YC =  INT (YC)
5885  PRINT "CS/H/":YC
5886  PRINT "IE/X/L": INPUT L1,T1
5887  PRINT "IE/X/T": INPUT L2,T2
5888 R1 = XC - L1:R2 = L2 - XC
5890 LC = (XC - L1) / 2 + L1
5891 LC =  INT (LC)
5892  PRINT "CS/V/":LC
5894  PRINT "IE/Y/L": INPUT L3,T3
5896  PRINT "IE/Y/T": INPUT L4,T4
5897  GOSUB 8000
5898  PRINT T4 - T3;"-- T4T3 ---R1--)":R1:"---T4-T3-R1=":T4 - T3 - R1
5899  IF  ABS ((T4 - T3) - R1) > R1 / 2 AND (YC < 4 OR XC < 4) THEN DR =
     0:D$ = 0:LL = R1 + R2 + 12:SM = 1: GOTO 5902 REM
      GOTO 5902MEANS CYLINDER
5900  GOTO 5920
5902  PRINT " CYLINDER , SM=";SM
5903  IF LL < 43 THEN HG = LL / 43: GOTO 5920
5904 HG = (LL - 43) / 54 + 1
5920  PRINT " ":DR;" DEG  ";HG;" INCH"
5930  GOSUB 7000
5940  PRINT "TH/":TN
5950  PRINT "CS/H/"; INT (YC)
5960  PRINT "CS/V/"; INT (XC)
5970  GOSUB 8000
5990  IF SM = 0 THEN  PRINT "CUBIC *******"
5995  INPUT AS$
5997  PRINT D$:"PR#4": PRINT D$:"IN#4"
5999  GOTO 27090
6000  PRINT "HG/":T:"/":B:"/":L:"/":R
6010  RETURN
```

```
7ØØØ  REM
7Ø1Ø  PRINT D$:"PR#2": PRINT D$;"IN#2"
7Ø2Ø  RETURN
8ØØØ  REM
8ØØ5  PRINT D$:"PR#Ø": PRINT D$;"              IN#Ø"
8Ø1Ø  RETURN
9ØØØ  REM       X  Y   Z    P  R  J   SP
1ØØØØ  PRINT D$
1ØØ1Ø  IF E = 1 THEN  INPUT I$: PRINT I$:E = Ø
1ØØ2Ø  RETURN
2ØØØØ  REM  ROBOT CONVERSION
2ØØØ2  HOME
2ØØØ3  PRINT " _____________________________________"
2ØØØ5  PRINT " "
2ØØ1Ø  PRINT "     ROBOT IS IN MOTION"
2ØØ2Ø  PRINT " "
2ØØ3Ø  PRINT " _____________________________________"
2ØØ6Ø  REM
2ØØ7Ø  REM
2ØØ8Ø  REM
2ØØ9Ø  REM
2ØØ97  PRINT D$:"PR#4": PRINT D$:"IN#4"
2Ø1ØØ  REM  DEFINE ROBOT ARM CONSTANTS
2Ø1Ø1 H = 7.625: REM  SHOULDER HEIGHT ABOVE TABLE
2Ø1Ø2 L = 7.Ø: REM  SHOULDER TO ELBOW AND ELBOW TO WRIST LENGTH
2Ø1Ø3 LL = 3.8: REM  WRIST TO FINGERTIP LENGTH
2Ø1Ø4  REM
2Ø11Ø  REM  DEFINE OTHER CONSTANTS
2Ø111 PI = 3.14159265
2Ø112 C = 57.2957795: REM  DEGREES IN 1.ØØRADIAN
2Ø113 R1 = 1: REM  FLAG FOR WORLD COORDINATES
2Ø114  REM
2Ø12Ø  REM  DEFINE ROBOT ARM SCALE FACTORS
2Ø121 S1 = 1125:S2 =  - S1: REM  STEPS/RADIAN, JOINTS 1&2
2Ø122 S3 =  - 661.2: REM  STEPS/RADIAN, JOINT 3
2Ø123 S4 =  - 244.4:S5 = S4: REM  STEPS/RADIAN, JOINTS 4&5
