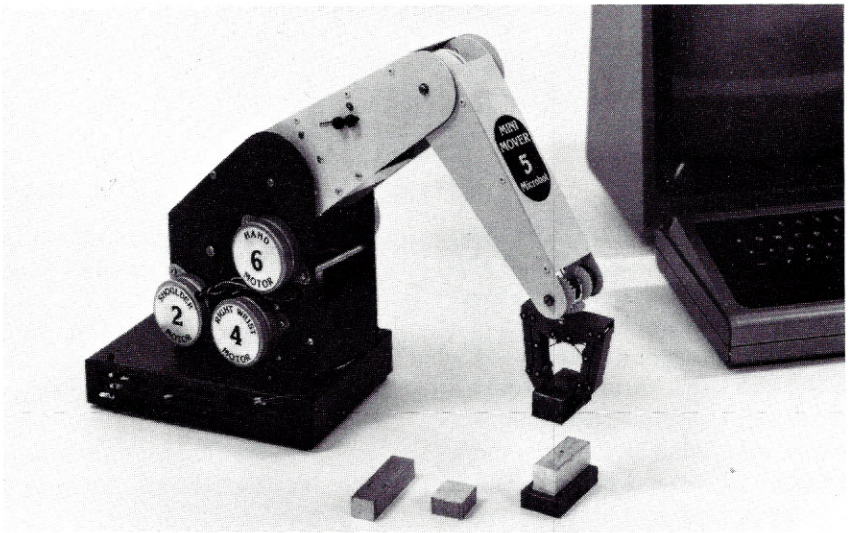


INTRODUCING MINI MOVER 5

John W. Hill, Ph.D.
Design Engineer, Microbot
1259 El Camino Real, Suite 200
Menlo Park, CA 94025

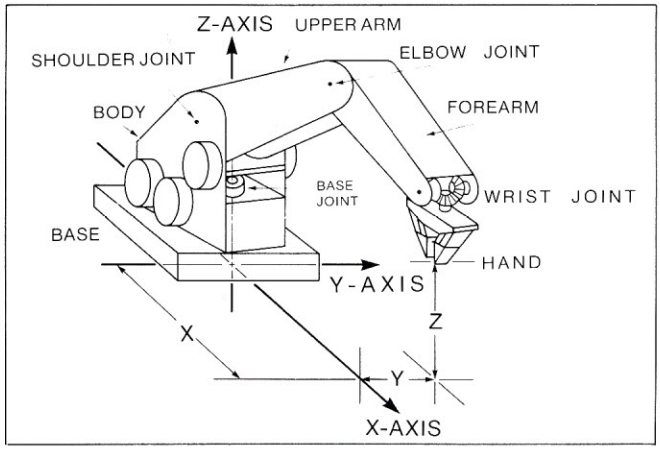


Figure 1. Major structural components. Note that a Cartesian coordinate reference frame is superimposed. The origin of this frame is where the centerline of the base joint meets the tabletop.

At the Fifth West Coast Computer Faire, held in San Francisco March 14-16, 1980, Microbot, Inc. introduced a new table-top robot arm. For a relatively low cost, the mechanical arm, its computer interface, and supporting software permit almost any microcomputer owner to get started in robotics without the effort of designing and building his own manipulator system. This versatile unit will prove useful to industry for robot evaluation as well as other light applications. Here now is a discussion of the design and operation of the Mini Mover 5, as told by its designer.

Previously, work with advanced computer-controlled manipulators was limited to laboratories at a few research oriented-universities in the US and abroad. This research has concentrated on languages for robot control and programming, computer programs that plan their own manipulation strategies, methods of efficiently performing

complex coordinate transformations, the use of sensory feedback to improve performance, and the study of manipulator kinematics. Typically, this work relies heavily on mathematical analysis, and requires highly sophisticated computing systems. Reference [1] provides a fairly complete survey of the work currently being done in these areas.

With the introduction of computer control to modern industrial robots, the results of this research are now becoming available for application in the factory environment. [2] Nevertheless, these complex new robots are still designed as heavy duty industrial equipment. Their price range, from $35,000 to $120,000 tends to hinder their use in robot experimentation by individuals and schools, as do the carefully guarded proprietary hardware and software details of these machines.

For two years, Microbot has been developing a low-cost manipulating system so that both amateur robotics enthusiasts and others could participate in this exciting field with a modest budget. The objective has been to provide a complete package including the arm, computer interface, software subroutines for coordinate conversions and control, and a high-level robot programming language, so that the user would not have to undertake a major development effort to put the unit in operation. The remainder of this article give a brief overview of the design and operation of the Mini Mover 5 as well as some typical methods of applying it. Readers interested in more technical detail will find it in the Mini Mover 5 Reference and Applications Manual [3].

Arm Mechanical Design

The arm consists of a stationary base and four movable segments connected in series by the base, shoulder, elbow, and wrist joints. The major structural components are illustrated in Figure 1. Implementation of the wrist joint is accomplished by a differential gear mechanism which can rotate the gripper in both pitch and roll. The resulting five degrees of freedom permit combined motions of the body, shoulder, elbow and wrist to position the arm anywhere from close to the base to 17.5 inches away, as shown in Figure 2. The configuration of these joints defines the position and orientation of the gripper within a partial sphere with a radius of 17.5 in. (444 mm).

