Interactive Computer Aided Design and Animation of Spatial Mechanisms

Summer 1983

Emmet B. Peter

University of Central Florida

Find similar works at: https://stars.library.ucf.edu/rtd

University of Central Florida Libraries http://library.ucf.edu

Part of the Engineering Commons

STARS Citation


https://stars.library.ucf.edu/rtd/708

This Masters Thesis (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of STARS. For more information, please contact leedotson@ucf.edu.
INTERACTIVE COMPUTER AIDED DESIGN AND ANIMATION
OF SPATIAL MECHANISMS

BY

EMMETT B. PETER III
B.S., University of Central Florida, 1978

RESEARCH REPORT
Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Engineering
in the Graduate Studies Program of the
College of Engineering
University of Central Florida
Orlando, Florida

Summer Semester
1983
ABSTRACT

The synthesis of planar and spatial mechanisms is often accomplished by either trial and error supplemented by computer analysis or by specific analytical techniques in the literature. In either case it is extremely helpful to be able to visualize a physical design as it emerges, and to see a graphic display of it in animated motion. This paper describes the development of a general interactive program for both analyzing and viewing a spatial 4-bar (RSSR) mechanism in animated motion. The analysis provides complete position, velocity, and acceleration information and, for the special case of the planar 4-bar, the same information is available for an arbitrary coupler point. The animation, while not real time, is sufficiently fast to provide the designer with a physical feel for the relative movement of the links. The program is written in interactive BASIC and is designed to run on a standard Apple II microcomputer. The result is a helpful tool for the mechanisms designer, and an example is presented to demonstrate the program's flexibility.
ACKNOWLEDGEMENTS

Special thanks must be extended to a friend and fellow student, Billy Koos. Billy is the author of the excellent plotting routine used in the program, and was invaluable in helping explain the intricacies of interfacing to machine language subroutines.

Thanks must also be extended to my advisor, Dr. Sayed Metwalli. Dr. Metwalli provided technical expertise as well as invaluable information on the proper presentation of technical writing. More importantly, he supplied the right combination of inspiration and encouragement to allow this project to be completed.

Above all, the highest appreciation must go to my lovely wife, Terre. Terre provided the encouragement and support needed to endure 5 years of part time graduate school, and she sacrificed many hours in order for me to complete my studies.
# TABLE OF CONTENTS

INTRODUCTION ................................................. 1

Chapter
   I. THEORY .................................................. 3
   II. COMPUTER IMPLEMENTATION ............................ 9
   III. EXAMPLE: STEERING LINKAGE .......................... 20
   IV. DISCUSSION AND CONCLUSIONS ......................... 25

Appendix
   A. RSSR ANALYSIS THEORY .................................. 27
   B. DISK ORGANIZATION ...................................... 32
   C. SUBPROGRAM LISTINGS: INPUT2, RSSR3, PRINT, and ANIMATE .............................. 35
   D. PLOTTING AND BINARY DISK STORAGE PROGRAM LISTINGS ................................ 50
   E. MEMORY MAP AND ARRAY FOR SUBPROGRAM ANIMATE; SUBLOGIC DATA SHEET ..................... 64

BIBLIOGRAPHY .................................................. 72
NOMENCLATURE

a .......... input link, spherical joint position
AO .......... input link, pivot (ground) position
Al .......... input link spherical joint initial position
b .......... output link, spherical joint position
BO .......... output link, pivot (ground) position
Bl .......... output link spherical joint initial position
d .......... coupler length
P .......... coupler point position, planar case
R .......... rotation matrix
UA, UB .... unit vectors, pivots a and b
α .......... input angle
β .......... output angle
ζ .......... coupler centered coordinate, along axis
η .......... coupler centered coordinate, right angle to coupler
GLOSSARY OF SELECTED COMPUTER TERMS

array: A set of lists of elements, usually variables or data.

BASIC: An interpretive computer language commonly used in microcomputers.

boot: Short for bootstrap, a technique or device designed to bring itself to a desired state by means of its own action.

call: A BASIC command that begins execution of a machine language program at the specified address.

code: A system of symbols which can be used by machines, in which specific arrangements have special meaning.

compiler: A language translation program, used to transform code meaningful to humans into code meaningful to a computer.

compressed: When used in reference to computer language or code, to remove all unnecessary statements, e.g. comment statements. This often makes programs execute faster at the expense of readability.

decision: A logical operation in which the resultant quantity is true if at least one (but not all) of the input values is true, and is false if the input values are all true or all false. This logic is sometimes used in memory based graphics systems to selectively erase lines or objects.

graphics dump: In memory based graphics systems, to copy or transfer the memory section containing the picture information, usually to an external storage device.

hexadecimal: Whole numbers in positional notation using 16 as a base. Hex numbers are noted by either a prefixed $ sign or a suffix of H, e.g. the number 15 is noted by either $0F or 0FH.

interpretive jump: In machine language, a command which causes another portion of the program to begin execution out of ordinary sequence.

lock out: The use of programmed logic to keep unwanted or invalid data from being entered into a program.

no op: A machine language command which the processor ignores. Usually used to reserve space for future code.

object code: The binary coded program which is the output after translation from the source language.

source code: The human-readable program which is translated into machine language object code.

utility: A standard routine to assist in the operation of the computer (e.g. device drivers, sorting routines)
INTRODUCTION

The use of the microcomputer in the engineering environment has increased dramatically in recent years. Current trends indicate that large networks of small computers will be displacing the larger centralized mainframe in many installations. The microcomputer (or personal computer) is interactive, friendly, and forgiving and there is every reason to believe that will be a useful engineering tool for years to come.

Microcomputers are well suited to mechanism design because this process is often one of synthesis by successive analysis, and interaction with the computer is important. It is the subject of this paper to describe the development of a general purpose microcomputer program for the analysis and animation of the spatial four-bar (RSSR) mechanism.

The RSSR spatial mechanism is one of the simplest of the spatial mechanisms, its two revolute ground joints and two spherical moving joints allowing for relatively straightforward physical implementation. Since its coupler is free to rotate about its axis, the RSSR's usefulness as a path and motion generator is limited; however, the planar four-bar is a special case of the
spatial RSSR and coupler motion can be considered and utilized.

A survey of existing computer programs reveals that there is already a move to adapt the larger programs to microcomputers. Micro-Kin-syn and Micro-Lineages are adaptations of the well-known 2-D synthesis programs, the former with hardware addition to a standard Apple IIe and the latter to the Terak computer (1). Other programs for analysis only include ADAMS (2), DRAM (3), DYMAC (4), IMP (5), and UCIN (6). These are very powerful general programs which include such features as dynamic analysis and generalized impact (DRAM); however, only ADAMS and IMP have 3-D graphics capability. There are no doubt many other more specialized programs and techniques for the analysis and synthesis of spatial mechanisms in the literature (6, 7). The program described herein should provide the designer with an inexpensive tool for the design and visualization of a large class of mechanism problems.
I. THEORY

Figure 1 shows the spatial RSSR conventions. The initial configuration of the mechanism is fully determined by 4 points and two unit vectors, and complete kinematic analysis can be conducted in closed form by additionally specifying the motion of the input link (angular travel, step size, velocity, and acceleration). This technique begins with a displacement constraint equation (the coupler must be a constant length) and solves for the unknown output angle. Similarly, velocity and acceleration constraint equations yield corresponding solutions for the velocity and acceleration of the output angle. Appendix A and Suh and Radcliffe (8) provide a more complete discussion of this analytical technique.

For the special case of the planar mechanism in three dimensional space, analysis of a general coupler point has meaning, since for this case the coupler can be imagined to be carried by revolute joints rather than spherical ones. It was decided that the general nature of the program could be preserved by allowing for the analysis of a general coupler point and by not restricting a planar mechanism to the X-Y plane, thus figure 2 illustrates the geometry used
Figure 1. RSSR Conventions
Figure 2. Planar Case
to describe the coupler point. The test for determining whether or not a mechanism is planar is shown in figure 3 and is conducted during the input section of the program.

Since the analytical technique yields the position, velocity, and acceleration of both point $A_1$ and $B_1$, kinematic analysis of the coupler point is straightforward. For the $j^{th}$ position of the planar case,

$$\hat{\zeta} = \text{direction}(b_j - a_j)$$

$$\hat{\eta}_j = \hat{\zeta}_j \text{rotated 90 deg about } UA \text{ (or } UB)$$

To evaluate $\hat{\eta}$ we must use $[R_\phi,u]$, a vector rotation matrix for rotating $\phi$ degrees about a unit vector $u$; for $\phi = 90$ deg and $u=UA$,

$$[R_{90},UA] = \begin{bmatrix}
UA_x^2 & UA_x UA_y - UA_z & UA_x UA_z + UA_y \\
UA_x UA_y + UA_z & UA_y^2 & UA_y UA_z - UA_x \\
UA_x UA_z - UA_y & UA_y UA_x + UA_z & UA_z^2
\end{bmatrix}$$

Thus $\hat{\eta}_j = [R_{90},UA] \hat{\zeta}_j \quad (1)$

The coupler point $P$ can be written as the sum of 3 vectors:

$$P_j = a_j + \zeta \hat{\zeta}_j + \eta \hat{\eta}_j \quad (2)$$

Substituting (1) for $\eta_j$ in (2)

$$P_j = a_j + \zeta \hat{\zeta}_j + \eta [R_{90},UA] \hat{\zeta}$$
Figure 3. Testing For Planar Inputs
But \( \xi_j = \text{direction (} b_j - a_j \text{)} \)
or \( \xi_j = \frac{(b_j - a_j)}{|b_j - a_j|} \)

Note that \( |b_j - a_j| = \text{constant} = \text{length of coupler d, thus} \)
we may write

\[
P_j = a_j + \xi \frac{d}{d} (b_j - a_j) + \eta \left[ R_{00}, UA \right] (b_j - a_j)
\] (3)

Taking derivatives to obtain velocity and acceleration,

\[
\dot{P}_j = \dot{a}_j + \xi \frac{d}{d} (\dot{b}_j - \dot{a}_j) + \eta \left[ \frac{d}{d} R_{00}, UA \right] (\dot{b}_j - \dot{a}_j)
\] (4)

\[
\ddot{P}_j = \ddot{a}_j + \xi \frac{d}{d} (\ddot{b}_j - \ddot{a}_j) + \eta \left[ \frac{d}{d} R_{00}, UA \right] (\ddot{b}_j - \ddot{a}_j)
\] (5)

Equations (3) through (5) represent the position, velocity and acceleration of an arbitrary coupler point specified by parameters \( \xi \) and \( \eta \). These are easily integrated into the computations as discussed in the next section.
II. COMPUTER IMPLEMENTATION

One of the primary objectives in the design of this program was to allow it to be run on a standard Apple II computer with 48k memory and a single disk drive. Another objective was for the analysis and animation to be fast enough to be considered interactive. In order to meet these objectives and to allow for user customization it was decided to use machine language only for the time consuming parts of the program. Figure 4 shows the overall program structure, with the major subprograms outlined. These subprograms are chained together and data is transferred by way of data files on disk. The complete package is contained on a single 5 1/4 in. diskette.

The technique used to increase computational speed in the analysis subprogram RSSR3 was EXPEDITER, a commercially available BASIC compiler. Speed enhancement in the animation subprogram ANIMATE was accomplished by the use of A2-3D2, a commercially available high speed graphics converter and line drawer. These topics and the subprograms are discussed in more detail in subsequent sections.
Figure 4. Program Structure
The BASIC Compiler EXPEDITER

The computer language BASIC is said to be an interpretive language, that is, the source code is acted on directly by the computer in a statement by statement fashion. BASIC is an excellent language for programs that require lots of interaction with the user, but long number crunching computations can become quite slow. In this case it is advantageous to compile, or convert into machine language object code, portions of programs that require little interaction and long computations. One such compiler available for the Apple II and used in the RSSR3 subprogram is EXPEDITER (9). EXPEDITER acts on BASIC source code and creates machine language object code, which greatly enhances speed. This greater speed does not come without some penalty, however, as the increase in speed is tempered by an increase in memory required for the program. In the case of the subprogram RSSR3, a fivefold increase in speed was accompanied by a threefold increase in memory required. Another penalty is that the compiled program consists of virtually unintelligible machine code and cannot be modified except by first changing the source code and re-compiling.

High Speed Animation

Animation is a means of illustrating movement by displaying discrete, stationary pictures at a high enough
rate to trick the eye. Movie projectors and television sets are good examples of how continuous motion can be implied from the rapid projection of still frames. In the case of this program, discrete positions of a spatial mechanism are projected onto the computer screen, along with the stationary reference frame axes, at a fast enough rate to illustrate movement. These positions are obtained from the analysis subprogram RSSR3.

Given that all the information is available for all positions of the mechanism, animation of a spatial mechanism requires three basic sequential steps for each position:

1. Screen erase
2. 3D-to-2D conversion
3. 2D Screen projection

Early attempts at coding this procedure in BASIC using Apple graphics commands proved too slow. One of the objectives of the program was to provide an animation that is fast enough to demonstrate relative motion and allow visualization of all positions. For this reason it was necessary to use a high speed graphics package, the A2-3D2 by Sublogic Corp (10), in the ANIMATE subprogram.

The Sublogic graphics package is a commercially available machine language program that resides in a certain location in memory. CALLing the location in BASIC activates the program and causes it to read an array (at
another specified location in memory) containing 3-D point and line information. Using pre-specified viewpoint information, the routine converts the 3-D data into 2-D lines and points and projects them on the screen on command. By creating (using the POKE command) one array in memory containing point and line information for all positions desired, and then CALLing the subroutine, an extremely rapid animation results. After interpreting the array, A2-3D2 returns control to BASIC.

A2-3D2 has many utilities that are helpful. For example, time delays, no-op, and interpretive jump commands can be placed in the array to change the resulting animation. Another utility that proved important was the "exclusive or" line drawing feature of A2-3D2. When a scene consists of a large number of stationary lines and only a few moving ones, much faster animation can be obtained by drawing over lines to erase them rather than erasing the entire screen. This feature is used in drawing the stick figure representing the mechanism: the same figure is drawn again in the exclusive or mode to erase it and prepare for the next frame. The resulting speed for the RSSR mechanism is in excess of 10 frames per second.

Appendix E contains information about A2-3D2, including a memory map for the subprogram ANIMATE, a description of the array, viewpoint conventions, and a summary of commands.
**Subprogram INPUT2**

INPUT2 is the first subprogram loaded on initial "boot" of the diskette, and prompts the user to enter the mechanism's geometry and motion characteristics. The graphics screen displays the RSSR conventions during this entry phase. In order to prevent bad input data from reaching the analysis program, several "lockouts" or safeguards had to be coded:

1. Check for valid unit vectors
2. Check for right angle between links and unit vectors (see fig.1)
3. Check for a maximum of 71 positions (memory size dictated)

In addition, INPUT2 contains the test for a planar mechanism and if true, the user is asked to enter the coupler point parameters if so desired.

**Subprogram RSSR3**

RSSR3 is the computer implementation of the closed form analytical technique found in Suh and Radcliffe (9). Implementing this technique in BASIC required considerably more code, as BASIC does not have the same powerful subroutine capabilities as FORTRAN. As a result the subprogram had to be compressed to the point where it is
not very easily read (i.e. no comment statements). To further increase speed this subprogram was compiled, and as a result, the maximum analysis time (71 positions) is around 1 min, 10 sec.