2Ø124 S6 = 371: REM  STEPS/INCH, HAND
2Ø125  REM
2Ø13Ø  REM  INITIALIZATION
2Ø131  DIM UU(7,4Ø): REM  ROOM FOR 4Ø STEPS
2Ø132  GOTO 265ØØ
2Ø133 P1 = Ø:P2 =  - 5Ø8:P3 =  + 1162:P4 =  + 384:P5 = P4:P6 = Ø
2Ø134  REM  LINE 133 IS THE NUMBER OF JOINT STEPS FROM T1=Ø,T2=Ø,T3=Ø,T4=
     Ø,T5=Ø, AND J=Ø, TO X=5,Y-Ø,Z=Ø,P=-9Ø,R=Ø, AND J=Ø
2Ø136  REM
2Ø137  REM  READ IN FIRST LINE FOR INITIALIZATION
2Ø138  READ X,Y,Z,P,R,GP,S
2Ø139  PRINT "SET ARM TO THE FOLLOWING POSTION & ORIENTATION"
2Ø14Ø  PRINT "USING TEACH CONTROL PENDANT, PRESS MODE KEY WHEN FINISHED"
2Ø141  PRINT "      X=";X;"INCHES"
2Ø142  PRINT "      Y=";Y;"INCHES"
2Ø143  PRINT "      Z=";Z;"INCHES"
2Ø144  PRINT "  PITCH=";P;"DEGREES"
2Ø145  PRINT "   ROLL=";R;"DEGREES"
2Ø146  PRINT "  HAND=";GP / S6;"INCHES"
2Ø147  PRINT "@DELAY 85": INPUT I
2Ø15Ø  PRINT "@SET 2ØØ": INPUT I: REM  MOVE ARM
2Ø155  PRINT "@RESET": INPUT I
2Ø157  HOME : VTAB 7: HTAB 5
2Ø159 U = Ø
2Ø16Ø  GOTO 27ØØØ
2Ø161  PRINT "INPUT X,Y,Z POSITION AND PITCH, ROLL, HAND GRIP, AND [illegible]"
2Ø162  INPUT "          X=";X
2Ø164  INPUT "          Y=";Y
2Ø166  INPUT "          Z=";Z
2Ø168  INPUT "      PITCH=";P
```

```
24070  PRINT "      Z=";Z;"INCHES"
24080  PRINT " PITCH=";P;"DEGREES"
24090  PRINT "  ROLL=";R;"DEGREES"
24100  PRINT "  HAND=";GP / S6;"INCHES"
24110  RETURN
24120  REM
24130  REM
24140  REM  ROUTINE TO CONVERT CARTESIAN COORDINATES
24150  REM  TO NUMBER OF JOINT STEPS AWAY FROM START POSITION
24160  REM
25000  REM
25010  REM  BACKWARD SOLUTION CALCULTIONS
25012  PRINT " ** TEST 12 C=";C;" PI=";PI;" LL=";LL
25014 LL = 3.8:L = 7.0:R1 = 1: REM  USED VARIABLE IN VPU
25020 P = P / C:R = R / C
25030  IF X = 0 THEN T1 =  SGN (Y) * PI / 2
25040  IF X (  ) 0 THEN T1 =  ATN (Y / X)
25050  IF T1 ( 0 THEN  PRINT : PRINT "------------------.T1=";T1
25060 RR =  SQR (X * X + Y * Y)
25070  IF RR ( 2.25 AND Z ( 15 THEN  PRINT : PRINT "HAND TOO CLOSE TO BOD
     Y. RR=";RR
25080  IF RR ) 17.8 THEN  PRINT : PRINT "REACH OUT OF RANGE. RR=":RR
25090 R0 = RR - LL *  COS (P)
25100  IF X ( 2.25 AND Z ( 1.25 AND R0 ( 3.5 THEN  IF P (  - 90 / C THEN
       PRINT : PRINT "HAND INTERFERENCE WITH BASE."