The arm members are hollow, formed from folded aluminum sheet metal. The three outer segments are joined by hinge shafts that define the axis of the joint. Each joint is controlled by a flexible cable that runs from a drive unit mounted on the base to a pulley mounted on the joint shaft. Cable drive was selected because it is lightweight, highly efficient, and permits all the drive motors to be mounted on the body. This keeps weight out of the extremities, resulting in greater payload capacity.*

The shaft of the base joint is hollow, so that the drive lines for each of the motors can be brought from the interface card contained in the base of the arm. Since the axes of the three outer joints are parallel, mounting the drive motors on the body instead of the base eliminates the complications of routing the drive cables for the outer segments through a rotating joint.

The drive unit for each joint consists of a stepping motor, reduction gearing and a cable drum. From each drum, a tensioned cable goes out over pulleys to the member being driven and then returns to the drum. Rotation of the drum causes rotation of each member in proportion to the ratio of the diameter of the drive pulley attached to that member to the diameter of the drum.

The arrangement of cables used in the Mini Mover 5 is shown in Figure 3. Since the cabling design was made to simplify the geometry and maintain a low cost, several of the joints interact. Movement of one joint may result in

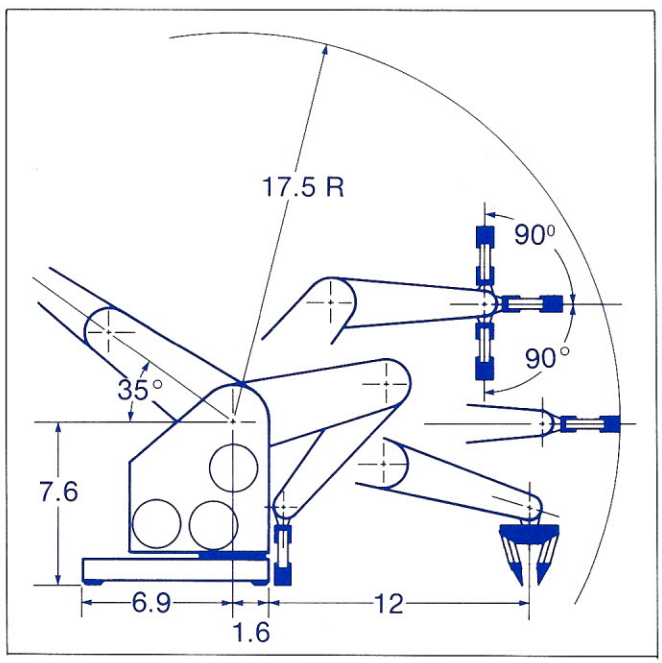


Figure 2. Mini Mover 5 working envelope.

Editor's Note: For descriptions of two other cable-driven manipulators and related design issues, see the article "Robotics Research in Japan" in the Winter 1979 issue of ROBOTICS AGE, Vol. 1, No. 2.

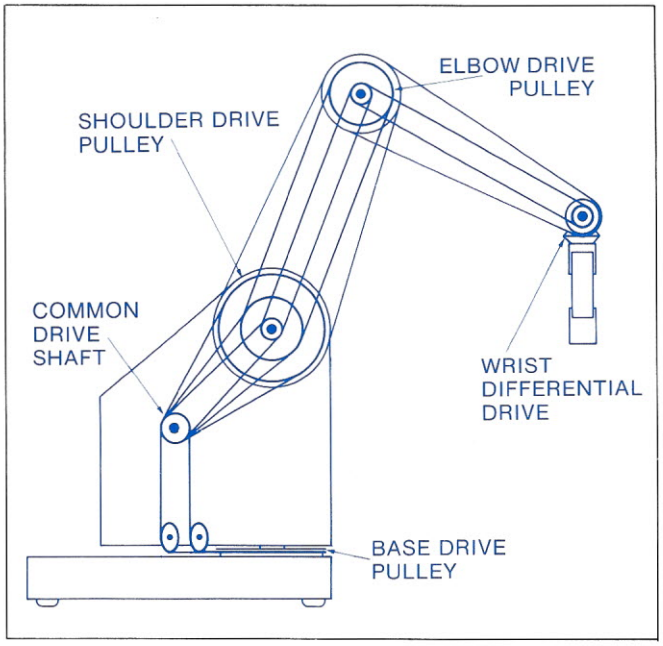


Figure 3. Cabling diagram.

possibly unwanted motion in another. Compensation for these interactions can be made in software. Thus, the cabling details must be understood in order to properly control the motion of the arm. As explained below, these interactions have been carefully designed so that in most cases they actually simplify the control problem.

Base Rotation (Joint 1)—The base drive cable passes over two idler pulleys, making a 90 degree bend, to a drive pulley fixed to the base. The base drive motor causes the entire arm to rotate about the base joint.

Shoulder Bend (Joint 2)—The shoulder drive cable passes around the drive pulley on the upper arm segment to rotate it about the shoulder joint.