Subprogram PRINT

The name print for this subprogram is a little misleading, as it actually performs many utilities, such as printing, plotting, coupler point analysis, and the calculation of some data for the animation subprogram.

The print utility is relatively straightforward, except, as many BASIC programmers realize, attractively formatting a printout is more difficult than in FORTRAN. Figure 5 shows a sample printout.

The plot routine is actually a separate subprogram but since it is called from PRINT, it will be discussed here. When the user desires a plot, he is allowed to choose between a number of mechanism parameters, any of which may be plotted against any other on a two-axis plot. For example, plotting the coupler point x position on the x axis and the coupler point y position on the y axis results in a plot of the coupler curve. Figure 6 illustrates a sample plot. Hardcopy of plots may be obtained if the appropriate printer and graphics "dump" routine are available.
R S S R  
MECHANISM ANALYSIS  
RESULTS  

**PROGRAM INPUTS**
---

**INITIAL MECHANISM CONFIGURATION:**

<table>
<thead>
<tr>
<th>POINT A UNIT VECTOR</th>
<th>POINT B UNIT VECTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>X</strong> = .28</td>
<td><strong>X</strong> = -.28</td>
</tr>
<tr>
<td><strong>Y</strong> = .947</td>
<td><strong>Y</strong> = .947</td>
</tr>
<tr>
<td><strong>Z</strong> = .16</td>
<td><strong>Z</strong> = .16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>POINT A0</th>
<th>POINT B0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>X</strong> = 4</td>
<td><strong>X</strong> = 34</td>
</tr>
<tr>
<td><strong>Y</strong> = 0</td>
<td><strong>Y</strong> = 0</td>
</tr>
<tr>
<td><strong>Z</strong> = 6</td>
<td><strong>Z</strong> = 6</td>
</tr>
</tbody>
</table>

**POINT A1 | POINT B1**

| **X** = 2.86 | **X** = 35.14 |
| **Y** = .807 | **Y** = .807 |
| **Z** = 3.22 | **Z** = 3.22 |

**ANGLE VELOCITY OF INPUT LINK:** 10 RAD/SEC
**ANGULAR ACCELERATION OF INPUT LINK:** 0 RAD/SEC/SEC
**TOTAL TRAVEL OF INPUT LINK:** 40 DEGREES
**INCREMENT:** 2 DEGREES

---

**RESULTS**

* DENOTES NO ASSEMBLY POSSIBLE

<table>
<thead>
<tr>
<th>ANGLE (DEG)</th>
<th>ANGLE BETA (RAD/SEC)</th>
<th>ACCELERATION (RAD/SEC/SEC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>9.74848616</td>
</tr>
<tr>
<td>2</td>
<td>9.74848616</td>
<td>-70.7984385</td>
</tr>
<tr>
<td>4</td>
<td>9.74848616</td>
<td>-68.8143135</td>
</tr>
<tr>
<td>6</td>
<td>9.74848616</td>
<td>-67.2931365</td>
</tr>
<tr>
<td>8</td>
<td>9.74848616</td>
<td>-66.1753935</td>
</tr>
<tr>
<td>10</td>
<td>9.74848616</td>
<td>-65.4122051</td>
</tr>
<tr>
<td>12</td>
<td>9.74848616</td>
<td>-64.9628287</td>
</tr>
<tr>
<td>14</td>
<td>9.74848616</td>
<td>-64.7933153</td>
</tr>
<tr>
<td>16</td>
<td>9.74848616</td>
<td>-64.8746002</td>
</tr>
<tr>
<td>18</td>
<td>9.74848616</td>
<td>-64.9628287</td>
</tr>
<tr>
<td>20</td>
<td>9.74848616</td>
<td>-65.1797773</td>
</tr>
<tr>
<td>22</td>
<td>9.74848616</td>
<td>-65.6878969</td>
</tr>
<tr>
<td>24</td>
<td>9.74848616</td>
<td>-66.3701425</td>
</tr>
<tr>
<td>26</td>
<td>9.74848616</td>
<td>-67.2318374</td>
</tr>
<tr>
<td>28</td>
<td>9.74848616</td>
<td>-68.2314418</td>
</tr>
<tr>
<td>30</td>
<td>9.74848616</td>
<td>-69.3601564</td>
</tr>
<tr>
<td>32</td>
<td>9.74848616</td>
<td>-70.601599</td>
</tr>
<tr>
<td>34</td>
<td>9.74848616</td>
<td>-71.935448</td>
</tr>
<tr>
<td>36</td>
<td>9.74848616</td>
<td>-73.357728</td>
</tr>
<tr>
<td>38</td>
<td>9.74848616</td>
<td>-74.8396957</td>
</tr>
<tr>
<td>40</td>
<td>9.74848616</td>
<td>-76.368716</td>
</tr>
</tbody>
</table>

---

Figure 5. Sample Printout
Figure 6. Sample Plot
PRINT also conducts the coupler analysis if this option was selected. Analysis results are used to calculate the position, velocity and acceleration of an arbitrary coupler point in the case of a planar mechanism.

Finally, if an animation is desired, a section of code in PRINT calculates a lot of the background information for the animation subprogram. Examples of this include properly scaled axes, arrowheads, and labels.

Subprogram ANIMATE

If an animation is desired, the subprogram ANIMATE is called from PRINT, otherwise, control goes back to the input subprogram for another entry. Much of this subprogram is devoted to managing the data arrays which the Sublogic graphics routine will interpret. On entry, the user is shown a view of the mechanism from a default viewpoint, and asked to upgrade this viewpoint to a more satisfactory one. The animation can be viewed from any location in space, and the display is a true perspective view from this viewpoint.

Once the viewpoint is perfected, the user is shown a complete cycle of the mechanism in motion. He is then allowed to adjust speed, select another viewpoint, or go back to the input program before beginning a continuous animation. Once a continuous animation is begun, it can be stopped at any time with the same options.
Different types of RSSR mechanisms can result in different animations. In the case of the crank-rocker configuration, if the input angle is a full circle, the animation will show complete, continuous motion. If only a portion of input link rotation is specified, the animation will show this portion and repeat. In other configurations, if the mechanism cannot be assembled for all specified input rotation, the resulting animation shows only the positions for which the mechanism can be assembled.
III. EXAMPLE: STEERING LINKAGE

Spatial mechanisms in general have not enjoyed the same widespread use in industry as planar ones. Perhaps this is because they are often difficult to visualize and hence, conceive, or because there is little application for them. One application that is relatively common for the RSSR mechanism is the steering linkage in many vehicles.

Figure 6 shows the schematic top view of a vehicle with front wheels turned. This particular turning geometry, called Ackerman steering geometry, provides that the axes of both front wheels intersect the rear wheel axis at one point. All 4 wheels are thus revolving about a single point. This prevents tire scrubbing on tight turns and is particularly important on vehicles with short wheelbases and tight turning radii. The mechanism is a spatial one because the kingpins are both inclined (to give caster) and canted (to provide scrub radius).

The 1978 "Mini Baja" contest winning amphibious vehicle entered by the University of Central Florida had the following vehicle geometry (see fig. 7):

Caster...............................9 deg
Kingpin inclination..............16 deg
Figure 7. Ackerman Steering Geometry
Wheelbase.........................152 cm
Kingpin-Kingpin Distance..........76 cm

From figure 7 the following design equation can be derived:

\[
\frac{1}{\tan \alpha'} + \frac{1}{2} = \frac{1}{\tan \beta'}
\]

Note that \(\alpha'\) and \(\beta'\) are not exact reflections of the mechanism conventions \(\alpha\) and \(\beta\), but should be close enough for the purposes of this analysis. The above equation describes a function generator application for the RSSR mechanism.

Other design considerations necessitated a maximum of 762 mm for the length of links A and B, and for the sake of mechanical advantage, it was decided for that to be the minimum as well. It was therefore only necessary to vary the offset angle until a configuration was found that matched the design equation as well as possible. A lone precision point was chosen for the minimum turning radius, since this is the point where scrubbing would be most critical.

After about 6 iterations with the program, a configuration that closely agreed with the design equation was obtained. The sample printout of figure 5 is the results for this configuration and figure 8 shows a plot of the design equation vs the actual mechanism performance.
Figure 8. Actual VS Ideal Performance
(1) ACTUAL MECHANISM
(2) IDEAL (DESIGN EQUATION)
It should be noted that, while the animation portion of the program was not necessary for this design, viewing it helped give a physical feel for the motion near the critical minimum turning radius.
IV. DISCUSSION AND CONCLUSIONS

Theory, implementation, and application of an interactive computer program for analyzing and visualizing a class of spatial mechanisms have been presented. Designed to run on a popular microcomputer, this program should prove to be an inexpensive and useful tool to the mechanisms designer. At this point a few reflections should be made:

1. Speed in certain areas of the program, while not objectionable, could be improved. Most of the delay is due to saving and loading large data files to and from disk. The binary load and save routine used in the plot routine unfortunately could not be used with the compiler EXPEDITER. As a result, ordinary textfiles had to be utilized for the storage of large numerical arrays, and loading and saving these arrays proved time consuming.

2. The use of subprograms in the program structure provides flexibility to allow customization, and several enhancements could be made. An optimization subprogram could be allowed to interpret the results from the analysis, and feed new data back into it for certain types of synthesis problems. The capability to handle different spatial mechanisms could be added to the package.
Finally, this program was not intended to be totally foolproof. While every attempt was made to "lock out" invalid responses and inputs, a reasonable understanding of the principles of kinematics and space mechanisms is necessary for its successful operation.
APPENDIX A

RSSR ANALYSIS THEORY
Referring again to (8) and figure 1, the initial configuration of the mechanism is fully determined by 4 points and two unit vectors. Complete kinematic analysis can be conducted in closed form by additionally specifying the motion of the input link.

**Position Analysis**

Analysis begins by writing a displacement constraint equation specifying constant length for the coupler:

\[(a - b)^T (a - b) = (a_1 - b_1)^T (a_1 - b_1)\]

where \(a\) is given in terms of the specified input angle \(\alpha\) from

\[a = [R_{\alpha \alpha \alpha}] (a_1 - a_0) + a_0\]

and \(b\) is a function of the unknown output angle \(\beta\)

\[b = [R_{\beta \beta \beta \beta}] (b_1 - b_0) + b_0\]

Recall now that the rotation matrix \([R_{\phi \phi \phi \phi}]\) can be written in the compact form

\[ [R_{\phi \phi \phi \phi}] = [I - Q_{\phi \phi \phi \phi}] \cos \phi + [P_{\phi \phi \phi \phi}] \sin \phi + [O_{\phi \phi \phi \phi}] \]

where

\[ [P_{\phi \phi \phi \phi}] = \begin{bmatrix} \emptyset & -u_z & u_y \\ u_z & \emptyset & -u_x \\ -u_y & u_x & \emptyset \end{bmatrix} \]
and

\[ [Q_u] = \begin{bmatrix}
  u^2_x & u_x u_y & u_x u_z \\
  u_x u_y & u_y^2 & u_y u_z \\
  u_x u_z & u_y u_z & u_z^2 \\
\end{bmatrix} \]

Substituting (2) and (3) into (1) by writing \((a - b)\) as

\[(a - b) = (a - b_0) - [R_{ub}] (b_1 - b_0)\]

and noting that

\[(a_1 - b_1)^T (a_1 - b_1) = (a - b_0)^T (a - b_0) + (b_1 - b_0)^T (b_1 - b_0) - 2(a - b_0) [R_{ub}] (b_1 - b_0)\]

we get

\[ E \cos \beta + F \sin \beta + G = 0 \]  

(4)

where

\[ E = (a - b_0)^T [I - Q_{ub}] (b_1 - b_0) \]
\[ F = (a - b_0)^T [P_{ub}] (b_1 - b_0) \]
\[ G = (a - b_0)^T [Q_{ub}] (b_1 - b_0) + \frac{1}{2} \left\{ (a_1 - b_1)^T (a_1 - b_1) - (a - b_0)^T (a - b_0) - (b_1 - b_0)^T (b_1 - b_0) \right\} \]

Note that (4) contains one unknown, \(\beta\), because \(a\) is known from the specified input angle \(a\) (see equation 2).

Solution of equation (4) yields two possible values of \(\beta\), which is expected as there are two possible mechanism configurations for a given input angle:

\[ \beta_{1,2} = 2 \tan^{-1} \left\{ -\frac{F \pm \sqrt{E^2 + F^2 - G^2}}{G - E} \right\} \]

Subprogram RSSR3 uses the value of \(\beta\) that is closest to the previously calculated one to avoid selecting the wrong
configuration. Subprogram RSSR3 also checks to see if the mechanism can be assembled by noting that when the term \((E^2 + F^2 - G^2)\) is negative, the solution does not exist. Equation (6) is then used to calculate the new position of \(b\) utilizing this newly found \(\beta\) value.

Velocity Analysis

As in many mechanism analyses, position analysis is the most difficult and, once accomplished, velocity and acceleration analyses are relatively straightforward.

We begin by differentiating the constraint equation (1):

\[
(a - b)^T (a - b) = 0.
\] (5)

Recall that

\[
\dot{a} = [W_{\dot{a}ua}] (a - a_0)
\]

where \([W_{\dot{a}ua}]\) is the spatial angular velocity matrix that is related to the rotation matrix by

\[
[W_{\dot{a}ua}] = \frac{d}{dt} [R_{ua}] = \dot{u} [P_{ua}].
\]

Therefore

\[
\dot{a} = \dot{u} [P_{ua}] (a - a_0).
\] (6)

Similarly,

\[
\dot{b} = [W_{\dot{b}ub}] (b - b_0) = \dot{\beta} [P_{ub}] (b - b_0).
\] (7)

Substituting (6) and (7) into (5) we get

\[
\dot{\beta} = \frac{(\dot{a})^T (a - b)}{(a - b)^T [P_{ub}] (b - b_0)}
\] (8)
with \( \dot{\beta} \) known, we can find \( \ddot{b} \) from (7)

**Acceleration Analysis**

Again differentiating the constraint equation,

\[
(\ddot{a} - \ddot{b})^T (a - b) + (\ddot{a} - \ddot{b})^T (\dddot{a} - \dddot{b}) = 0
\]

(9)

where

\[
\ddot{a} = [\ddot{\omega}_{au}] (a - a_0)
\]

\[
\dddot{a} = \left\{ \dddot{\alpha} [P_{ua}] + \dddot{\alpha}^2 [P_{ua}] [P_{ua}] \right\} (a - a_0).
\]

(10)

A similar equation for \( b \) can be written:

\[
\dddot{b} = \left\{ \ddot{\beta} [P_{ub}] + \ddot{\beta}^2 [P_{ub}] [P_{ub}] \right\} (b - b_0).
\]

(11)

Finally substituting (10) and (11) into (9) we get

\[
\dddot{\beta} = \frac{(a-b)^T \left\{ \ddot{\beta}^2 [P_{ub}] [P_{ub}] (b-b_0) \right\} + (\dddot{a} - \dddot{b})^T (\dddot{a} - \dddot{b})}{(a - b)^T [P_{ub}] (b - b_0)}.
\]

And of course \( \dddot{b} \) can be found by substituting \( \dddot{\beta} \) into (11)

Subprogram RSSR3 incorporates these principles to perform a complete position, velocity and acceleration analysis on a given RSSR mechanism (see appendix C for program listings).
APPENDIX B

DISK ORGANIZATION
The following is a catalog of the Apple II diskette:

A 004  HELLO
A 021  RSSR3
A 015  INPUT2
A 039  PRINT
T 002  INPUT FILE
T 123  OUTPUT FILE
*B 003  RBOOT
*B 005  RLOAD
*R 012  HRCG
*B 005  ASCII.SET
*B 005  LANDSCAPE.SET
B 002  BLOAD ARRAY
B 002  BSAVE ARRAY
A 013  PLOT ROUTINE.C
A 068  RSSR3.OBJ
T 002  PLOT DESCRIPTOR
B 027  PLOT DATA
B 034  A2-3D2
B 010  RSSR CONVENTIONS
A 024  ANIMATE
B 003  RSSR SKELETON ARRAY
T 056  ANIMATION FILE

The main subprograms as discussed in the text are easily identified: RSSR3, INPUT2, PRINT, PLOT ROUTINE.C, RSSR3.OBJ, A2-3D2, and ANIMATE. Appendices C and D contain complete program listings for these programs.