25110  REM  NOTE THAT THE ABOVE STATEMENT MAY BE ALTERED TO ACCOMODATE MO
     VES CLOSE TO THE BASE
25120 Z0 = Z - LL *  SIN (P) - H
25130  IF R0 = 0 THEN B = ( SGN (Z0)) * PI / 2
25140  IF R0 (  ) 0 THEN B =  ATN (Z0 / R0)
25150 A = R0 * R0 + Z0 * Z0
25160 A = 4 * L * L / A - 1
25170  IF A ( 0 THEN  PRINT : PRINT "REACH OUT OF RANGE FOR SHOULDER
             AND ELBOW.": GOTO 25500
25180 A =  ATN ( SQR (A))
25190 T2 = A + B
25200 T3 = B - A
25210  IF T2 ) 144 / C OR T2 (  - 35 / C THEN  PRINT : PRINT "SHOULDER OU
     T OF RANGE. T2=":T2 * C
25220  IF T2 - T3 ( 0 OR T2 - T3 ) 149 / C THEN  PRINT : PRINT "ELBOW OUT
      OF RANGE. T3=":T3 * C
25230  IF (R ) 270 / C OR R (  - 270 / C) THEN  IF (P ) ((90 / C + T3) -
     (R + 270 / C)) OR P ( (( - 90 / C + T3) + (R - 270 / C))) THEN  PRINT
     : PRINT "PITCH OUT OF RANGE. PITCH=":P * C
25240  IF P ) (90 / C + T3) OR P ( ( - 90 / C + T3) THEN  PRINT : PRINT "
     PITCH OUT OF RANGE. PITCH=":P * C
25250  IF (R ) (360 / C -  ABS (P - T3)) OR R ( ( - 360 / C +  ABS (P - T
     3))) THEN  PRINT : PRINT "ROLL OUT OF RANGE. ROLL=":R * C
25260 T4 = P - R - R1 * T1
25270 T5 = P + R + R1 * T1
25500  REM  **** CHECK #5170 ******
26000  REM  CORRECT COORDINATES
25010 W1 =  INT (S1 * T1 + .5) - P1
26020 W2 =  INT (S2 * T2 + .5) - P2
26030 W3 =  INT (S3 * T3 + .5) - P3
26040 W4 =  INT (S4 * T4 + .5) - P4
26050 W5 =  INT (S5 * T5 + .5) - P5
26060  RETURN
25165  INPUT "       Z=":Z
25500  REM  FROM 20132 GOTO 20133
25510  REM   SS=1 MEANS ORIGIN Z=8, X=9
26520 SS = 1
26530  IF SS = 0 THEN  GOTO 20133
26590 UU(1,0) = 0
26600 UU(2,0) =  - 862
26610 UU(3,0) =  - 931
```