Elbow Bend (Joint 3)—The elbow drive cable passes around an idler pulley on the shoulder axis to a drive pulley fixed to the lower arm segment. Joints 2 and 3 interact, so that changing the shoulder angle results in an equal change in the elbow angle. This interaction has been designed so that the elbow drive, in effect, controls the angle of the forearm with respect to the horizontal. However, the limits of forearm motion are measured with respect to the upper arm, not the horizontal (x-y) plane.

Wrist (Joint 4, left and Joint 5, right)—The fourth and fifth drive cables pass around idler pulleys on both the shoulder and elbow joints and terminate on the drive pulleys of the left and right wrist differential gears. The interaction of these joints with joints 2 and 3 has the effect that the wrist drives control the angle of the hand with respect to the horizontal (pitch) and

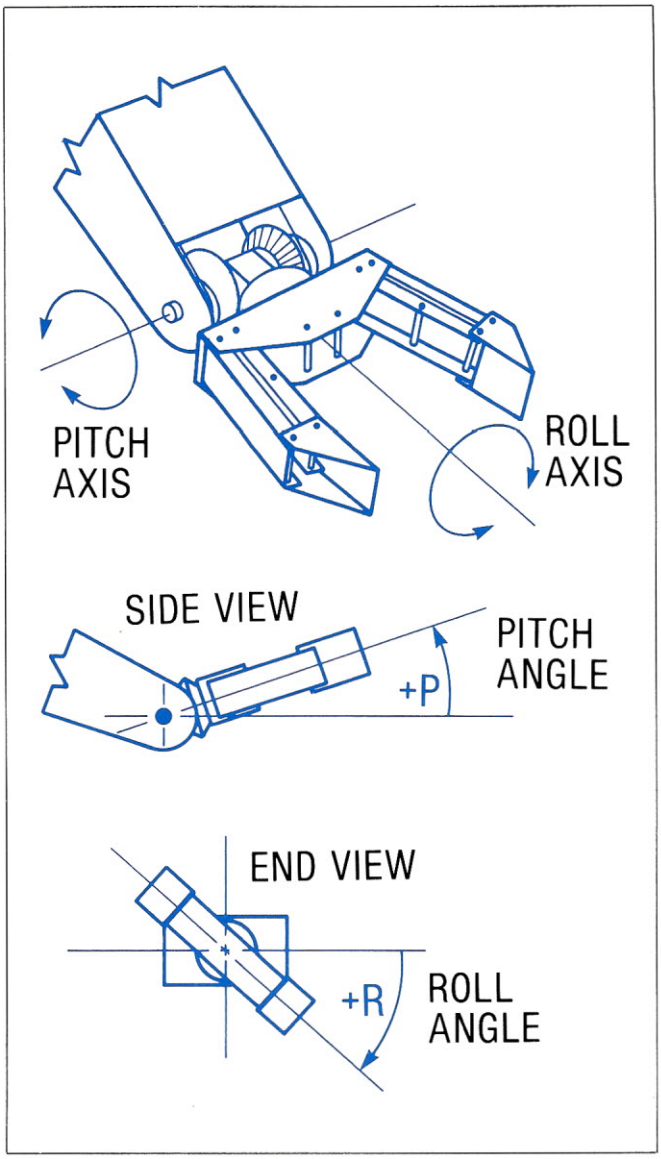


Figure 4. Definition of Roll and Pitch angles. All other joint angles are measured from the X axis.

the rotation of the hand around its pointing vector (roll) as shown in Figure 4. Through the action of the differential, the pitch angle P equals the average of the positions of the left and right gears and the roll angle R is their difference. The range of motion is limited to ±270 degrees at each of the wrist gears due to limits of the cable length and to ±90 degrees in pitch due to interference from the lower arm.

Hand (Joint 6)—The hand drive cable passes over idler pulleys located on the shoulder and elbow joints, through the center of the wrist differential, and finally terminates at the hand. This drive interacts with joint 3 (elbow) in that elbow bend will cause the hand to open slightly. This can be compensated for by closing the hand exactly the same number of steps that the elbow is raised.