Several data files are also contained on the disk. INPUT FILE is the input array from the input program INPUT2. OUTPUT FILE contains the results from the analysis program RSSR3. RSSR CONVENTIONS is the data base that contains the conventions for INPUT2. RSSR SKELETON ARRAY is a portion of the animation array used in ANIMATE. ANIMATION FILE contains the position data, as well as
arrowheads, axes, etc., for the animation program. PLOT DESCRIPTOR contains the parameters that will serve as axis labels in the plot routine, and PLOT DATA is the binary array of plot data created and retrieved by the machine language subroutines BLOAD and BSAVE ARRAY. These programs are listed in appendix D.

Other utilities contained on the disk are used in the plot routine for displaying horizontal and vertical characters on the Apple II screen. These include RBOOT, RLOAD, HRGC, ASCII.SET, and LANDSCAPE.SET.
APPENDIX C

SUBPROGRAMS INPUT2, RSSR3, PRINT, AND ANIMATE
REM SUBPROGRAM INPUT2

LIST 50,820

50 HIMEM: B191
180 DIM UA(2),AO(2),Al(2),UB(2),B8(2),Bl(2),A(2)
120 DIM IN(2,9)
140 D$ = "*: REM CNTRL-D
150 PRINT DS:"LOAD A2-3D2"
152 PRINT DS:"LOAD RSSR CONVENTIONS"
153 HGR
154 CALL 24576
155 CALL 24576
160 INS($) = "X=":INS(1) = "Y=":INS(2) = "Z=
200 PRINT INPUT PIVOT A UNIT VECTOR....
220 FOR I = 0 TO 2: PRINT INS(I): VTAB 23: HTAB 3: INPUT UA(I): NEXT I
225 B = SQR(UA(0) * UA(0) + UA(1) * UA(1) + UA(2) * UA(2))
230 IF B > .95 THEN IF B < 1.01 THEN GOTO 240
235 PRINT "NOT A VALID UNIT VECTOR. REENTER": GOTO 200
240 IF CF% = 1 THEN GOTO 342
260 PRINT "INPUT THE COORDINATES OF PIVOT A....."
280 FOR I = 0 TO 2: PRINT INS(I): VTAB 23: HTAB 3: INPUT AO(I): NEXT I
300 IF CF% = 1 THEN GOTO 342
320 PRINT "INPUT THE INITIAL POSITION OF POINT Al."
341 REM TEST FOR INPUT COMPATIBILITY
342 FOR I = 0 TO 2: A(I) = Al(I) - AO(I):
344 B = 0: FOR I = 0 TO 2: B = B + A(I) * UA(I):
346 IF B > - .95 THEN IF B < 1.01 THEN GOTO 360
348 PRINT "AO AND Al INCOMPATIBLE WITH UA. REENTER": GOTO 200
360 IF CF% = 1 THEN GOTO 760
380 PRINT "INPUT THE ANGULAR VELOCITY OF LINK 2.....": INPUT W2
380 IF CF% = 1 THEN GOTO 760
400 PRINT "INPUT THE ANGULAR ACCEL. OF LINK 2.....": INPUT A2
400 IF CF% = 1 THEN GOTO 760
420 PRINT "INPUT THE TOTAL TRAVEL OF THE INPUT LINK DESIRED.....": INPUT ALPHA
420 IF CF% = 1 THEN GOTO 760
440 PRINT "INPUT THE TRAVEL INCREMENT.....": INPUT DOTALPHA
462 NPT = INT(ALPHA / DOTALPHA)
484 IF NPT > 71 THEN PRINT "REENTER FOR MAXIMUM OF 120 POINTS...": GOTO 680
500 IF CF% = 1 THEN GOTO 760
520 PRINT "WOULD YOU LIKE TO CHANGE ANYTHING": INPUT I1$:
520 IF I1$ = "N" THEN GOTO 760
540 PRINT "WOULD YOU LIKE TO CHANGE ANYTHING ELSE": INPUT I1$: CF% = 0
568 IF LEFIs (11$,1) = "Y" THEN GOTO 848
588 IF LEFIs (11$,1) = "N" THEN GOTO 1280
600 GOTO 748
REM WHICH CHANGE?
CF% = 1
VTAB 24
WHICH CB.ANGE?
I2$ = "UA=1...AO=2...Al=3...UB=4...Bl=6...ANG VEL OF LINK 2=7...ANG A
CELL OF LINK 2=8...TOTAL TRAVEL=9...TRAVEL INCRIMENT=10..."
VTAB 24: HTAB 1: PRINT LEFT$ (I2$,39);I2$ = MID$ (I2$,2) + LEFT$ (I2$,1)
K = PEEK (-16384); IF K < 128 THEN FOR K = 1 TO 75: NEXT K:K = PEEK (0): GOTO 948
INPUT CN
IF CN% > 0 THEN GOTO 1028
GOTO 948
IF CN% < 11 THEN GOTO 1068
GOTO 948
INPUT CN% = 1
IF CN% = 1 THEN GOTO 280
IF CN% = 2 THEN GOTO 260
IF CN% = 3 THEN GOTO 328
IF CN% = 4 THEN GOTO 380
IF CN% = 5 THEN GOTO 440
IF CN% = 6 THEN GOTO 500
IF CN% = 7 THEN GOTO 560
IF CN% = 8 THEN GOTO 620
IF CN% = 9 THEN GOTO 680
IF CN% = 10 THEN GOTO 680
REM TEST FOR PLANAR MECHANISM
IF UA(0) = UB(0) THEN IF UA(1) = UB(1) THEN IF UA(2) = UB(2) THEN GOTO 1340
IF UA(0) = UB(0) THEN IF UA(1) = UB(1) THEN IF UA(2) = UB(2) THEN GOTO 1340
IF UA(0) = UB(0) THEN IF UA(1) = UB(1) THEN IF UA(2) = UB(2) THEN GOTO 1340
IF AO(0) = AO(0) THEN IF AO(1) = AO(1) THEN IF AO(2) = AO(2) THEN GOTO 1340
IF I = 0 TO 2: A(I) = B(I) - AO(I); NEXT I
IF B = 0 THEN GOTO 1440
GOTO 1500
PRINT "YOU HAVE ENTERED A PLANAR MECHANISM..."
PRINT "WOULD YOU LIKE A COUPLER POINT TRACE?": INPUT II$
IF LEFT$ (II$,1) = "N" THEN GOTO 1500
PRINT "ENTER THE COUPLER POINT PARAMETERS": PRINT "ZETA-":
VTAB 23: HTAB 6: INPUT ZETA:
IN(0,0) = UA(I): IN(0,1) = AO(I): NEXT I
IN(8,0) = UB(I): IN(8,1) = Bl(I): NEXT I
FOR I = 0 TO 2: IN(I,0) = UA(I): IN(I,1) = AO(I): IN(I,2) = Al(I): IN(I,3) = U
B(I) = Bl(I): IN(I,4) = B(I): IN(I,5) = Bi(I): NEXT I
IF J = 0 TO 9: PRINT IN(I,J): PRINT IN(2,J): NEXT J
PRINT D$: "CLOSE INPUT FILE"
PRINT D$: "OPEN INPUT FILE"
PRINT D$: "WRITE INPUT FILE"
PRINT D$: "RUN RSSR3.OBJ"
REM SUBPROGRAM RSSR3 SOURCE CODE LISTING

LIST

20   DIM UA(2),AO(2),AJ(2,119),UB(2),BS(2),Bl(2),BJ(2,119),VA(2,119),VB(2,19),
     AA(2,119),AB(2,119),BB(2,119),PM(2,2),QM(2,2),DI(2,2),RM(2,2,1),WM(2,2,0),WD(2,2)
     ,T1(2),T2(2),T3(2),T4(2),T5(2),T6(2),T7(2),T8(2),T9(2),TA(2),TB(2)
40   DIM TC(2),AS(119),VU(2,2),Z(119),T(119),U(2,2),DU(2,2),V(119),DD(119),IN(2,2)

55   DS = "": REM CHR\&-D
60   PRINT DS;"OPEN INPUT FILE": PRINT DS;"READ INPUT FILE"
70   FOR J = 0 TO 9: INPUT IN(J,0): INPUT IN(1,J): INPUT IN(2,J): NEXT J
75   PRINT DS:"CLOSE INPUT FILE"
90   FOR I = 0 TO 2:UA(I) = IN(1,0):AO(I) = IN(1,1):Al(I) = IN(1,2):UB(I) = IN(1,3):B0(I) = IN(1,4):BI(I) = IN(1,5): NEXT I
110  FOR I = 0 TO 8:FOR J = 0 TO 8:PRINT D$;"I!i1(I,J) = A2": NEXT I
170  PRINT D$:"CLOSE INPUT FILE"
220  QM(8,8) • UB(0)
280  01(8,0) • FOR I = 0 TO 8:FOR J = 0 TO 8:PRINT D$;"I!i1(I,J) = A2": NEXT I
340  CV = ATN(1) / 45
360  DU = DO * CV
380  NP = INT(AL / DU)
410  BE = 0
440  FOR J = 0 TO NP
455  AS%\& = J
350  X1 = J = DU
365  IF X1 -(CV * 180) < - .828 THEN GOTO 428
370  IF X1 -(CV * 180) < .810 THEN X1 = CV * 180 - .810
420  FOR I = 0 TO 2:U(I) = UA(I): NEXT I
430  PH = X1
440  GOSUB 4000
460  FOR I = 0 TO 2:UA(I) = U(I): NEXT I
470  X1 = PH
490  FOR I = 0 TO 2:AJ(I,J) = RM(1,0,1) * (Al(0) - AO(0)) + RM(1,1,1) * (Al(1) - AO(1)) + RM(1,2,1) * (Al(2) - AO(2)) + AO(1): NEXT I
520  FOR I = 0 TO 2:PM(I) = AJ(I,1) - AO(I):T3(I) = Aj(I,1) - B0(I): NEXT I
690  FOR I = 0 TO 2:T6(I) = 0: FOR K = 0 TO 2:T6(I) = T6(I) + QI(I,K) * T2(K): NEXT I
EXT K: NEXT I
750  E = 0: FOR I = 0 TO 2:E = E + T3(I) * T6(I): NEXT I
790  FOR I = 0 TO 2:T7(I) = T7(I) + PM(I,K) * T2(K): NEXT I
EXT K: NEXT I
830  G = 1: FOR I = 0 TO 2:G1 = G1 + T(I) * T8(I): NEXT I
950  G2 = 0: FOR I = 0 TO 2:G2 = G2 + T4(I) * T8(I): NEXT I
960  G3 = 0: FOR I = 0 TO 2:G3 = G3 + T3(I) * T8(I): NEXT I
970  G4 = 0: FOR I = 0 TO 2:G4 = G4 + T2(I) * T8(I): NEXT I
1000 G = G1 + 3 * (G2 - G3 - G4)
1800 Y1 = E * E + F * D - G * G
1810 IF YI < 0 THEN GOTO 1140
1820 Y1 = SQRT(Y1)
1830 Z1 = G - E
1840 IF ABS(Z1) < 1E - 10 THEN Z1 = 1E - 10
1850 A1 = 2 * ATN((-F - Y1) / Z1)
1860 XA = 2 * ATN((-F + Y1) / Z1)
1870 T1 = ABS(A1 - BE)
1880 T2 = ABS(XA - BE)
1890 BE = A1
1100 IF T2 < T1 THEN GOTO 1128
1110 GOTO 1130
1120 BE = XA
1130 GOTO 1160
1140 AS(I,J) = 0
1150 GOTO 1970
1160 FOR I = 0 TO 2:U(I) = UB(I): NEXT I:PH = BE
1170 FOR I = 0 TO 2:U(I) = UB(I): NEXT I:PH = BE
1180 GOSUB 4000
1200 FOR I = 0 TO 2:UB(I) = U(I): NEXT I:BE = PH
1220 FOR I = 0 TO 2
1230 BJ(I,J) = RM(I,0,1) * (Bl(0) - B0(0)) + RM(I,1,1) * (Bl(1) - B0(1)) + RM(I,2,1) * (B1(2) - B0(2)) + B0(I)
1240 NEXT I
1250 Z2(J) = BE / CV:Z3(J) = X1 / CV
1300 FOR I = 0 TO 2:U(I) = UA(I): NEXT I:VP = W2
1310 GOSUB 5000
1330 FOR I = 0 TO 2:UA(I) = U(I): NEXT I:W2 = VP
1350 FOR I = 0 TO 2:VA(I,J) = 0: FOR K = 0 TO 2:VA(I,J) = WM(I,K,0) * TS(K) + V
A(I,J): NEXT K: NEXT I
1360 FOR I = 0 TO 2
1390 DU(I) = X0: XV(I) = 0
1400 TB(I) = AJ(I,J) - BJ(I,J)
1410 T9(I,J) = BJ(I,J) - B0(I)
1420 NEXT I
1440 FOR I = 0 TO 2:TA(I) = 0: FOR K = 0 TO 2:TA(I) = TA(I) + PM(I,K) * T9(K): NEXT K: NEXT I
1460 VX = 0: FOR I = 0 TO 2:VX = VX + VA(I,J) * TB(I): NEXT I
1490 TX = 0: FOR I = 0 TO 2:TX = TX + TB(I) * TA(I): NEXT I
1500 DB(J) = VX / TX
1520 FOR I = 0 TO 2:U(I) = UB(I): NEXT I:VP = DB(J)
1530 GOSUB 5000
1540 FOR I = 0 TO 2:U(I) = U(I): NEXT I:DB(J) = VP
1560 FOR I = 0 TO 2:VB(I,J) = 0: FOR K = 0 TO 2:VB(I,J) = WM(I,K,0) * T9(K) + V
B(I,J): NEXT K: NEXT I
1590 FOR I = 0 TO 2:U(I) = UA(I): NEXT I:VP = W2:AP = A2
1600 GOSUB 7000
1610 FOR I = 0 TO 2:UA(I) = U(I): DU(I) = VU(I): NEXT I:W2 = VP:AP = A2
1630 FOR I = 0 TO 2:AA(I,J) = 0: FOR K = 0 TO 2:AA(I,J) = WD(I,K,0) * TS(K) + A
A(I,J): NEXT K: NEXT I
1650 FOR I = 0 TO 2:TB(I) = 0: FOR K = 0 TO 2:TB(I) = T6(I) + QI(I,K) * T9(K): NEXT K: NEXT I
1670 FOR I = 0 TO 2:TB(I) = 0: FOR K = 0 TO 2:TB(I) = T7(I) + PM(I,K) * T9(K): NEXT K: NEXT I
1680 FOR I = 0 TO 2:TC(I) = VA(I,J) - VB(I,J): NEXT I
1720 Z4 = 0: FOR I = 0 TO 2:Z4 = Z4 + AA(I,J) * TB(I): NEXT I
1730 Z5 = 0: FOR I = 0 TO 2:Z5 = Z5 + TB(I) * TS(K): NEXT I
1740 Z6 = 0: FOR I = 0 TO 2:Z6 = Z6 + TC(I) * TC(I): NEXT I
1750 Z7 = 0: FOR I = 0 TO 2:Z7 = Z7 + TB(I) * T7(I): NEXT I
1760 X7 = Z4 + DB(J) * DB(J) * Z5 + Z6
1770 X6 = Z7
1780 DD(J) = X7 / X8
1800 FOR I = 0 TO 2:U(I) = UB(I):VU(I) = XV(I): NEXT I:VP = DB(J):AF = DD(J)
1810 GOSUB 7000
1820 FOR I = 0 TO 2:UB(I) = U(I): XV(I) = VU(I): NEXT I:DB(J) = VP:DD(J) = AF
1840 FOR I = 0 TO 2:AB(I,J) = 0: FOR K = 0 TO 2:AB(I,J) = WD(I,K,0) * T9(K): NEXT K: NEXT I
1900 PRINT "YOU ARE ";"%"; "MPT:" " THE WAY THERE"
1970 NEXT J
1972 PRINT DS;"OPEN OUTPUT FILE"
1974 PRINT D4;"WRITE OUTPUT FILE"
1978 FOR I = 0 TO NP: PRINT BJ(0,I): PRINT BJ(1,I): PRINT BJ(2,I): NEXT I
1980 FOR I = 0 TO NP: PRINT VA(0,I): PRINT VA(1,I): PRINT VA(2,I): NEXT I
1982 FOR I = 0 TO NP: PRINT VB(0,I): PRINT VB(1,I): PRINT VB(2,I): NEXT I
1984 FOR I = 0 TO NP: PRINT AB(0,I): PRINT AB(1,I): PRINT AB(2,I): NEXT I
1986 FOR I = 0 TO NP: PRINT AB(0,I): PRINT AB(1,I): PRINT AB(2,I): NEXT I
1988 FOR I = 0 TO NP: PRINT AB(0,I): NEXT I
40