```
20170  INPUT "     ROLL=";R
20171  INPUT "     HAND=";GP
20172  INPUT "    SPEED=";S
20175  INPUT "HIT RETURN TO GO ON:";A$
20176  IF SS = 0 GOTO 20180
20178 X = X + 4:Z = Z + B
20180  IF X ( 0 THEN 21000
20190  GOSUB 24000 REM    SHOW COORDINATES
20200  GOSUB 25000
20202  PRINT "UU(1,";U;")=";UU(1,U);"      UU(2,";U;")=";UU(2,U)
20203  PRINT "UU(3,";U;")=";UU(3,U);"      UU(4,";U;")=";UU(4,U)
20204  PRINT "UU(5,";U;")=";UU(5,U);"      UU(6,";U;")=";UU(6,U)
20205  PRINT "UU(7,";U;")=";UU(7,U)
20206  PRINT "W1=";W1;"    W2=";W2;"    W3=";W3;"    W4=";W4
20207  PRINT "W5=";W5;"    GP=";GP;"    S=";S
20210  PRINT "@STEP ";S;",";W1 - UU(1,U);",";W2 - UU(2,U);",";W3 - UU(3,U
      );",";W4 - UU(4,U);",";W5 - UU(5,U);",";W3 - UU(3,U): INPUT I
20220 U = U + 1
20221 UU(1,U) = W1
20222 UU(2,U) = W2
20223 UU(3,U) = W3
20224 UU(4,U) = W4
20225 UU(5,U) = W5
20226 UU(6,U) = GP
20227 UU(7,U) = S
20240  PRINT "@STEP";S;",0,0,0,0,0,";GP: INPUT I
20242  IF QQ = 1 THEN  GOTO 27070
20244  IF QQ = 2 THEN  GOTO 27690
20246  IF QQ = 3 THEN  GOTO 27072
20248  IF QQ = 4 THEN QQ = 0: GOTO 27750
20250  IF QQ = 5 THEN QQ = 0: GOTO 27900
20252  IF QQ = 6 THEN QQ = 0: GOTO 28000
20254  IF QQ = 7 THEN QQ = 0: GOTO 28060
20256  IF QQ = 8 THEN QQ = 0: GOTO 28200
20257  IF QQ = 9 THEN QQ = 0: GOTO 28400
20258  IF QQ = 10 THEN QQ = 0: GOTO 28500
20259  IF QQ = 11 THEN QQ = 0: GOTO 28560
20260  GOTO 20161
20270  REM  COMPARE 20290 THRU 20320 WITH OTHERS ,DIFFERENT RESULT?
21000 CY = 0
21001  PRINT "RETURN TO INITIAL POSITION"
21002  PRINT "@READ": INPUT I
21003  INPUT A,B,C,D,E,F,G
21004  PRINT "@CLOSE 242": INPUT I
21005  PRINT "@STEP 242,"; - A;","; - B;","; - C;","; - D;","; - E;","; -
     C: INPUT I
21008  PRINT "RUN THE PROGRAM"
21010  FOR I = 1 TO U
21020  PRINT "@STEP ";UU(7,I);",";UU(1,I) - UU(1,I - 1);",";UU(2,I) - UU(2
     ,I - 1);",";UU(3,I) - UU(3,I - 1);",";UU(4,I) - UU(4,I - 1);",";UU(5
     ,I) - UU(5,I - 1);",";UU(3,I) - UU(3,I - 1): INPUT N
21030  IF UU(6,I) ( 0 THEN 21060
21040  PRINT "@STEP ";UU(7,I)",0,0,0,0,0,";UU(6,I): INPUT N
21050  GOTO 21100
21060  PRINT @CLOSE245":INPUT N
21080  PRINT "@STEP 240,0,0,0,0,0,";UU(6,I): INPUT N
21100  NEXT I
21110 CY = CY + 1
21120  PRINT "CYCLE ";CY
21130 ,30000
24000  REM
24010  REM  DISPLAY COORDINATES
24020  PRINT "ROBOT ARM IS MOVING TO THE FOLLOWING"
24030  PRINT "COORDINATES:"
24050  PRINT : PRINT "      X=";X;"INCHES"
24060  PRINT "      Y=";Y;"INCHES"
```

```
26620 UU(4.0) = 0
26630 UU(5.0) = 0
26640 UU(6.0) =  - 931
26650 UU(7.0) = 230
26670  GOTO 20133
27000  REM   FROM INITIALIZATION  20160
27010 X = 6:Y = 0:Z = 17
27020 P = 45:R = 0:GP = 0
27030  GOSUB 24000
27040  GOSUB 25000
27050 OO = 1: REM   TO BE COMMING BACK
27060  GOTO 20202
27070 OO = 0
27072  PRINT " ------------------------------------"
27075 OT = 1
27077  IF OT = 0 THEN OO = 3: GOTO 20161
27080 OO = 0: REM  FROM 20161 THRU---
27081  REM  GET DATA OF VPU
27082  HOME : GOTO 102: REM  GOING TO VPU
27090  REM  FROM 5999 VPU FINISHED
27092 XR = (430 - XC) / 41
27100 YR = (YC - 200) / 41
27110 ZR = HG + 1: REM     OR ZR=(HG-1)/2 IF CUBIC
27120 GG = FG
27140 PR =  - 90
27160  PRINT "XR=":XR:"  YR=":YR:"  ZR=":ZR
27170  PRINT "PITCH PR=":PR:"  ROTATION  DR=":DR:"  HAND-GRIP GG=":GG
27180  REM  CONVERSION FROM CARTESIAN TO ROBOTIC REAL CARTESIAN
27491  IF XR ) 3 THEN  GOTO 27500
27492  IF YR ( 2 THEN  GOTO 27494
27493 YR = YR - 0.27:XR = XR + 0.16: GOTO 27519
27494  IF YR ( 0 THEN  GOTO 27496
27495 YR = YR - 0.15:XR = XR + 0.16: GOTO 27519
27496  IF YR (  - 2 THEN  GOTO 27498
27497 YR = YR + 0.12:XR = XR + 0.16: GOTO 27519
27498 YR = YR + 0.2:XR = XR + 0.12: GOTO 27519
27499  IF YR ) 0 THEN YR = YR - 0.1: GOTO 27500
27500  IF XR ) 5 THEN  GOTO 27508
27501  IF YR ( 2 THEN  GOTO 27503
27502 YR = YR - 0.2:XR = XR + 0.05: GOTO 27519
27503  IF YR ( 0 THEN  GOTO 27505
27504 YR = YR - 0.2:XR = XR + 0.0: PRINT " TEST$$": GOTO 27519
27505  IF YR (  - 2 THEN  GOTO 27507
27506 YR = YR + 0.1:XR = XR + 0.0: GOTO 27519
27507 YR = YR + 0.2:XR = XR + 0.0: GOTO 27519
27508  IF YR ( 2 THEN  GOTO 27510
27509 YR = YR - 0.25: GOTO 27519
27510  IF YR ( 0 THEN  GOTO 27512
27511 YR = YR - 0.19:XR = XR - 0.1: GOTO 27519
27512 YR = YR:XR = XR - 0.06: GOTO 27519
27519 Y = YR:X = XR + 4
27520 Z = ZR + 8
27530 R =  - DR
27540 P =  - 90
27550 GP = 0
27555 LL = 3.8: REM  LL WAS LENTH IN VPU LL=3.8 IS FOR ROBOT
27570 S = 230
27580  PRINT "X=":X:"  Y=":Y:"  Z=":Z:"  R=":R
27590  PRINT " P=":P:"  GP=";GP;" *** HG=";HG;"  S6=":S6
27600  GOSUB 24000
27610  GOSUB 25000
27620 OO = 2
27630  GOTO 20202
27690  REM  FROM 20202-20244
27692 OO = 0
27700  REM  TO CATCH OBJECT
```

display 4?