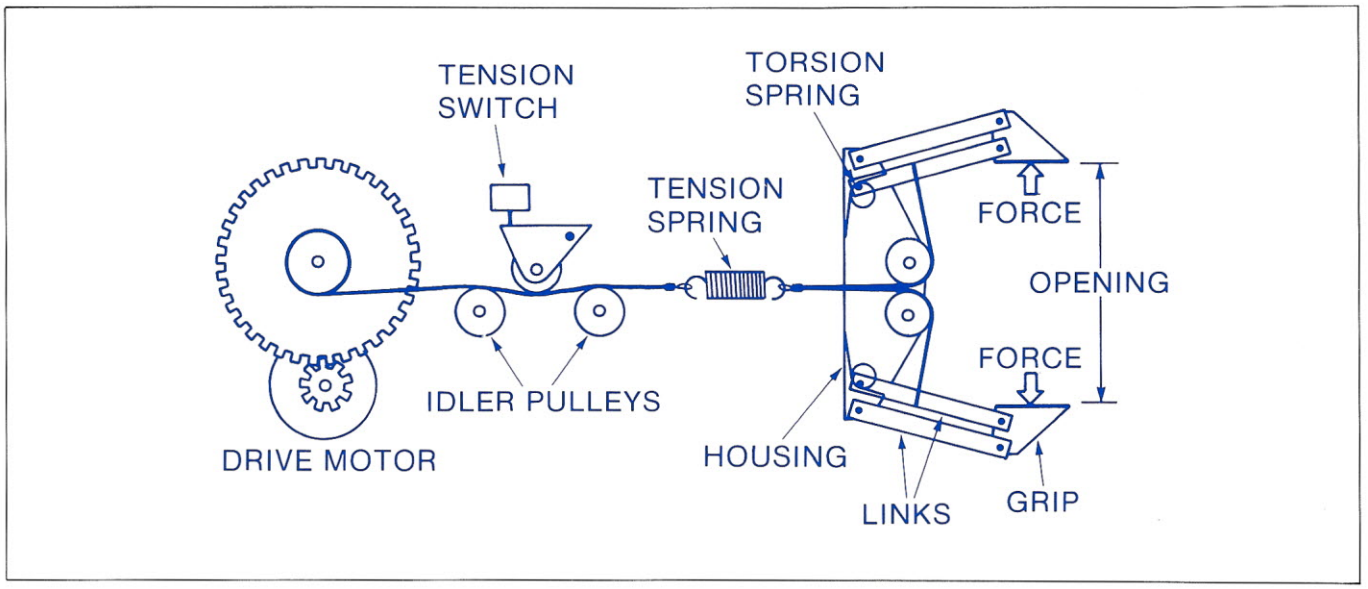


Figure 5. Control system for gripping.

The hand housing is a attached to the output miter gear of the differential gear set. The hand housing supports the two pairs of links, each pair of which terminates in a grip, as shown in Figure 5. The housing, links and grips are attached to each other by small pins. Torsion springs provide the return force to open the hand as the hand cable is slackened. As the hand cable is pulled in, the hand closes upon an object to be grasped and the cable tension mounts, activating a tension sensitive switch. This state of this switch can be read by the control computer which may stop the drive motor as soon as it closes, in which case the grasping force is limited to about 2 ounces (0.3N).

Gripping force can be increased by pulling in more cable after the tension switch closure is detected. Additional cable take-up stretches the extension spring which is mounted in series with the hand drive. This additional cable take-up is converted into gripping force at the rate of one pound for every 52 steps. The relationship between gripping force, hand opening, and drive motor rotation is shown in Figure 6. Thus, gripping force, as well as hand opening, can be controlled by the same cable drive. The performance specifications of the Mini Mover 5 are shown in Table 1.

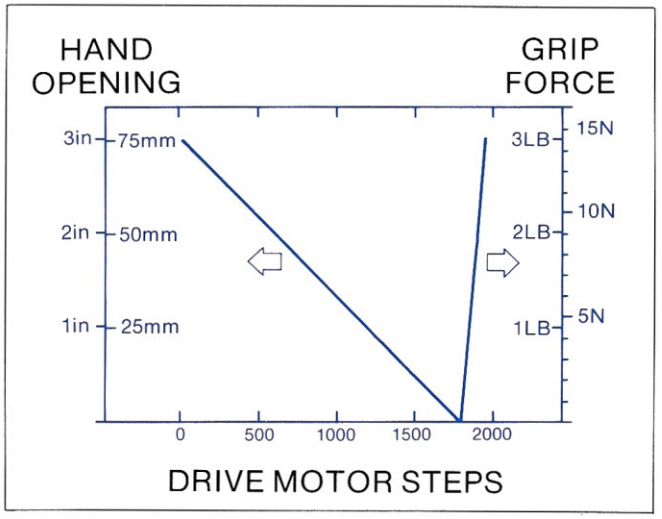


Figure 6. Plot of hand opening and grip force vs number of steps. Grip opening decreases as cable is taken up (left axis), until resistance is encountered. Thereafter, grip force increases (right axis). The plot shows the case when the hand is empty.

Table 1

Mini Mover 5 Performance Characteristics

GENERAL

Configuration	5 revolute axes and integral gripper
Drive	Electric Stepper Motors, open loop
Controller	Microcomputer provided by user
Interface	TRS-80™* or a 8-bit parallel
Program language	ARMBASIC for TRS-80™ [Z-80 driver available]
Power requirement	12 volts, 4 amps DC

PERFORMANCE

Resolution	.013 in (0.3) maximum on all axes
Load Capacity	8 oz (225 gm) at full extension 16 oz (450 gm) at half extension
Gripping force	3 lbs (13 N) max.
Reach	17.5 in (444 mm)
Static load force	4 lbs (18 N) max.