1990 FOR I = 0 TO NP: PRINT Z3(I): NEXT I
1992 FOR I = 0 TO NP: PRINT Z2(I): NEXT I
1994 FOR I = 0 TO NP: PRINT DB(I): NEXT I
1996 FOR I = 0 TO NP: PRINT DD(I): NEXT I
1998 PRINT 0$: "CLOSE OUTPUT FILE"
2000 PRINT D$: "RUN PRINT"

4000 C = COS (PH)*S = SIN (PH)*V = 1 - C
4040 RM(0,0,1) = U(0)*U(0)*V + C
4050 RM(0,1,1) = U(0)*U(1)*V - U(2)*S
4060 RM(0,2,1) = U(0)*U(2)*V + U(1)*S
4070 RM(1,0,1) = U(0)*U(1)*V + U(2)*S
4080 RM(1,1,1) = U(1)*U(1)*V + C
4090 RM(1,2,1) = U(1)*U(2)*V - U(0)*S
4100 RM(2,0,1) = U(0)*U(2)*V - U(1)*S
4110 RM(2,1,1) = U(1)*U(2)*V + U(0)*S
4120 RM(2,2,1) = U(2)*U(2)*V + C
4130 RETURN

5000 WM(0,0,0) = 0: WM(1,1,0) = 0: WM(2,2,0) = 0
5010 WM(0,0,0) = 0: WM(1,1,0) = 0: WM(2,2,0) = 0
5020 WM(0,1,0) = - U(2) * VP
5030 WM(0,2,0) = U(1) * VP
5040 WM(1,0,0) = - WM(0,1,0)
5050 WM(1,2,0) = - U(0) * VP
5060 WM(2,0,0) = - WM(0,2,0)
5070 WM(2,1,0) = - WM(1,2,0)
5080 RETURN

7000 WD(0,0,0) = (U(0)*U(0) - 1) * VP * VP
7010 WD(0,0,0) = (U(0)*U(0) - 1) * VP * VP
7020 WD(0,1,0) = U(0)*U(1)*VP*VP - VU(2)*VP - U(2)*AP
7030 WD(0,2,0) = U(0)*U(2)*VP*VP + VU(1)*VP + U(1)*AP
7040 WD(1,0,0) = U(0)*U(1)*VP*VP + VU(2)*VP + U(2)*AP
7050 WD(1,1,0) = (U(1)*U(1) - 1) * VP * VP
7060 WD(1,2,0) = U(1)*U(2)*VP*VP - VU(0)*VP - U(0)*AP
7070 WD(2,0,0) = U(2)*U(0)*VP*VP - VU(1)*VP - U(1)*AP
7080 WD(2,1,0) = U(2)*U(1)*VP*VP - VU(0)*VP + U(0)*AP
7090 WD(2,2,0) = (U(2)*U(2) - 1) * VP * VP
7100 RETURN
REM SUBPROGRAM PRINT

LIST

90 REM PROGRAM PRINT
100 DIM AJ(2,71), BJ(2,71), VA(2,71), VB(2,71), AA(2,71), AB(2,71), AS(71), Z3(71), Z2(71), Z1(2), DB(71), DD(71), UA(2), AO(2), Al(2), UB(2), BA(2), Bi(2)
110 DIM IN(2,9), R(2,2), P(2,71), DP(2,71), P2(2,71), S(2), PD(9,71)
115 DIM AN(2), TA(2), BN(2), BT(2), AL(2), BR(2), XM(2), YM(2), ZM(2), X4(2), X5(2), X6(2), X7(2), X8(2), X9(2)
120 DIM Y4(2), Y5(2), Y6(2), Y7(2), Y8(2), Y9(2), Z4(2), Z5(2), Z6(2), Z7(2), Z8(2), Z9(2)
130 REM RETRIEVE VALUES FROM DISK FILES
140 D$ • • • s
150 REM CNTRL-D
160 PRINT D$; OPEN INPUT FILE
170 PRINT D$; READ INPUT FILE
180 FOR J • 8 TO 9
190 INPUT IN(0, J): INPUT IN(1, J): NEXT J
200 PRINT D$; CLOSE INPUT FILE
210 FOR I • 0 TO 2
220 UA(I) • IN(I, S): AO(I) • IN(I, 1): Al(I) • IN(I, 2): UB(I) • IN(I, 3): B1(I) • IN(I, 4): Bl(I) • IN(I, 5): NEXT I
230 DO • IN(0, 6): A2 • IN(0, 7): AL • IN(0, 8): IN(1, 6): ETA • IN(1, 7): ETA • IN(1, 8)
240 NP • INT (AL / DO)
250 PRINT D$; OPEN OUTPUT FILE
260 PRINT D$; READ OUTPUT FILE
270 FOR I • 8 TO NP
280 INPUT AJ(I, 0): INPUT AJ(I, 1):
290 INPUT AJ(I, 2): NEXT I
300 FOR I • 0 TO NP
310 INPUT BJ(I, 0): INPUT BJ(I, 1):
320 INPUT BJ(I, 2): NEXT I
330 FOR I • 0 TO NP
340 INPUT VA(I, 0): INPUT VA(I, 1):
350 INPUT VA(I, 2): NEXT I
360 FOR I • 0 TO NP
370 INPUT VB(I, 0): INPUT VB(I, 1):
380 INPUT VB(I, 2): NEXT I
390 FOR I • 0 TO NP
400 INPUT AA(I, 0): INPUT AA(I, 1):
410 INPUT AA(I, 2): NEXT I
420 FOR I • 0 TO NP
430 INPUT AB(I, 0): INPUT AB(I, 1):
440 INPUT AB(I, 2): NEXT I
450 FOR I • 0 TO NP
460 INPUT AS(I): NEXT I
470 FOR I • 0 TO NP
480 INPUT Z3(I): NEXT I
490 FOR I • 5 TO NP
500 INPUT Z2(I): NEXT I
510 FOR I • 0 TO NP
520 INPUT DB(I): NEXT I
530 FOR I • 0 TO NP
540 INPUT DD(I): NEXT I
550 PRINT D$; CLOSE OUTPUT FILE
560 REM COUPLER POINT PARAMETERS
570 REM COMPUTE THE ROTATION MATRIX
580 FOR I • 0 TO 2
590 U(I) • UA(I): NEXT I
600 GOSUB 14000
610 FOR I • 0 TO 2
620 S(I) • BL(I) - Al(I): NEXT I
630 SU • 0:
640 FOR I • 9 TO 2
650 SU • SU + S(I) * S(I): NEXT I
660 D • SQ(R(SU))
670 ED • ZETA / D: ETA • ETA / D
680 IF ETA = 0 THEN IF ETA = 0 THEN GOTO 1800
690 REM LOOP FOR COUPLER POINTS
700 FOR J • 0 TO NP
710 FOR I • 0 TO 2
720 P(I, J) • AJ(I, J) + ZD * (BJ(I, J) - AJ(I, J)) + ED * (R(I, 0) * (BJ(I, 0) - AJ(I, 0)) + R(I, 1) * (BJ(I, 1) - AJ(I, 1)) + R(I, 2) * (BJ(I, 2) - AJ(I, 2))): NEXT I
730 NEXT J
740 REM VELOCITY OF COUPLER POINT J
750 FOR I • 0 TO 2
760 DP(I, J) • VB(I, J) + ZD * (VB(I, J) - VA(I, J)) + ED * (R(I, 0) * (VB(I, 0) - VA(I, 0)) + R(I, 1) * (VB(I, 1) - VA(I, 1)) + R(I, 2) * (VB(I, 2) - VA(I, 2))): NEXT I
770 NEXT I
780 REM ACCELERATION OF COUPLER POINT J
790 FOR I • 0 TO 2
800 P2(I, J) • AA(I, J) + ZD * (AB(I, J) - AA(I, J)) + ED * (R(I, 0) * (AB(I, 0) - AA(I, 0)) + R(I, 1) * (AB(I, 1) - AA(I, 1)) + R(I, 2) * (AB(I, 2) - AA(I, 2))): NEXT I
810 NEXT J
820 NEXT I
830 REM PRINT, PLOT, OR ANIMATE?
840 PRINT "WOULD YOU LIKE A PRINT OF THE RESULTS": VTAB 23; HTAB 39: INPUT 11$
850 IF LEFTS (11$, 1) • "Y" THEN GOTO 5000
860 IF LEFTS (11$, 1) • "N" THEN PRINT "WOULD YOU LIKE A PLOT OF THE RESULTS"