Backboard

```
27701  IF GG < 1 THEN GG = 1
27702  IF GG > 2 THEN  PRINT "*** TOO BIG ***": GOTO 29000
27703  PRINT "@STEP 220,0,0,0,0,0,";: INT ((GG + 1.0) * 371): INPUT I
27707 Z = 8.4:S = 215:GP = 0
27709 R =  - DR:P =  - 90
27710  GOSUB 24000
27720  GOSUB 25000
27730 OO = 4
27740  GOTO 20210: REM  INSTEAD OF 20202
27750  PRINT "@CLOSE 220 "
27760  INPUT I
27800 Z = 9.4:S = 220:GP = 0
27802 P =  - 90:R =  - DR
27810  GOSUB 24000
27820  GOSUB 25000
27820 OO = 5
27840  GOTO 20210
27900  REM  UP ON THE GOAL PLACE
27910  IF SM = 0 THEN Z = 9.4:S = 220:GP = 0:X = 6:Y = 4:R = 0:P =  - 90
27920  IF SM = 1 THEN Z = 9.4:S = 220:GP = 0:X = 6:Y =  - 4:R = 0:P =  -
     90
27930  GOSUB 24000
27940  GOSUB 25000
27950 OO = 6
27960  GOTO 20210
27970  REM  FROM
28000  REM  FROM 2770
28010 Z = 8.4:S = 200:GP = 0:P =  - 90
28020  GOSUB 24000
28030  GOSUB 25000
28040 OO = 7
28050  GOTO 20210
28060  REM  FROM 2770
28070  PRINT "@STEP 220,0,0,0,0,0,";400
28080  INPUT I
28100  REM  UP OVER THAT GOAL PLACE
28110 Z = 10.2:P =  - 90:GP = 0
28120 S = 220
28130  GOSUB 24000
28140  GOSUB 25000
28150 OO = 8
28160  GOTO 20210
28200  REM  CLOSE HAND
28210  PRINT "@CLOSE 235"
28212  INPUT I
28300  REM  GO BACK TO ORIGIN
28310 X = 9:Y = 0:Z = 10.2
28320 GP = 0:S = 230:R = 0
28330 P =  - 90
28340  GOSUB 24000
28250  GOSUB 25000
28360 OO = 9
28370  REM
28400  REM  GOTO ORIGIN Z=8+2.2
28410 X = 9:Y = 0:Z = 10.2
28420 GP = 0:S = 230:R = 0
28430 P =  - 90
28440  GOSUB 24000
28450  GOSUB 25000
28460 OO = 10
28470  GOTO 20210
28500  REM  FROM 20258
28510 Z = 8:P =  - 90:GP = 0
28520  GOSUB 24000
28530  GOSUB 25000
28540 OO = 11
```

```
28550  GOTO 20210
28560  REM  FROM 20259
29000  REM  FROM 27702
29004  PRINT " ____________________________________"
29100  REM
29102  PRINT "  YOU DID A GOOD JOB THANK YOU !"
29200 VZ = 1
29300  IF VZ = 0 GOTO 20161
29400  END
30000  DATA     5, 0,   0, -90, 0. 0.230
30010  DATA    8, 0.   2. -90. 0. 800, 242
30020  DATA    8. 0,0.5, -90, 0. -50, 242
30030  DATA    8, 0.   2. -90. 0.    0. 240
30040  DATA    6. 5.   2, -90. 0.    0, 240
30050  DATA    6, 5,0.5, -90,90. 300, 242
30060  DATA    6. 5.   2. -90.90,   -1, 242
30070  DATA    8, 0,   2. -90, 0. 800. 240
30080  DATA  -999,0,  0.     0. 0,    0,   0
30090  END
```