PERFORMANCE DETAILS

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 (20 mm/s)

PHYSICAL CHARACTERISTICS

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

*Registered Trade Mark of Tandy Corporation

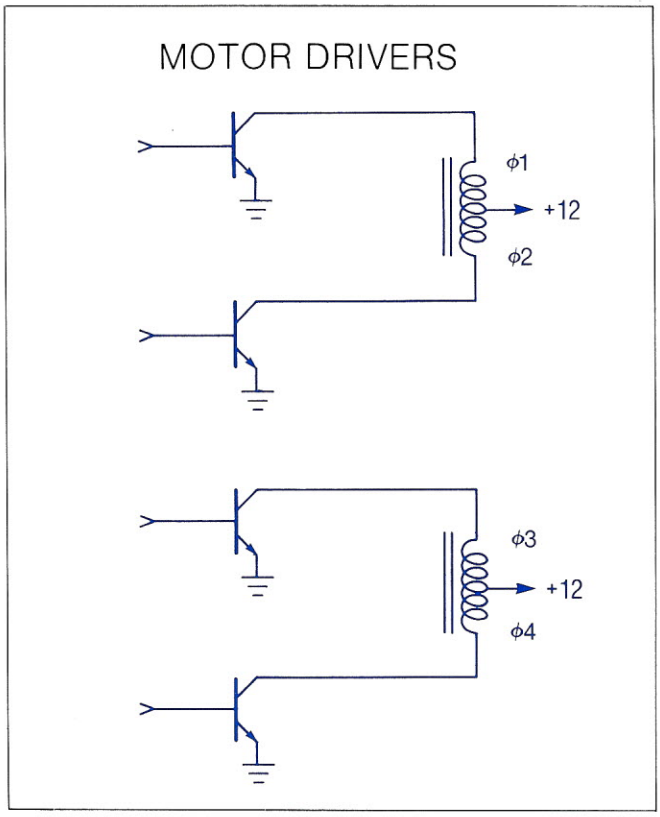


Figure 7. Simplified stepper motor drive circuit. Coils φ1 and φ2 are on one magnetic path. Coils φ3 and φ4 are on another.

Stepping Motor Control

Each of the cable drives is controlled by a stepper motor. The motors used have four separate windings, each driven by a power transistor. The drive is digital, with the transistors switched on or off to obtain a desired pattern of currents in the motor. By appropriately changing the pattern of currents a rotating magnetic field is obtained inside the motor, causing it to rotate in small increments or steps.* In the Mini Mover 5, each of the four coils is individually controlled from the computer. The simplified drive circuit is shown in Figure 7. This allows the pattern of motor currents to be controlled by the computer software, simplifying the drive circuit and reducing its cost.

In order to step a motor, the sequence of binary patterns shown in Figure 8 is output to the desired motor—the pattern on each row of the table, when sent to the corresponding drive transistors of the motor, will cause it to move on 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. To move the motor in the opposite direction, the sequence in which the entries are output must be reversed.

The particular entries shown generate a sequence known as "half-stepping", that is, the angular steps produced by the sequence are half the size specified by the motor manufacturer. Compared to full stepping, half-stepping produces smoother slow speed motions, doubles the accuracy of motor positioning, and reduces the power required. The motors used to drive the Mini Mover 5 are specified by the manufacturer at 48 steps per revolution and thus are stepped at 96 steps per revolution by the algorithm described.

Computer Interface

The Mini Mover 5 interface has been developed to connect the manipulator to Radio Shack's TRS-80™ Level II computer. By changing internal jumpers, the interface can be modified so that it will operate with any 8-bit parallel I/O ports (TTL levels). The interface is organized as seven 4-bit parallel outputs and a single 4-bit input, as shown in Figure 9. Information on the address lines is used to channel the four bits of output data to the appropriate motor drive through individual 4-bit latches. In this way the computer can control the 4-bit phase patterns on each motor with a minimum of hardware. After a phase pattern is sent to a 4-bit latch, it is held by the latch until the next pattern is sent. Outputs of the latches control the power drivers and their respective motor coils, shown previously in Figure 7.

	φ1	φ2	φ3	φ4	
CLOCKWISE ↓	0	1	0	1	↑ COUNTERCLOCKWISE
	0	1	0	0	
	0	1	1	0	
	0	0	1	0	
	1	0	1	0	
	1	0	0	0	
	1	0	0	1	
	0	0	0	1	

Figure 8. Table of drive patterns for stepper motor coils.

* More information on the operation and control of stepper motors can be found in References [4] and [5].

The grip switch is connected to one of the inputs of the auxillary port, permitting the computer to close the hand until the gripping force builds up. The other inputs and outputs of the auxillary port are available to the user for experimentation with other sensors or controls. These 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. Note also that a four-bit auxillary output port is available as well. This may be used to control another stepping motor such as might be used to drive a work turntable, for example, to control external parts feeders, or synchronize other equipment with arm operations.

ARMBASIC

A language for control of an arm is important to do useful work with it. For eample, printing on an output device is simplified in BASIC by the PRINT and LIST commands or in FORTRAN by the WRITE command. These high level commands remove the problem of controlling the carriage and printing mechanisms and keeping track of their position to produce the printout. In the same sense, arm control can be more effectively done through such high level commands.

ARMBASIC was developed for controlling the Mini Mover 5 from the TRS-80 computer. ARMBASIC is an extension to the Level II BASIC supplied by Radio Shack, consisting of 5 commands which allow complete control of the manipulaltor. These commands are @STEP, @CLOSE, @SET, @RESET, and @READ. Each of the ARMBASIC commands is implemented in Z-80 assembly language, both to conserve memory and to provide more rapid response of the manipulator. The listings of these routines and more detailed explanations of what the assembly language code does are provided in the Applications Manual [3].

The @STEP command causes each 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 motor steps, and (J1) through (J6) are expressions for the number of steps 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.