41
HEX$ (II$) THEN PRINT "WOULD YOU LIKE TO SEE AN ANIMATION":
1100 IF LEFT$ (II$, 1) = "y" THEN GOTO 12000
1140 IF LEFT$ (II$, 1) = "n" THEN PRINT D$; • RUN INPUT2
5040 GOTO 1040
5000 REM PRINT ROUTINE
5020 REM PRINTER IN SLOT 1.AUTO LINE FEED ON, 88 COL.
5040 D$ = • • • REM CNTRL-D • • •
5060 PRINT D$; • PRRT •
5080 POKE 1657.80: • REM AUTHOR'S SETUP ~LY
5100 PRINT •
5120 PRINT •
5130 PRINT •
5140 PRINT • MECHANISM ANALYSIS •
5150 PRINT •
5160 PRINT •
5170 PRINT •
5180 PRINT •
5190 PRINT • PROGRAM INPUTS •
5200 PRINT • INITIAL MECHANISM CONFIGURATION •
5210 PRINT •
5220 PRINT •
5230 PRINT •
5240 PRINT • POINT A UNIT VECTOR POINT B UNIT Vec
5250 PRINT SPC( 14); "X = " UA(0); SPC( 53 - POS (0)); "X = " UB(0)
5270 PRINT SPC( 14); "Z = " UA(2); SPC( 53 - POS (0)); "Z = " UB(2)
5280 PRINT •
5290 PRINT •
5300 PRINT • POINT A8 •
5310 PRINT SPC( 14); "Y = " AO(1); SPC( 53 - POS (0)); "Y = " BO(1)
5320 PRINT SPC( 14); "Z = " AO(2); SPC( 53 - POS (0)); "Z = " BO(2)
5330 PRINT •
5340 PRINT •
5350 PRINT SPC( 14); "X = " Al(0); SPC( 53 - POS (0)); "X = " Bl(0)
5360 PRINT SPC( 14); "Y = " Al(1); SPC( 53 - POS (0)); "Y = " Bl(1)
5370 PRINT SPC( 14); "Z = " Al(2); SPC( 53 - POS (0)); "Z = " Bl(2)
5380 PRINT •
5390 PRINT • ANGULAR VELOCITY OF INPUT LINK •••••• •1w2; • RAD/SEC•
5400 PRINT •
5410 PRINT • TOTAL TRAVEL OF INPUT LINK •••••••••• 1AL; • DEGREES•
5420 PRINT •
5430 IF ZETA = 0 THEN IF ETA = 0 THEN GOTO 5490
5440 PRINT •
5450 PRINT • THESE INPUTS INDICATE A PLANAR MECHANISM. •
5460 PRINT •
5470 PRINT •
5480 PRINT •
5490 PRINT •
5500 PRINT SPC( 36); "RESULTS" •
5510 PRINT SPC( 35); "--------•
5520 PRINT •
5530 PRINT SPC( 25); "* DENOTES NO ASSEMBLY POSSIBLE" •
5540 PRINT •
5550 PRINT •
5560 PRINT •
5570 PRINT •
5580 PRINT •
5590 FOR I = 0 TO NP
5600 IF AS%(I) = 0 THEN PRINT Z3(I); SPC( 21 - POS (0)); "••••
5610 PRINT SPC( 42 - POS (}
43 1•••1 SPC(23 - POS (8)))1 GOTO 5620
PRINT Z3(I): SPC(21 - POS (8)):Z2(I): SPC(42 - POS (0)):DB(I): SPC(23 POS (S)) 1DD(I)
NEXT I ST$ • •POINT Al POSITION• GOSUB 6500
FOR I = 0 TO llP GOSUB 6800
IF Pt = 1 THEN GOTO 5718
PRINT Z3(I): SPC(21 - POS (8)):AJ(0,I)1 SPC(42 - POS (0)):AJ(l,1): SPC(-POS (8)):AJ(2,I)
NEXT I ST$ • •POINT Bl POSITION• GOSUB 6580
FOR I = 0 TO NP GOSUB 6880
IF Fl • 1 THEN GOTO 5788
PRINT Z3(I): SPC(21 - POS (8)):BJ(0,I)1 SPC(42 - POS (0)):BJ(l,1): SPC(-POS (8)):BJ(2,I)
NEXT I ST$ • •POINT Al VELOCITY• GOSUB 6500
FOR I = 8 TO llP GOSUB 6880
IF Fl • 1 THEN GOTO 5870
PRINT Z3(I): SPC(21 - POS (0)):VA(0,I): SPC(42 - POS (0)):VA(l,1):
NEXT I ST$ • •POINT Bl VELOCITY• GOSUB 6500
FOR I = 8 TO llP GOSUB 6800
IF Fl • 1 THEN GOTO 5940
PRINT Z3(I)1 SPC(21 - POS (0)):VB(0,I): SPC(42 - POS (0)):VB(l,1):
NEXT I ST$ • •POINT Al ACCELERATION GOSUB 6500
FOR I = 8 TO NP GOSUB 6800
IF Fl • 1 THEN GOTO 6030
PRINT Z3(I)1 SPC(21 - POS (8)):AA(0,I): SPC(42 - POS (0)):AA(l,1):
NEXT I ST$ • •POINT Bl ACCELERATION• GOSUB 6580
FOR I = 8 TO llP GOSUB 6800
IF Fl • 1 THEN GOTO 6100
PRINT Z3(I): SPC(21 - POS (8)):AB(0,I): SPC(42 - POS (0)):AB(l,1):
NEXT I IP ZETA • 8 THEN IF ETA • 8 THEN GOTO 6990
ST$ • •COUPLER POINT POSITION• GOSUB 6500
FOR I = 8 TO llP GOSUB 6800
IF Fl • 1 THEN GOTO 6190
PRINT Z3(I): SPC(21 - POS (0)):P(0,I)1 SPC(42 - POS (0)):P(l,1)1 SPC(POS (0)):P(2,1)
NEXT I
6250 PRINT Z3(I); SPC(21 - POS(0)); DP(0,I); SPC(42 - POS(0)); DP(1,I); SPC(23 - POS(0)); DP(2,I)
6260 NEXT I
6265 ST$ = "COUPLER POINT ACCELERATION"
6270 GOSUB 6500
6300 FOR I = 0 TO NP
6310 GOSUB 6800
6315 IF F% = 1 THEN GOTO 6330
6320 PRINT Z3(I); SPC(21 - POS(0)); DP(0,I); SPC(42 - POS(0)); DP(1,I); SPC(23 - POS(0)); DP(2,I)
6330 NEXT I
6340 GOTO 6990
6500 PRINT PRINT PRINT
6520 PRINT "ANGLE"; SPC(32); ST$
6540 PRINT "ALPHA";
6560 PRINT "(DEG)"; SPC(20); X; SPC(20); Y; SPC(28); Z
6580 PRINT---------------------------------------
6600 RETURN
6800 F% = 0: IF AS$(1) = 0 THEN PRINT Z3(I); SPC(21 - POS(0));""; SPC(42 - POS(0));""; SPC(23 - POS(0));""; F% = 1: RETURN
6820 RETURN
6990 PRINT D$;"PR"$;
7000 PRINT "END OF ROUTINE"
7020 PRINT "WOULD YOU LIKE PlOTS OF THE RESULTS"; VTAB 23; HTAB 38; INPUT 11$;
GOTO 1080
10000 REM BUILD PLOT DATA FILE
10020 FOR I = 0 TO NP
10040 PD(0, I) = Z3(I)
10060 PD(1, I) = Z2(I)
10080 PD(2, I) = BJ(0, I)
10100 PD(3, I) = BJ(1, I)
10120 PD(4, I) = BJ(2, I)
10140 PD(5, I) = DB(I)
10160 PD(6, I) = DD(I)
10180 PD(7, I) = P(0, I)
10200 PD(8, I) = P(1, I)
10220 PD(9, I) = P(2, I)
10240 NEXT I
11000 REM BUILD PLOT DESCRIPTOR FILE
11020 PRINT D$;"OPEN PLOT DESCRIPTOR"
11040 PRINT D$;"WRITE PLOT DESCRIPTOR"
11060 PRINT 9; REM NO. OF VARIABLES
11080 PRINT 71; REM MAX NO. OF POINTS
11100 PRINT NP; REM ACTUAL NO. OF POINTS
11120 PRINT "INPUT ANGLE ALPHA"
11140 PRINT "OUTPUT ANGLE BETA"
11160 PRINT "POS. OF PT B X"
11180 PRINT "POS. OF PT B Y"
11200 PRINT "POS. OF PT B Z"
11220 PRINT "OUTPUT ANGULAR VEL."
11240 PRINT "OUTPUT ANGULAR ACC."
11260 PRINT "COUPLER PT X POS"
11280 PRINT "COUPLER PT Y POS"
11300 PRINT "COUPLER PT Z POS"
11320 PRINT D$;"CLOSE PLOT DESCRIPTOR"
11340 PRINT D$;"RUN BSSAVE ARRAY"
11360 & SAVE PD;"PLOT DATA"
11380 PRINT D$;"RUN PLOT ROUTINE.C"
11400 END
12000 REM BUILD DATA FOR ANIMATION
12010 REM FIND SHORTEST LINK
12020 FOR I = 0 TO 2iS(I) = Al(I) - AO(I); NEXT I
12030 SU = 0; FOR I = 0 TO 2; SU = SU + S(I) * S(I); NEXT I
12050 LA = SQR(SU)
12080 FOR I = 0 TO 2iS(I) = Bi(I) - B0(I); NEXT I
12100 SU = 0: FOR I = 0 TO 2: SU = SU + S(I) * S(I): NEXT I
12120 LB = SQRT(SU)
12140 LM = D: IF TA < LM THEN LM = TA
12160 IF LB < LM THEN LM = LB
12180 LF = LM / 2: LA = LM / 4
12200 REM CALCULATE THE ARROW BODIES
12220 FOR I = 0 TO 2
12240 AN(I) = AO(I) + UA(I) * LF
12260 TA(I) = AO(I) - UA(I) * LF
12280 BN(I) = BO(I) + UB(I) * LF
12300 BT(I) = BO(I) - UB(I) * LF
12320 NEXT I
12330 AN(2) = - AN(2): TA(2) = - TA(2): BN(2) = - BN(2): BT(2) = - BT(2)
12340 REM CALCULATE ARROWHEADS
12360 K1 = 2 * LF / 3: K2 = LF / 6
12380 REM ARROWHEAD FOR UNIT VECTOR A
12400 U(0) = UA(2): U(1) = UB(2): U(2) = - UA(0)
12420 GOSUB 14000
12440 FOR I = 0 TO 2
12460 AL(I) = AO(I) + UA(I) * K1 + K2 * (R(I,0) * UA(0) + R(I,1) * UA(1) + R(I,2) * UA(2))
12480 AR(I) = AO(I) + UA(I) * K1 - K2 * (R(I,0) * UA(0) + R(I,1) * UA(1) + R(I,2) * UA(2)) * UB(2)
12500 NEXT I
12510 AL(2) = - AL(2): AR(2) = - AR(2)
12520 REM ARROWHEAD FOR UNIT VECTOR B
12540 U(0) = UB(2): U(1) = - UB(2)
12560 GOSUB 14000
12580 FOR I = 0 TO 2
12600 BL(I) = BO(I) + UB(I) * K1 + K2 * (R(I,0) * UB(0) + R(I,1) * UB(1) + R(I,2) * UB(2))
12620 BR(I) = BO(I) + UB(I) * K1 - K2 * (R(I,0) * UB(0) + R(I,1) * UB(1) + R(I,2) * UB(2)) * UB(2)
12640 NEXT I
12650 BL(2) = - BL(2): BR(2) = - BR(2)
12670 REM DETERMINE AXES
12700 IF AO(0) > BO(0) THEN XM = AO(0) + LM
12720 XM = BM + BO(LM + LM)
12740 IF AO(0) > BO(0) THEN YM = AO(0) + LM
12760 YM = BO(LM + LM)
12780 IF AO(2) > BO(2) THEN ZM = AO(2) + LM
12800 ZM = BM + BO(LM + LM)
12820 ES(M) = XM: YM(M) = YM: ZM(M) = ZM
12900 REM CREATE ARROWHEADS AND LABELS
12920 A = .15: B = .3: C = .6
12940 Y4(I) = YM - B: Y5(I) = YM - B: Y6(I) = YM + B: Y7(I) = YM + B: Y8(I) = YM - B: Y9(I) = YM + A
13000 Z4(I) = ZM - B: Z5(I) = ZM - B: Z6(I) = ZM + B: Z7(I) = ZM + B: Z8(I) = ZM + C: Z9(I) = ZM + C: ZA(I) = ZM + C: ZB(I) = ZM + C
13060 PRINT D5: "OPEN ANIMATION FILE": PRINT D5: "WRITE ANIMATION FILE"
13100 FOR I = 0 TO NP: PRINT BJ(I,0): PRINT BJ(I,1): PRINT - BJ(I,2): NEXT I
13120 FOR I = 0 TO NP: PRINT P(I,0): PRINT P(I,1): PRINT - P(I,2): NEXT I
13140 FOR I = 0 TO NP: PRINT AS(I,0): NEXT I
13220 PRINT D5: "CLOSE ANIMATION FILE"
13220 PRINT D$: "RUN ANIMATE"
14000 R(0,0) = U(0) * U(0); R(0,1) = U(0) * U(1) - U(2); R(0,2) = U(0) * U(2) + U(1); R(1,0) = U(0) * U(1) + U(2); R(1,1) = U(1) * U(1)
14020 R(1,2) = U(1) * U(2) - U(0); R(2,0) = U(0) * U(2) - U(1); R(2,1) = U(1) * U(2) + U(0); R(2,2) = U(2) * U(2)
14030 RETURN
REM SUBPROGRAM ANIMATE

LIST

50 LOMEM: 16384
70 HIMEM: 24576
100 REM PROGRAM ANIMATE
120 DIM A%(1,8), B%(1,8), C%(1,8), D%(1,8), E%(1,8), F%(1,8), G%(1,8), H%(1,8), I%(1,8), J%(1,8), K%(1,8), L%(1,8), M%(1,8), N%(1,8), O%(1,8), P%(1,8), Q%(1,8), R%(1,8), S%(1,8), T%(1,8), U%(1,8), V%(1,8), W%(1,8), X%(1,8), Y%(1,8), Z%(1,8)
140 DIM A%(1,8), B%(1,8), C%(1,8), D%(1,8), E%(1,8), F%(1,8), G%(1,8), H%(1,8), I%(1,8), J%(1,8), K%(1,8), L%(1,8), M%(1,8), N%(1,8), O%(1,8), P%(1,8), Q%(1,8), R%(1,8), S%(1,8), T%(1,8), U%(1,8), V%(1,8), W%(1,8), X%(1,8), Y%(1,8), Z%(1,8)
160 DIM A%(1,8), B%(1,8), C%(1,8), D%(1,8), E%(1,8), F%(1,8), G%(1,8), H%(1,8), I%(1,8), J%(1,8), K%(1,8), L%(1,8), M%(1,8), N%(1,8), O%(1,8), P%(1,8), Q%(1,8), R%(1,8), S%(1,8), T%(1,8), U%(1,8), V%(1,8), W%(1,8), X%(1,8), Y%(1,8), Z%(1,8)
180 REM SCALE
200 FOR I = 0 TO 2: A%(I) = IN(I,0): B%(I) = IN(I,1): C%(I) = IN(I,2): D%(I) = IN(I,3): E%(I) = IN(I,4): F%(I) = IN(I,5): G%(I) = IN(I,6): H%(I) = IN(I,7): NEXT I
220 DO = IN(9,0): ETA = IN(9,1): ALPHA = IN(9,2)
240 A(2) = - A(1) - B(2) = - B(1) - A(2) = - A(2) - B(2)
320 NF = INT (ALPHA / DO)
340 SF1 = 250: REM SCALE FACTOR
360 FOR I = 0 TO 2: INPUT X: INPUT Y: INPUT Z: A%(I,0) = X * SF1: A%(I,1) = Y * SF1: A%(I,2) = Z * SF1: NEXT I
380 FOR I = 0 TO 2: INPUT X: INPUT Y: INPUT Z: B%(I,0) = X * SF1: B%(I,1) = Y * SF1: B%(I,2) = Z * SF1: NEXT I
460 FOR I = 0 TO 2: INPUT X: INPUT Y: INPUT Z: P%(I,0) = X * SF1: P%(I,1) = Y * SF1: P%(I,2) = Z * SF1: NEXT I
480 FOR I = 0 TO 2: INPUT X: INPUT Y: INPUT Z: Q%(I,0) = X * SF1: Q%(I,1) = Y * SF1: Q%(I,2) = Z * SF1: NEXT I
520 FOR I = 0 TO 2: INPUT X: INPUT Y: INPUT Z: M%(I,0) = X * SF1: M%(I,1) = Y * SF1: M%(I,2) = Z * SF1: NEXT I
540 FOR I = 0 TO 2: INPUT X: INPUT Y: INPUT Z: N%(I,0) = X * SF1: N%(I,1) = Y * SF1: N%(I,2) = Z * SF1: NEXT I
560 PRINT D$: "CLOSE ANIMATION FILE"
580 PRINT D$: "LOAD RSSR SKELETON ARRAY" AND THE GRAPHICS GENERATOR
600 PRINT D$: "LOAD RSSR SKELETON ARRAY"
620 PRINT D$: "CLOSE ANIMATION FILE"
640 PRINT D$: "OPEN ANIMATION FILE"
660 PRINT D$: "OPEN ANIMATION FILE"
1826 MEM = 33174: FOR I = 0 TO 2:BV4 = INT (TA(I) * SF%): GOSUB 6000: NEXT I
1840 MEM = 33181: FOR I = 0 TO 2:BV4 = INT (AM(I) * SF%): GOSUB 6000: NEXT I
1860 MEM = 33188: FOR I = 0 TO 2:BV4 = INT (AL(I) * SF%): GOSUB 6000: NEXT I
1880 MEM = 33195: FOR I = 0 TO 2:BV4 = INT (AR(I) * SF%): GOSUB 6000: NEXT I
1100 MEM = 33062: FOR I = 0 TO 2:BV4 = INT (AO(I) * SF%): GOSUB 6000: NEXT I
1120 MEM = 33097: FOR I = 0 TO 2:BV4 = INT (BB(I) * SF%): GOSUB 6000: NEXT I
1140 MEM = 33282: FOR I = 0 TO 2:BV4 = INT (BT(I) * SF%): GOSUB 6000: NEXT I
1160 MEM = 33289: FOR I = 0 TO 2:BV4 = INT (BN(I) * SF%): GOSUB 6000: NEXT I
1180 MEM = 33223: FOR I = 0 TO 2:BV4 = INT (BL(I) * SF%): GOSUB 6000: NEXT I
1200 MEM = 33216: FOR I = 0 TO 2:BV4 = INT (BR(I) * SF%): GOSUB 6000: NEXT I
1220 MEM = 33230: FOR I = 0 TO 2:BV4 = INT (X9(I) * SF%): GOSUB 6000: NEXT I
1240 MEM = 33237: FOR I = 0 TO 2:BV4 = INT (XY(I) * SF%): GOSUB 6000: NEXT I
1260 MEM = 33244: FOR I = 0 TO 2:BV4 = INT (X8(I) * SF%): GOSUB 6000: NEXT I
1280 MEM = 33251: FOR I = 0 TO 2:BV4 = INT (X6(I) * SF%): GOSUB 6000: NEXT I
1300 MEM = 33258: FOR I = 0 TO 2:BV4 = INT (X5(I) * SF%): GOSUB 6000: NEXT I
1320 MEM = 33265: FOR I = 0 TO 2:BV4 = INT (XM(I) * SF%): GOSUB 6000: NEXT I
1340 MEM = 33272: FOR I = 0 TO 2:BV4 = INT (X4(I) * SF%): GOSUB 6000: NEXT I
1360 MEM = 33069: FOR I = 0 TO 2:LO% = AJ%(I,0,0):HI% = AJ%(I,1,0): GOSUB 7000: NEXT I
1290 IF ETA = 0 THEN IF ETA = 0 THEN MEM = 33086: FOR I = 0 TO 2:LO% = AJ%(I,0,0):HI% = AJ%(I,1,0): GOSUB 7000: NEXT I: GOTO 1293
1291 MEM = 33087: FOR I = 0 TO 2:LO% = P(I,0,0):HI% = P(I,1,0): GOSUB 7000: NEXT I
1292 MEM = 33088: FOR I = 0 TO 2:LO% = BJ%(I,0,0):HI% = BJ%(I,1,0): GOSUB 7000: NEXT I
1293 MEM = 33089: FOR I = 0 TO 2:LO% = P(I,0,0):HI% = P(I,1,0): GOSUB 7000: NEXT I
1294 IF ETA = 0 THEN IF ETA = 0 THEN MEM = 33090: FOR I = 0 TO 2:LO% = BJ%(I,0,0):HI% = BJ%(I,1,0): GOSUB 7000: NEXT I
1295 MEM = 33091: FOR I = 0 TO 2:LO% = P(I,0,0):HI% = P(I,1,0): GOSUB 7000: NEXT I
1296 IF ETA = 0 THEN GOTO 1335
1300 EY(0) = XM(0) / 3: EY(1) = YM(1) / 4: EY(2) = - 2 * XM(0)
1310 PI = 0:BA = 0:HE = 0
1315 X% = 1
1320 MEM = 33092: FOR I = 0 TO 2:BV4 = INT (EY(I) * SF%): GOSUB 6000: NEXT I
1330 POKE 33036,PI: POKE 33039,BA: POKE 33040,HE
1335 CALL 24576: IF I% = 1 THEN I$ = "N": GOTO 1330
1340 PRINT "EYE POSITION SATISFACTORY (Y%):";I$
1360 IF LEFTS(I$,1) = "Y" THEN GOTO 3000
1360 IF LEFTS(I$,1) = "N" THEN INPUT "NEW EYE POSITION (X,Y,Z)";EY(0),EY(1),EY(2)
1390 EY(2) = - EY(2):X% = 0
1400 INPUT "NEW PITCH, BANK, AND HEADING... (ENTER PSEUDODEGREES; 256=FULL CIRCL E):";PI,BA,HE: PRINT : PRINT : PRINT : PRINT : GOTO 1320
3000 REM THE ANIMATION LOOP
3020 POKE 33027,17: POKE 33028,17
3040 POKE 33040,121: POKE 33059,13
3050 MI = 32356:MF = NP * 70 + 71 + MI
3055 FOR M = MI TO MF: POKE M,17: NEXT M
3060 A = 0:NP:C = 1:BV4 = 0
3120 FOR J = A TO B STEP C
3140 M = 33356 + J * 70
3200 MEM = 33069: FOR I = 0 TO 2:LO% = AJ%(I,0,J):HI% = AJ%(I,1,J): GOSUB 7000: NEXT I
3230 IF ETA = 0 THEN IF ETA = 0 THEN MEM = 33076: FOR I = 0 TO 2:LO% = AJ%(I,0,J):HI% = AJ%(I,1,J): GOSUB 7000: NEXT I: GOTO 3266
3240 MEM = 33077: FOR I = 0 TO 2:LO% = P(I,0,J):HI% = P(I,1,J): GOSUB 7000: NEXT I
3260 IF ETA = 0 THEN GOTO 3370
3270 IF ETA = 0 THEN MEM = 33090: FOR I = 0 TO 2:LO% = BJ%(I,0,J):HI% = BJ%(I,1,J): GOSUB 7000: NEXT I: GOTO 3300
3280 MEM = 33091: FOR I = 0 TO 2:LO% = P(I,0,J):HI% = P(I,1,J): GOSUB 7000: NEXT I
3300 MEM = 33059:1 = 0:BV4 = M - 65856: GOSUB 6000
3320 CALL 34576
3330 FOR I = M + 34 TO M STEP - 1: IF PEER (I) = 121 THEN POKE I,17: GOTO 33
3335  NEXT I
3336  POKE M + 33.27; POKE M + 34,SP$
3340  MEM = 33059: I = 8;BV$ = (M + 35) - 65536: GOSUB 6000
3345  CALL 24576
3350  FOR I = M + 69 TO M + 35 STEP -1: IF PEEK(I) = 121 THEN POKE I,17: GO TO 3380
3355  NEXT I
3360  NEXT J
3365  HGR: REM CLEAR GRAPHICS SCREEN
3370  POKE 33059,26: POKE 33060,43: POKE 33061,8
3375  CALL 24576
3380  REM ANIMATION CALL
3385  PRINT "PRESS ANY KEY TO STOP ANIMATION"
3390  POKE 33059,17; POKE 33060,17; POKE 33061,17
3395  CALL 24576
3400  REM CLEAR GRAPHICS SCREEN
3405  PRINT "PRESS ANY KEY TO STOP ANIMATION"
3410  IF PEEK( - 16384) > 127 THEN GOTO 3580
3415  POKE 33053,12: POKE 33054,81
3420  CALL 24576
3425  IF PEEK( - 16384) > 127 THEN GOTO 3580
3430  GOTO 3480
3435  IF PEEK( - 16384) > 127 THEN GOTO 3580
3440  INPUT "SPEED O.K.? (Y/N)"; I1$: IF I1$ = "N" THEN INPUT "SELECT SPEED. (5 SLOWEST, 8 FASTEST)"; SP$: FOR I = 33390 TO MF STEP 78; POKE I,SP$: NEXT I: GOTO 3480
3445  INPUT "VIEWPOINT O.K.? (Y/N)";I1$: IF I1$ = "N" THEN P4 = 1: POKE 33054 ,8: FOR I = 33055 TO 33057; POKE I,17; NEXT I: POKE 33104,17: POKE 33027,8: POKE 33128,8: GOTO 3290
3450  INPUT "CONTINUE WITH ANIMATION ? (Y/N)"; I1$: IF I1$ = "Y" THEN GOTO 3480
3455  INPUT "TRY A DIFFERENT MECHANISM ? (Y/N)"; I1$: IF I1$ = "Y" THEN PRINT D$: "RUN INPUT2"
3460  END
3465  BL% = BV% / 256
3470  IF BV% < 0 THEN HI% = HI% + 1
3475  LO$ = BV% - HI% * 256
3480  IF BV% > -1 THEN GOSUB 7000: RETURN
3485  IF BV% < -1 THEN GOSUB 7000: RETURN
3490  IF LO% < 256 THEN GOSUB 7000: RETURN
3495  IF HI% = 255 THEN GOSUB 7000: RETURN
3500  IF LO% = 0 THEN GOSUB 7000: RETURN
3505  POKE (MEM + 2 * I) + 1,LO$
3510  POKE (MEM + 2 * I) + 2,HI$
3515  RETURN
]
APPENDIX D

PLOTTING AND BINARY DISK STORAGE PROGRAMS
REM PLOT ROUTINE UNCOMPRESSED...WILL NOT RUN...

LIST

100 REM SSSSSSSSSSSSSSSSSSSSSSSSSSS
110 REM S INITIALIZE $  
120 REM SSSSSSSSSSSSSSSSSSSSSSSSS$  
130 REM * SET LOMEM ABOVE *  
135 REM * GRAPHICS *  
140 LOMEM: 16384  
150 REM * SAVE HIMEM *  
160 HL%= PEEK (115);HH%= PEEK (116)  
170 REM * LOAD HRGC, ASCII.SET *  
180 REM * AND LANDSCAPE.SET *  
190 ONERR GOTO 440  
200 TEXT 1: HOME: HGR ADRS $  
210 PRINT CHR$ (4): BLOAD RBOOT: CALL 520  
220 ADRS = USR (0): HRGC  
230 POKE 216,: REM RESET ONERR  
240 IF ADRS < 0 THEN ADRS = ADRS + 65536  
250 CS = ADRS - 2 * 768: HIMEM = CH  
260 CH = INT (CS / 256): CL = CE - CH * 256  
270 POKE ADRS + 7,CL: POKE ADRS + 8,CH  
280 DS = CHR$ (4): REM CTRL-D  
290 PRINT DS; "BLOAD ASCII.SET,A";CS  
300 PRINT DS; "BLOAD LANDSCAPE.SET,A";CS + 768  
310 CALL ADRS + 3  
320 REM * CTRL CHARACTERS *  
330 CP$ = CHR$ (16): REM CLR PAGE  
340 CL$ = CHR$ (12): REM LOWER CASE  
350 CK$ = CHR$ (11): REM UPPER CASE  
355 CI$ = CHR$ (9): REM INVERSE VIDEO  
360 CO$ = CHR$ (15): REM OPTIONS  
365 CM$ = CHR$ (14): REM NORMAL VIDEO  
370 CS$ = CHR$ (19): REM SHIFT  
380 CA$ = CHR$ (1): REM SELECT CHR SET OR PG 1  
390 REM * PLOT DENSITY *  
400 ID = 196,VD = 160  
410 PRINT CI$  
420 PRINT "AFTER PLOT IS FINISHED PRESS S TO STORE IMAGE OF PLOT OR ANY KEY TO CONTINUE"  
425 PRINT CI$  
430 GOTO 9999  
440 TEXT  
450 PRINT "ERROR IN RLOAD OF RBOOT"  
460 POKE 216,:  
470 END  
1000 REM SSSSSSSSSSSSSSSSSSSSSSSSSSS  
1010 REM S READ PLOT $  
1015 REM S DESCRIPTOR FILE $  
1020 REM SSSSSSSSSSSSSSSSSSSSSSSSSSS  
1030 PRINT DS; "OPEN PLOT DESCRIPTOR"  
1040 PRINT DS; "READ PLOT DESCRIPTOR"  
1050 REM  
1060 REM GET PLOT DATA ARRAY DIMENSIONS.  
1070 INPUT NV$: INPUT EX$  
1080 REM GET ACTUAL NUMBER OF POINTS TO PLOT.  
1090 INPUT OP  
1110 REM * GET DESCRIPTIVE *  
1115 REM * VARIABLE NAMES *  
1120 FOR I = 8 TO NV$  
1130 INPUT HAS$(I)  
1140 NEXT I  
1150 PRINT DS; "CLOSE PLOT DESCRIPTOR"  
1160 RETURN  
2000 REM SSSSSSSSSSSSSSSSSSSSSSSSSSS
REM READ PLOT DATA FILE S
REM INIT MACHINE CODE ROUTINE TO LOAD IN ARRAY.
PRINT DS;"BURN LOAD ARRAY"
REM DIMENSION THE ARRAY
DIM PD(NV%,HX%)
REM AND READ IT IN.
2100 & LOAD PD"PLOT DATA"
REM SSSSSSSSSSSSSSSSSSSSSSS
REM DUTMACHINE CODE ROUTIHE TO LOAD I.N ARRAY.
PRINT D$: BRUN BLOAD ARRAY
REM
REM DIMENSION THE ARRAY
DIM PD(NV%,JX%)  REM AND READ IT IN.
LOAD PD"PLOT DATA"
REM
REM GET X & Y VARIABLES  &
REM SSSSSSSSSSSSSSSSSSSSSSS
REM GET X
REM SSSSSSSSSSSSSSSSSSSSSSS
REM FIND MINS, MAXS & SCALE FACTOR &
YN • PD(Y%,0):YM • PD(Y%,8):XN • PD(X%,0):XM • PD(X%,0)
FOR I • 0 TO OP
IF PD(Y%,I) < YN THEN YN • PD(Y%,I)
IF PD(Y%,I) > YM THEN YM • PD(Y%,I)
IF PD(X%,I) < XN THEN XN • PD(X%,I)
IF PD(X%,I) > XM THEN XM • PD(X%,I)
NEXT I
REM CALCULATE SCALE FAC.*
YF • ABS(YM - YB) / VD
XF • ABS(XM - XN) / HD
REM CALCULATE DELTAS*
YD • ABS(YM - YH) / 4
XD • ABS(XM - XH) / 4
REM FIND VERTICAL LOC.*
YTICK • VD / 41XTICK • HD / 4
FOR I • 8 TO 4
BPLOT 278 - HD,I * YTICK + 4 TO 280 - HD,I * YTICK + 4
BPLOT (279 - HD) + XTICK * I,LX% + 1 TO (279 - HD) + XTICK * I,LX% + 1
NEXT I
REM DRAW AXES*
PRINT CP$: REM CLR PG
BCOLOR = 3
REM DRAW AXES*
HPLT 279 - HD,4 TO 279 - HD,VD + 4
HPLT 279 - HD,LX% TO 279,LX%
REM DRAW TICK MARKS*
XTICK • VD / 4:XTICK • HD / 4
FOR I • 0 TO 4
HPLT 278 - HD,I * XTICK + 4 TO 280 - HD,I * XTICK + 4
HPLT (279 - HD) + XTICK * I,LX% + 1 TO (279 - HD) + XTICK * I,LX% + 1
NEXT I
REM LABEL AXES*
PRINT CAS$:2: REM LANDSCAPE.SET
TA% = INT(VD / 16) - INT(LEN(NAS$(Y%)) / 2)
5160 IF TA$ < 1 THEN TA$ = 1
5170 VTAB TA$; HTAB 1
5180 FOR I = 0 TO LEN (NA$(Y$)) - 1
5190 PRINT MID$(NA$(Y$), LEN (NA$(Y$)) - I, I)
5200 NEXT I
5205 PRINT CA$; 1: REM ASCII.SET
5210 TA$ = INT (HD / 14) - INT (LEN (NA$(X$)) / 2)
5220 IF TA$ < 1 THEN TA$ = 1
5230 PRINT NA$(Y$)
5240 REM * ANNOTATE THE AXIS *
5250 REM * WITH AXIS INCREMENTS *
5260 FOR J = 0 TO 4
5270 YTICK = YN + J * YD
5280 XTICK = XN + J * XD
5290 FOR I = 1 TO OP
5300 YP$ = VD + 4 - INT ((PD(Y$, I) - YN) / YF)
5310 XP$ = INT ((PD(X$, I) - XN) / XF) + 279 - BD
5320 HPLLOT XP$, YP$
5330 NEXT I
5340 RETURN
5350 PRINT YTICK$;
5360 VTAB INT (VD / 8 + .5) + 2
5370 HTAB INT ((279 - HD) / 7 + J * 5 + 1)
5380 PRINT XTICK$
5390 NEXT J
5400 RETURN
5410 PRINT CA$; 1: REM ASCII.SET
5420 TA$ = INT (HD / 14) - INT (LEN (NA$(X$)) / 2)
5430 IF TA$ < 1 THEN TA$ = 1
5440 PRINT NA$(Y$)
5450 REM * ANNOTATE THE AXIS *
5460 REM * WITH AXIS INCREMENTS *
5470 FOR J = 0 TO 4
5480 YTICK = YN + J * YD
5490 XTICK = XN + J * XD
5500 REM * FORMAT YTICK & XTICK *
5510 NUM$ = YTICK$: GOSUB 7000: YTICK$ = NUM$
5520 NUM$ = XTICK$: GOSUB 7000: XTICK$ = NUM$
5530 PRINT INT (VD / 8 - J * 5 + 1)
5540 PRINT YTICK$
5550 NEXT J
7278 REM CONCAT EXPONENT PART AND RETURN.
7280 NUMS = MANS + MID$ (NUMS,1)
7290 RETURN
8000 REM MMMMMMMMMMMMMMMMMMMMM
8610 REM M
8820 REM M GENERAL PURPOSE M
8830 REM M PLOTTING ROUTINE M
8840 REM M
8850 REM M UPDATED: 05/24/82 M
8860 REM M
8870 REM MMMMMMMMMMMMMMMMMM
8880 REM
8130 REM AAAAAAAAAAAAAAAAAAAAAA
8140 REM A READ PLOT A
8150 REM A DESCRIPTOR FILE A
8160 REM AAAAAAAAAAAAAAAAAAAAAA
8170 GOSUB 1900
8180 REM AAAAAAAAAAAAAAAAAAAAAA
8190 REM A READ PLOT DATA FILE A
8200 REM AAAAAAAAAAAAAAAAAAAAAA
8210 GOSUB 2900
8220 REM AAAAAAAAAAAAAAAAAAAAAA
8230 REM A GET X & Y VARIABLES A
8240 REM AAAAAAAAAAAAAAAAAAAAAA
8250 GOSUB 3900
8260 REM AAAAAAAAAAAAAAAAAAAAAA
8270 REM A FIND MINS, MAXS AND A
8280 REM A SCALE FACTOR A
8290 REM AAAAAAAAAAAAAAAAAAAAAA
8300 GOSUB 4900
8310 IF YF = 0 THEN PRINT "THE DEPENDENT VARIABLE IS CONSTANT ",PD(Y%,0)," .":
8315 GOTO 8470
8320 REM AAAAAAAAAAAAAAAAAAAAAA
8330 REM A DRAW & LABEL AXIS A
8340 REM AAAAAAAAAAAAAAAAAAAAAA
8350 GOSUB 5900
8360 REM AAAAAAAAAAAAAAAAAAAAAA
8370 REM A PLOT THE POINTS A
8380 REM AAAAAAAAAAAAAAAAAAAAAA
8390 GOSUB 6900
8400 REM * WAIT FOR KEYPRESS *
8410 GET A$
8420 REM * IF CHR=S THEN STORE *
8430 REM * PLOT ON DISK *
8440 IF A$ < > "E" THEN GOTO 8470
8442 REM DISCONNECT HRGC SO AS TO NOT MESS UP PLOT
8444 PRINT 54,248: POKE 55,253: CALL 1002: PRINT
8450 PRINT D$:"SAVE ",NA$(Y%):".PIC.A$2000,L$2000
8452 REM RECONNECT HRGC
8454 POKE 54,244: POKE 55,143: CALL 1002
8460 REM * CLEAR PAGE *
8470 PRINT CPS$: PRINT
8480 VTAB 12
8490 PRINT CI$: REM INVERSE VIDEO
8500 PRINT "WOULD YOU LIKE ANOTHER PLOT ?"
8505 GET A$: PRINT
8507 PRINT CN$: REM NORMAL VIDEO
8510 IF LEFT$(A$,1) = "N" THEN GOTO 8540
8518 GOTO 8220
8520 REM
8525 REM CLEAN UP & EXIT.
8530 REM
8540 REM
8545 REM CLEAR & KSW FOR NORMAL I/O.
8550 POKE 54,248: POKE 55,253: POKE 56,27: POKE 57,253
8559 REM RECONNECT DOS.
8560 CALL 1002
8610 REM RESET HIMEM
8620 POKE 115,HL$: POKE 116,HH$
8630 REM RETURN TO PROGRAM PRINT
8640 PRINT : TEXT : HOME
8650 PRINT D$:"RUN PRINT"
}
***************