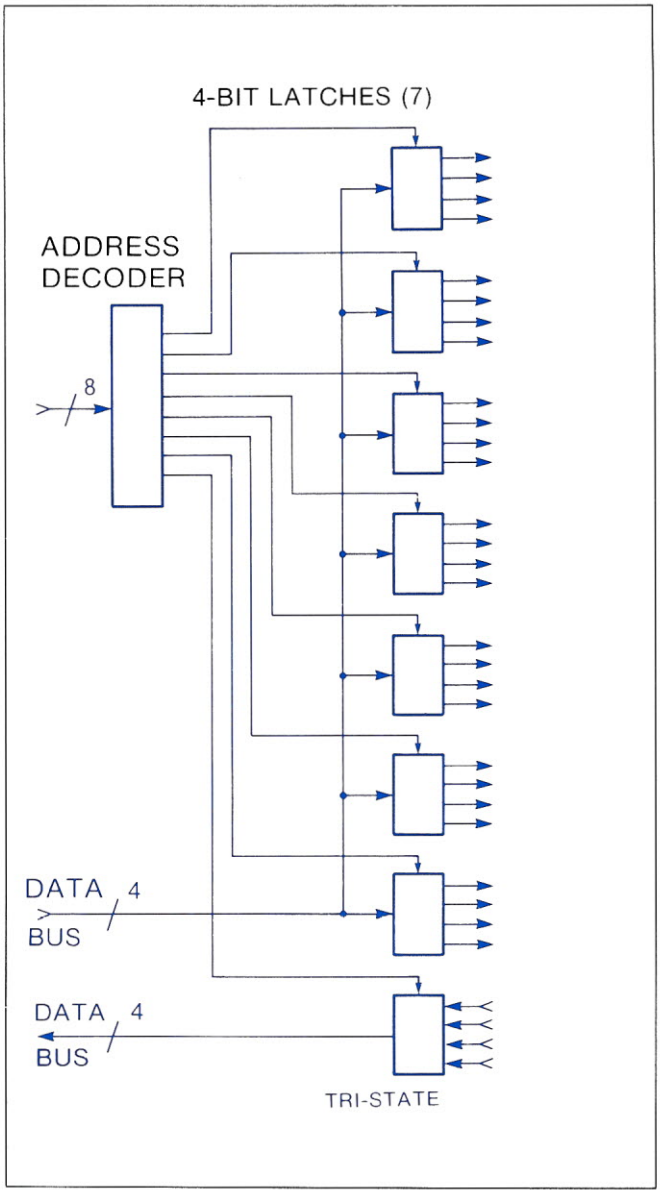


Figure 9. Block diagram of computer interface.

For each additional unit of delay, the driver will wait an additional .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 and the magnitude indicates the number of steps.

Additionally, the @STEP command compensates automatically for the interaction of the elbow joint and the hand opening, such that a command to move the elbow joint will not effect the hand opening. Note that the steps specified for joints 4 and 5 refer to the motors driving the left and right differential gears, so that the relation between the

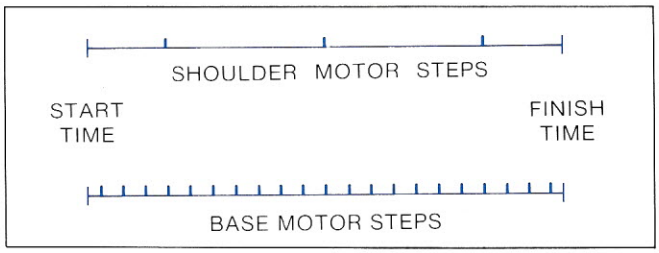


Figure 10. Step timing diagram coordinated movement.

gear positions and the wrist roll and pitch angles must be determined by the user's program.

An important function performed by the @STEP command is to linearly coordinate unequal motor commands. For example, if the base motor is told to move 21 steps, and the shoulder motor is told to move 3 steps, appropriate delays will be inserted between the steps of the shoulder motor so that each joint will begin and end its movement simultaneously, resulting in the smooth and uniform timing shown in Figure 10.

The @CLOSE command causes the hand to close until the grip-switch closes, indicating that the gripping force has built up to the threshold determined by the tension switch. The syntax of the CLOSE command is:

@CLOSE (D)

where the optional delay, (D), determines the speed of closing, as discussed in the @SET command.

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

@SET (D)

The optional delay (D) determines the speed of motion as discussed in the @STEP command. The keys and the joints that they control are shown in Figure 11. This mode is terminated by depressing the "0" (zero) key. Note that since the driver is capable of reading all the keys held down at once, depressing more than one key will result in all simultaneous movement of the respective joints. Releasing the keys will stop the motion.

As the above commands are executed, the number of steps commanded for each joint are counted and maintained a set of six 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 simply:

@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

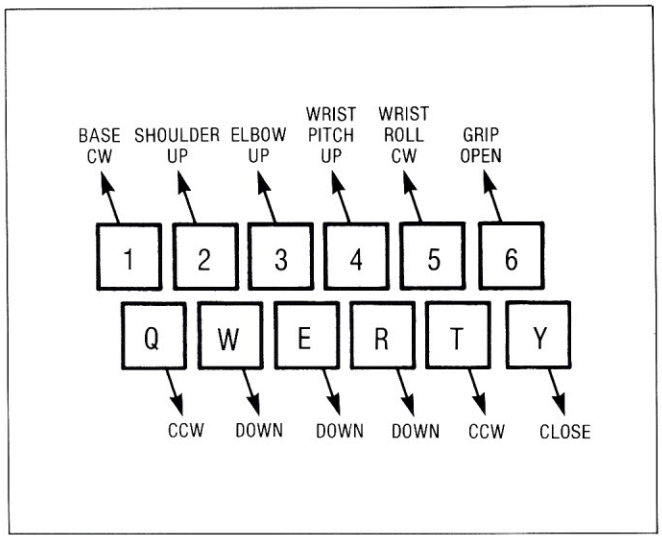


Figure 11. Use of keyboard to control arm.

backdriven (manually moved about). @RESET is used for arm initialization, as discussed in the following section. The syntax of the @READ command is:

@READ (V1), (V2), (V3), (V4), (V5), (V6)

where V1 to V6 are any 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.

Programming the Mini Mover 5

To illustrate the use of ARMBASIC in manipulation, we will discuss serveral methods of programming a "pick-and-place" task. The 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 deposited in another machine or onto a moving conveyor, for instance. To simplify the example, we will assume an unending supply of parts at the pick-up point and continual removal of parts at the destination point.

The task is defined in Figure 12. Pickup is from A and placement is at D. The approach and departure trajectories, AB and CD are usually required for practical reasons: an object resting on a flat surface normally must be lifted before it is translated instead of just sliding it along the surface. In an industrial environment, such a movement could possibly damage the part, the manipulator, or some other object in the workspace.

A simple sequence of ARMBASIC commands for performing the pick-and-place task is shown in Table 2. Movements are performed by the @STEP command, including opening the gripper (statement 60), P=(P1, P2, P3, P4, P5), Q=(Q1, Q2, Q3, Q4, Q5) and R=(R1, R2, R3, R4, R5) define vectors whose elements are the number of steps of each joint needed to move the arm from A to B, B to C, and C to D respectively. S is the number specifying

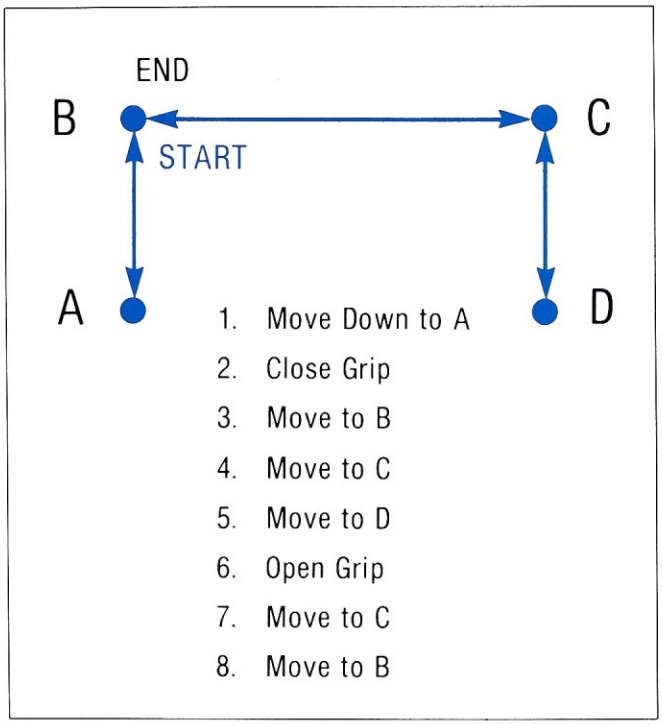


Figure 12. Description of the Pick-and-Place task (side view).

the speed and GR is the number of steps the gripper is to be opened to release the part.

This simple approach is not very practical, as 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 12) and letting the computer count and remember the number of steps each joint moves. This operation is frequently called "teach mode."

The teach program shown in Table 3 illustrates this 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. Even the desired grip opening is conveyed by teach control. Only the speed is entered numerically.

After the arm is positioned at each of the four locations, the values of the software position counters are read by the @READ command and 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), and D=(D1, D2, D3, D4, D5), are the values of the position counters of each joint. The P, Q and R vectors of the simple program (Table 3) can therefore be obtained by subtraction as follows: P=B-A, Q=C-B, R=D-C. The desired grip opening, GR, is measured by reading the grip position counter before and after closing the grip using the @CLOSE command. (Statements 170-200 in Table 2).