* BLOAD ARRAY *
* W. M. ROOS JR. *
 */ LAST UPDATE: 6/16/82 *

***************

* THIS PROGRAM DOES A BINARY
* LOAD OF ARRAYS PREVIOUSLY SAVED
* WITH THE COMPANION PROGRAM
* "BSAVE ARRAY" AND IS DESIGNED
* TO BE USED FROM WITHIN AN ASOFT
* PROGRAM VIA THE AMPERAND CALL.

* THE PROPER SYNTAX IS
* &LOAD ARRAYNAME "FILENAME"
* WHERE ARRAYNAME IS A VALID
* PREVIOUSLY DIMENSIONED ARRAY
* (ONLY FIRST TWO CHAR. USED AS
* IN ASOFT) AND FILENAME IS THE
* DISK FILE YOU WISH THE ARRAY TO
* BE RETRIEVED FROM.
* NOTE THAT ARRAYNAME IS ONLY
* THE NAME OF THE ARRAY WITH NO
* PARENTHESIS OR DIMENSIONS.

* PREPARATIONS FOR USE:
* BRUN BLOAD ARRAY FROM THE
* CALLING PROGRAM.

* ORG $2E5
* SET UP THE $ VECTOR AND RETURN.

02E5: A9 F9  LDA $BLOADPGM ;START OF THIS PGM.
02E7: 8D F6 03 STA $3F6
02EA: A9 02  LDA $BLOADPGM
02EC: 8D F7 03 STA $3F7
02EF: 60  RTS

*BEGINNING OF DOS BSAVE MSG.

BLOADMSG HEX $D8  ;CR,CTRL-D

02F0: 8D 84  BLOADMSG HEX $D8  ;CR,CTRL-D
02F2: C2 CC CF  ASC "LOAD "
02F5: C1 C4 A0  HEX 00 ;EOL

* EXTERNAL SUBROUTINES:

02F8: 00  ERROR = $D412 ;ASOFT ERROR PROCESSING -

CHGET = $B1 ;ASOFT CHGET S/R CALL -
CHGOT = $B7 ;GETS NEXT SEQUENTIAL CHR
* OR TOKEN - LOADS A-REG FROM LOCN SPECIFIED
* "BSAVE ARRAY" (S80-S89) - CARRY IS CLR'D IF
* CHR IS NUMERIC OTHERWISE SET -
* 2-FLAG SET IF CHR IS 0 (EOL) OR ; (EOS),
* OTHERWISE 2-FLAG CLR'd -
* CHRGOT INCREMENTS TXTPTR BEFORE GETTING
* CHR; CHRGOT LEAVES TXTPTR UNCHANGED.

51
52
53
54
55
56
57
58
59
60
61
62
* HANDLER IF ORGAN ACTIVE
  * OTHERWISE PRINTS ERROR MSG
  * BASED ON CODE IN X-REG.
  * SEE A SofT REF MANUAL FOR CODES.
  * LINPRT = $ED24 ; A SofT PRINT 2-BYTE
  * UNSIGNED INTGER IN X-REG (LSB) & A-REG (MSB).
  * STROUT = $DB3A ; A SofT PRINT STRING
  * POINTED TO BY Y-REG (MSB) &
  * A-REG (LSB); STRING MUST END
  * WITH A ZERO OR A QUOTE.

  03: 1B0E 20 4A FF 106 BLOADPGM JSR SAVE ;SAVE ALL REGS
  02PC: A9 00 107 LDA $0 ;INITIALIZE NAME
  02FE: 85 06 108 STA NAME
  0100: 85 07 109 STA NAME+1
  0102: A9 B6 110 LDA $182 ;CHECK FOR TOKENIZED LOAD
  0104: 20 C0 DE 111 JSR SYNCHR
  0107: 20 B7 00 112 JSR CHRGET ;GET THE ARRAY NAME
  010A: F0 4F 113 BEQ SYNTAX ;AT TXTPTR
  010C: 85 06 114 STA NAME
  010E: 20 B1 00 115 JSR CHRGET
  0111: F0 48 116 BEQ SYNTAX
  0113: C9 22 117 CMP $522 ;CMP TO QUOTE
  0115: P0 08 118 BEQ FINDARRY ;JMP IF ONE CHR ARRAYNAME
  0117: 85 07 119 STA NAME+1 ;ELSE STORE 2ND CHR
  0119: 20 B1 00 120 FINDQUOT JSR CHRGET ;AND MOVE TXTPTR TO QUOTE
  011C: P0 3D 121 BEQ SYNTAX
  011E: C9 22 122 CMP $922 ;CMP TO QUOTE
  0120: D0 F7 123 BNE FINDQUOT
  0122: A0 00 124 FINDARRY LDY $0
  0124: A5 6B 125 LDA ARYTEAB ;ARYPTR <-- ARYTEAB
  0126: 85 06 126 STA ARYTEPR
  0128: A5 6C 127 LDA ARYTEAB+1
  012A: 85 09 128 STA ARYTEPR+1

*EXTERNAL PARAMETER STORAGE
* NAME = $06 ;PGM STORAGE OF
* ARRAY NAME GOTTEN FROM A SofT TEXT.
* ARYTAB = $6B ;A SofT PTR TO
* BEGINNING OF ARRAY SPACE.
* STREND = $6D ;A SofT PTR TO
* END OF NUMERIC STORAGE.
* ARYPTR = $08 ;PGM POINTER USED
* TO INCREMENT THROUGH MEMORY.
* TEMP = $6B ;TEMPORARY STORAGE.
* ARYLEN = $6C ;PGM VARIABLE FOR
* LENGTH OF ARRAY.
* ARYDIM = $6D ;PGM VARIABLE FOR
* NO. OF DIMENSIONS OF ARRAY.
* REMSTK = $F8 ;A SofT STACK PTR SAVED
* BEFORE EACH STATEMENT.
* ERRNUM = $DE ;A SofT ERROR CODE STORE
* PROGRAM BEGINNING


02F9: 20 4A FF 106 BLOADPGM JSR SAVE ;SAVE ALL REGS
02FC: A9 00 107 LDA $0 ;INITIALIZE NAME
02FE: 85 06 108 STA NAME
0300: 85 07 109 STA NAME+1
0302: A9 B6 110 LDA $182 ;CHECK FOR TOKENIZED LOAD
0304: 20 C0 DE 111 JSR SYNCHR
0307: 20 B7 00 112 JSR CHRGET ;GET THE ARRAY NAME
030A: F0 4F 113 BEQ SYNTAX ;AT TXTPTR
030C: 85 06 114 STA NAME
030E: 20 B1 00 115 JSR CHRGET
0311: F0 48 116 BEQ SYNTAX
0313: C9 22 117 CMP $522 ;CMP TO QUOTE
0315: P0 08 118 BEQ FINDARRY ;JMP IF ONE CHR ARRAYNAME
0317: 85 07 119 STA NAME+1 ;ELSE STORE 2ND CHR
0319: 20 B1 00 120 FINDQUOT JSR CHRGET ;AND MOVE TXTPTR TO QUOTE
031C: P0 3D 121 BEQ SYNTAX
031E: C9 22 122 CMP $922 ;CMP TO QUOTE
0320: D0 F7 123 BNE FINDQUOT
0322: A0 00 124 FINDARRY LDY $0
0324: A5 6B 125 LDA ARYTEAB ;ARYPTR <-- ARYTEAB
0326: 85 06 126 STA ARYTEPR
0328: A5 6C 127 LDA ARYTEAB+1
032A: 85 09 128 STA ARYTEPR+1
U3·1:5.i ue

CMPNAME LDA (ARYPTR),Y
0320: C5 06 130 CMP NAME
0330: D0 26 131 BNE NOMATCH
0332: C9 132 INY
0333: B1 08 133 LDA (ARYPTR),Y
0335: C5 07 134 CMP NAME+1
0337: F0 2A 135 BEQ FOUND
136
137 *FALLS THRU TO HERE IF NAME NAME COMPARE FAILS
138
0339: C8 139 OFFSET INY 1GET OFFSET TO
033A: B1 08 140 LDA (ARYPTR),Y 1NEXT ARRAY
033C: 18 141 CLC 1NAME AND ADD
033D: 65 08 142 ADC ARYPTR 1TO ARYPTR
033F: 65 08 143 STA TEMP
0341: CB 144 INY
0342: B1 08 145 LDA (ARYPTR),Y
0344: 65 09 146 ADC ARYPTR+1
0346: 85 09 147 STA ARYPTR+1
0348: 85 08 149 STA ARYPTR
034A: C5 6D 150 CMP STREND 1CHECK FOR
034B: A5 09 151 LDA ARYPTR+1 1END OF
034D: 6E 08 152 BSR STREND+1 1ARRAY STORAGE
0352: B0 07 153 BGE SYNTAX
0354: A0 00 154 LDY $0
0356: F0 D4 155 BEQ CMPNAME 1UNCOND JMP
156
157 *HERE IF 1ST CHR DOESNT MATCH
158
0358: C8 159 NOMATCH INY 1BUMP Y PAST 2ND CHR OF NAME
0359: D0 DE 160 BNE OFFSET 1UNCOND JMP
161
162 *HERE IF ERROR IN STATEMENT SYNTAX
163
035B: A6 F8 164 SYNTAX LDX REMSTK 1RESTORE STACK FROM BEFORE THE &
035D: 9A 165 TXS
035E: A2 10 166 LDX $16 1SYNTAX ERROR CODE
0360: 4C 12 167 JMP ERROR 1REPORT THE ERROR
168
169 *HERE WHEN ARRAY FOUND
170
0363: A2 00 171 FOUND LDX $0 1SAVE ARRAY LENGTH
0364: C9 172 FOUNDLP INY
0366: B1 08 173 LDA (ARYPTR),Y
0368: 95 EC 174 STA ARYLEN,X
036A: EB 175 INX
036C: ED 03 176 CPX #3
036D: 90 F6 177 BLT FOUNDLP
036F: A9 F0 178 LDA $BLOADMSG 1SEND BLOAD TO DOS.
0371: A0 02 179 LDY $BLOADMSG
0373: 20 3A DB 180 JSR STROUT
0376: A9 EA 181 LDA $8EA 1MODIFY CHRGET TO
0378: 85 C0 182 STA $C0 1ALLOW SPACES
037A: 85 C1 183 STA $C1
037C: B5 B1 00 184 SENDFILE JSR CHRGET 1GET FILENAME
037F: F0 11 185 BEQ ENDSND 1AT TXTPTR
0381: C9 22 186 CMP $22 1CMP TO QUOTE
0383: F0 08 187 BEQ FINTXT
0385: 89 80 188 ORA $90 1NEG ASCII
0387: ED FD 189 JSR COUT
0388: 18 190 CLC 1TO DOS
038A: 80 191 BCC SENDFILE 1UNCOND JMP
038D: 20 B1 00 192 FINTXT JSR CHRGET 1FINISH OUT
0390: D0 F8 193 BNE FINTXT 1TXT TO EOL
0392: A9 F0 194 ENDSND LDA $8F0 1RESTORE CHRGET
LDA 1$EP STA $Cl ;SEND ADDR
LDA 1•,• ;SEND COMMA
JSR COUT
LDA 1•A• 1SEND
JSR COUT
LDA ARYPTR
;SEND ADDR VIA LINPRT
LDA ARYPTR+l
JSR LINPRT
TAX
LDA ARYPTR+1
;THE FILE LOAD.
LDX 10 ;CMP ARRAY
LDY 12 1AND DIMENSIONS TO
CMPARYLP
LDA (ARYPTR),Y ;THOSE SAVED BEFORE
CMP ARYPTR,Y ;THE FILE LOAD.
BNE SYNTAX
INX
CPX 13
BLT CMPARYLP
LDY 10 ;NOW RESTORE THE
LDA NAME+1
STA (ARYPTR),Y
JMP RESTORE ;RETURN VIA RESTORE
*Routine to restore REGS.
*NOTE: CODE MUST END BEFORE $3DO
*(DOS VECTORS).