Where information on pickup or placement originates in

Table 2

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
```

Table 3

Pick-and-Place Teach Program

```
10 REM          ************************************************
20 REM          *                                              *
30 REM          *                  MINI MOVER 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 GRIPPER ON PART, TYPE 0 WHEN DONE"
150 PRINT "     ADJUST GRIP OPENING TO CLEAR PART"
160 @SET
170 @READ A1,A2,A3,A4,A5,C
180 @CLOSE:  REM     CLOSE GRIPPER 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 0 WHEN DONE"
220 @SET
230 @READ B1,B2,B3,B4,B5
240 PRINT "POSITION PART ABOVE PLACEMENT SITE, TYPE 0 WHEN DONE"
250 @SET
260 @READ C1,C2,C3,C4,C5
270 PRINT "POSITION PART AT PLACEMENT SITE, TYPE 0 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
```

Table 4

Pick-and-Place X Y Z Control Program

```
10 REM          ************************************************
20 REM          *                                              *
30 REM          *                  MINI MOVER 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=3.5   :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=618               :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 GRIP IF GR>0
470 @STEP S,0,0,0,0,0,GR
480 GOTO 300
490 REM   CLOSE GRIP 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
5020 P=P/C:R=R/C                          (IN RADIANS)
5030 RR=SQR(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(SQR(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
```

a computer, teach programming is often not practical. For example, to move pieces on a chess- or checkerboard 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 worktable facilitates reading of the pick-up and set-down coordinates directly.

The Cartesian coordinate control program shown in Table 4 shows how this is done with the Mini Mover 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 1). The *x, y, z,* coordinates (in inches) and pitch and roll angles (in degrees) at the fingertips are:

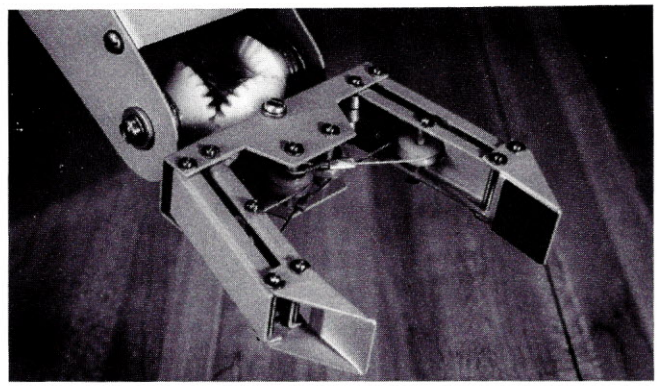
A=(8, 0, 0.5, -90, 0)
B=(8, 0, 1.5, -90, 0)
C=(6, 5, 1.5, -90, 90)
D=(6, 5, 0.5, -90, 90)

Note that the object is gripped at a point 0.5 inches above the table top (z=0) and is raised (or lowered) 1.0 inches for lift off (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 gripper is always straight down (pitch=-90 degrees), but the object is rotated 90 degrees on its vertical axis between pickup and set down (roll changes from 0 to 90 degrees).

Data for the Cartesian coordinate program is located after statement 10000. Each data statement specifies an arm position and orientation in terms of x, y, z, pitch and roll, and also includes a sixth parameter, governing the grip opening (if positive) or squeezing force (if negative) after each move, and a seventh parameter governing the speed setting during each move. The actual arm solution, beginning at statement 5000, takes an arm configuration specified by the variables X, Y, Z, P, and R, and finds the joint angles (in radians) that would place the arm in that configuration. It also includes several tests to determine if the postion and orientation requested is within the reach of the arm.

The first data statement gives the initialization point, which is used by the initialization phase of the program, statements 140-260, to tell the computer where the arm is in the coordinate system. Note that the arm position registers are not used by the program (although the @RESET is necessary). Instead, the operator is instructed to position the arm at the initialization point under control of the @SET command. The joint positions corresponding to that point are computed and stored in variables W1-W5. In the main part of the program, the relative offsets from one point to the next are computed and used to step the arm. Thus, all arm movements are made relative to the initialization point.

Since the computer has no way of measuring the arm's joint angles upon startup, such an initialization must be made by every program that relies on an external coordinate system. Similarly, all arm movements are made "open loop," or without position feedback. Therefore, a re-initialization is necessary if the arm is unable to successfully perform a movement due to overloading of collision. The use of open loop control by stepper motors eliminates the need for joint position sensors and associated interface circuitry and is thus an important design tradeoff that reduces the cost of the arm while still providing accurate positioning.



During the second phase of the program, statements 300-530, the arm is moved from data point to data point as the solutions are carried out. An arm solution takes less than one second. The joint angles corresponding to each data point are stored in array UU. After all the solutions are obtained, control is transferred to the third phase, beginning at statement 1000, where the pick-and-place cycle is run repeatedly without coordinate conversions by using the joint angles in UU.

Those who analyze the Cartesian coordinate program carefully will note that it is in fact general purpose, capable of generating arbitrary motion trajectories with up to 50 movements. This is sufficient for many assembly tasks such as building structures from blocks. The power of the ARMBASIC approach can be seen by comparing the simplicity of this program, which is less than 100 lines of code, with previous approaches which have required 8K or more of assembly language coding to do the same type of task. More information on this and other manipulation programs, as well as coordinate transformations, is given in the Mini Mover 5 Reference Manual [3].

References

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

[2] Weisel, W. and A. Katoh, "Beachheads for Robotics," *Proc. Fifth International Symposium on Industrial Robots*, pp. 1-10, Society of Manufacturing Engineers, Dearborn, Michigan (September 1975).

[3] Mini Mover 5 Reference and Applications Manual, Microbot, 1259 El Camino Real, Suite 200, Menlo Park, California (1980).

[4] Bober, R. E., "Taking the First Step," *BYTE*, pp. 35-112 (February 1978).

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