--END ASSEMBLY--

ERRORS: 0

232 BYTES

SYMBOL TABLE - ALPHABETICAL ORDER:

? ARYDIM =$6D ARYLEN =$6C ARYPTR =$08 ARYTAB =$6B
BLOADMSG =$02F0 BLOADPGM =$02F9 CHRGOT =$B1 CHRGOT =$77
CMPARYLP =$0383 CMPNAME =$032C COUT =$FD8E CRUT =$FD8E
ENDSEND =$0392 ? ERRNUM =$DE ERROR =$D412 FINDARRY =$0322
FINDQUOT =$0319 FINTXT =$038D FOUND =$0363 FOUNDLP =$0365
LINPRT =$ED24 NAME =$06 NOMATCH =$0356 OFFSET =$0339
 remnants =$FF7 RESTORE =$FF3F SAVE =$FF4A ?endaddr =$039A
SENDFILE =$037C STREND =$6D STROUT =$D83A SYNCHR =$D83A
SYNTAX =$035B TEMP =$EB

SYMBOL TABLE - NUMERICAL ORDER:

NAME =$06 ARYPTR =$08 ARYTAB =$6B STREND =$6D
CHRGOT =$B1 CHRGOT =$77 ? ERRNUM =$DE TEMP =$EB
ARYLEN =$6D ? ARYDIM =$ED REMSTK =$F8 BLOADMSG =$02F0
BLOADPGM =$02F9 FINDQUOT =$0319 FINDARRY =$0322 CMPNAME =$032C
OFFSET =$0339 FOUNDLP =$0365 SENDFILE =$037C FINTXT =$038D ENDSEND =$0392
? SENDADDR =$039A SYNCHR =$D83A LINPRT =$ED24 CRUT =$FD8E COUT =$FDED
RESTORE =$FF3F SAVE =$FF4A
*THIS PROGRAM DOES A BINARY
*SAVE OF ARRAYS AND IS DESIGNED
*TO BE USED FROM WITHIN AN ASOFT
*PROGRAM VIA THE AMPERSAND CALL.

* THE PROPER SYNTAX IS
* &SAVE ARRAYNAME "FILENAME"
*WHERE ARRAYNAME IS A VALID
*PREVIOUSLY DIMENSIONED ARRAY
*(ONLY FIRST TWO CHARS. USED AS
*IN ASOFT) AND FILENAME IS THE
*DISK FILE YOU WISH THE ARRAY TO
*BE STORED IN.

* NOTE THAT ARRAYNAME IS ONLY
*THE NAME OF THE ARRAY WITH NO
*PARENTHESIS OR DIMENSIONS.

* PREPARATIONS FOR USE:
*BRUN BSAVE ARRAY FROM THE
*CALLING PROGRAM.

ORG $2FA

*SET UP THE & VECTOR AND RETURN.

LDA $BSAVEPGM ;START OF THIS PGM.
STA $3F6
LDA $BSAVEPGM
STA $3F7
RTS

*BEGINNING OF DOS BSAVE MSG.

BSAVEMSG BEX 8084 ASC "SAVE"
BEX 00
EXTERNAL SUBROUTINES:

CHRGOT = $81 ;ASOFT CHRGOT S/R CALL
CHRGOT = $87 ;GETS NEXT SEQUENTIAL CHR

OR TOKEN - LOADS A-REG FROM LOCN SPECIFIED
BY TXTPTR ($88-$89) - CARRY IS CLRD IF
CHR IS NUMERIC OTHERWISE SET -
Z-FLAG GET IF CHR IS O (EOL) OR : (EOS),
OTHERWISE Z-FLAG CLRD -
CHRGOT INCREMENTS TXTPTR BEFORE GETTING
CHR; CHRGOT LEAVES TXTPTR UNCHANGED.
ERROR = $D412 ;ASOFT ERROR PROCESSING -
CHECKS ERRFLG AND JMPS TO
HANDLER IF ONERR ACTIVE
OTHERWISE PRINTS ERROR MSG
61

* BASED ON CODE IN A-REG.
63 • SEE ASOFT REF MANUAL FOR CODES.
64 • LMPTR = $ED24 ;ASOFT PRINT 2-BYTE
65 • UNTIGER IN X-REG (LSB) & A-REG (MSB).
66 • STRONT = $DB3A ;ASOFT PRINT STRING
67 • POINTED TO BY Y-REG (MSB) &
68 • A-REG (LSB); STRING MUST END
69 • WITH A ZERO OR A QUOTE.
70 • COUT = $FDED ;MON CHR OUTPUT ROUTINE
71 • CROUT = $FDEE ;MON ROUTINE TO PRINT CR
72 • SYNCHR = $DECO ;ASOFT SYNTAX CHR CHECK -
73 • CHECKS TO VERIFY TXTPTR POINTS
74 • TO SAME CHR AS THAT IN A-REG.
75 • NORMAL EXIT IS THROUGH CRGET
76 • THEREBY INCREMENTING TXTPTR;
77 • ELSE SYNTAX ERROR GENERATED;
78 • Y-REG IS CLEARED EITHER WAY.
79 • SAVE = $FF4A ;MON SAVE ALL REGS.
80 • RESTORE = $FF3F ;MON RESTORE ALL REGS.
81 • EXTERNAL PARAMETER STORAGE
82 • NAME = $06 ;PGM STORAGE OF
83 • ARRAY NAME GOTTEN FROM ASOFT TEXT.
84 • ARVTAB = $6B ;ASOFT PTR TO
85 • BEGINNING OF ARRAY SPACE.
86 • STREM = $6D ;ASOFT PTR TO
87 • END OF NUMERIC STORAGE.
88 • ARYPTR = $08 ;PGM POINTER USED
89 • TO INCREMENT THROUGH MEMORY.
90 • TEMP = $EB ;TEMPORARY STORAGE.
91 • REMSTK = $FB ;ASOFT STACK PTR SAVED
92 • BEFORE EACH STATEMENT.
93 • ERRNUM = $DE ;ASOFT ERROR CODE STORE
94 • PROGRAM BEGINNING
95
96 030E: 20 4A FF 100 BSAPGM JSR SAVE ;SAVE ALL REGS
97 0311: A9 00 101 LDA $0 ;INITIALIZE NAME
98 0313: BS 06 102 STA NAME
99 0315: BS 07 103 STA NAME+1
00 0317: A9 B7 104 LDA $183 ;CHECK FOR TOKENIZED SAVE
01 0319: 20 C0 DE 105 JSR SYNCHR
02 031C: 20 B7 00 106 JSR CRGET ;GET THE ARRAY NAME
03 031F: F0 4F 107 BEQ SYNTAX ;AT TXTPTR
04 0321: BS 06 108 STA NAME
05 0323: 20 B1 00 109 JSR CRGET
06 0326: F0 48 110 BEQ SYNTAX
07 0328: C9 22 111 CMP $522 ;CMP TO QUOTE
08 032A: F0 0B 112 BEQ FINDARY ;JMP IF ONE CHR ARPAYNAME
09 032C: BS 07 113 STA NAME+1 ;ELSE STORE 2ND CHR
10 032E: 20 B1 00 114 FINDQUF JSR CRGET ;AND MOVE TXTPTO QUOTE
11 0331: F0 3D 115 BEQ SYNTAX
12 0333: C9 22 116 CMP $522 ;CMP TO QUOTE
13 0335: D0 F7 117 BNE FINDARY
14 0337: A0 00 118 FINDARY LDY $0
15 0339: A5 6B 119 LDA ARYTAB ;ARYPTR <-- ARYTAB
16 033B: BS 08 120 STA ARYPTR
17 033D: A5 6C 121 LDA ARYTAB+1
18 033F: BS 09 122 STA ARYPTR+1
19 0341: B1 0B 123 CMPNAME LDA (ARYPTR),Y
20 0343: C5 06 124 CMP NAME
21 0345: D0 26 125 BNE NOMATCH
22 0347: C8 126 INY
23 0348: B1 08 127 LDA (ARYPTR),Y
24 034A: C5 07 128 CMP NAME+1
FALLS THRU TO HERE IF NAME NAME COMPARE FAILS

OFFSET LDA (ARYPTR),Y ;GET OFFSET TO
0352: 65 08 136 CLC ;NAME AND ADD
0354: 85 EB 137 STA TEMP
0356: C8 138 INY
0357: B1 08 139 LDA (ARYPTR),Y
0359: 65 09 140 ADC ARYPTR+1
0361: C5 6D 144 CMP STREND ;CHECK FOR
0363: A5 09 145 LDA ARYPTR+1 ;END OF
0365: E5 6E 146 SBC STREND+1 ;ARRAY STORAGE
0367: B0 07 147 BNE SYNTAX
0369: A0 00 148 LDY $0
036B: FO D4 149 BEQ CMPNAME ;UNCOND JMP
036E: DO DE 154 BNE OFFSET ;UNCOND JMP
0370: A6 F8 158 SYNTAX LDX REMSTK ;RESTORE STACK FROM BEFORE THE 6
0372: 9A 1159 TXS
0373: A2 10 160 LDX #$16 ;SYNTAX ERROR CODE
0375: 4C 22 161 JMP ERROR ;REPORT THE ERROR
0378: A9 05 165 FOUND LDA #$SAVEMSG ;SEND BSAVE TO DOS
037A: A9 03 166 LDY #$>SAVEMSG
037C: 20 3A DB 167 JSR ETRNT
037F: A9 EA 168 LDA #$EA ;MODIFY CHRGET TO
0381: B5 C0 169 STA $C0 ;ALLOW SPACES
0383: B5 C1 170 STA $C1
0385: 20 B1 00 171 SENDFILE JSR CHRGET ;GET FILENAME
0388: 01 11 172 BEQ ENDSEND ;AT TXTPTR
038A: C9 22 173 CMP #$22 ;CMP TO QUOTE
038C: 00 08 174 BEQ FINTXT
038E: 09 80 175 ORA #$80 ;NEG ASCII
0390: 20 ED FD 176 JSR COUT ;AND SEND
0393: 18 177 CLC ;TO DOS
0394: 90 EF 178 BCC SENDFILE ;UNCOND JMP
0396: 20 B1 00 179 FINTXT JSR CHRGET ;FINISH OUT
0399: DF FB 180 BNE FINTXT ;TXT TO EOL
039B: A9 F0 181 ENDSEND LDA #$F0 ;RESTORE CHRGET
039D: B5 C0 182 STA $C0
039F: A9 EF 183 LDA #$EF
03A1: B5 C1 184 STA $C1
03A3: A9 AC 185 sendsend LDA "$", ;SEND COMMA
03A5: 20 ED FD 186 JSR COUT ;SEND A
03A8: A9 C1 187 LDA "$A", ;SEND ADDR VIA LINPRT
03AA: A6 08 189 LDA ARYPTR
03AD: AA 180 190 TAX
03B0: A5 09 191 LDA ARYPTR+1
03B2: 20 24 ED 192 JSR LINPRT
03B5: A9 AC 193 LDA "$", ;SEND ANOTHER COMMA
03B7: 20 ED FD 194 JSR COUT
       *HERE WHEN ARRAY FOUND
       HERE IF 1ST CHR DOESN'T MATCH
       HERE IF ERROR IN STATEMENT SYNTAX
       *HERE WHEN ARRAY FOUND
       HERE IF 1ST CHR DOESN'T MATCH
UJBA: AY CC 295  LDA YL
03BC: 20 ED FD 196  JSR COUT
03BF: A0 02 197  LDD #2
03C1: B1 0B 198  LDA (ARYPTR),Y
03C3: AA 199  TAX
03C4: C0 200  INY
03C5: B1 0B 201  LDA (ARYPTR),Y
03C7: 20 24 ED 202  JSR LINPRT
03CA: 20 8E FD 203  JSR CROUT ;SEND CR
03CD: 4C 3F FF 204  JMP RESTORE ;RETURN VIA RESTORE
205  * ROUTINE TO RESTORE REGS.
206
207  *NOTE: CODE MUST END BEFORE $3DO
208  *(DOS VECTORS).

---END ASSEMBLY---

ERRORS: 0

214 BYTES

SYMBOL TABLE - ALPHABETICAL ORDER:

ARYPTR $08  ARYTAB $6B  BSAVEMSG $0305  BSAVEPGM $030E
CHRGET $B1  CHRGOT $B7  CMPNAME $0341  COUT $FDED
CROUT $FD8E  ENDSEND $039B  ? ERRNUM $DE  ERROR $FDED
FINDARRY $0337  FINDQUOT $032E  FINTXT $0396  FOUND $037B
LINPRT $ED24  NAME $06  NOMATCH $036D  OFFSET $034E
REMSTK $F8  RESTORE $FF3F  SAVE $FF4A  ? SENDADDR $03A3
SENDFILE $0385  STREND $6D  STROUT $DB3A  SYNCHR $DECO
SYNTAX $0370  TEMP $EB

SYMBOL TABLE - NUMERICAL ORDER:

NAME $05  ARYPTR $08  ? ARYTAB $6B  STREND $6D
CHRGET $B1  CHRGOT $B7  ? ERRNUM $DE  TEMP $EB
REMSTK $F8  BSAVEMSG $0305  BSAVEPGM $030E  FINDQUOT $032E
FINDARRY $0337  CMPNAME $0341  OFFSET $034E  NOMATCH $036D
SYNTAX $0370  FOUND $0378  SENDFILE $0385  FINTXT $0396
ENDSEND $039B  ? SENDADDR $03A3  ERROR $D412  STROUT $DB3A
SYNCHR $DECO  LINPRT $ED24  CROUT $FD8E  COUT $FDED
RESTORE $FF3F  SAVE $FF4A
APPENDIX E

MEMORY MAP AND ARRAY FOR SUBPROGRAM ANIMATE

SUBLOGIC DATA SHEET
(Courtesy Sublogic Corp., Champaign, IL)
### MEMORY MAP OF SUBPROGRAM ANIMATE

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 ($0)</td>
<td>Used by Apple</td>
</tr>
<tr>
<td>2048 ($800)</td>
<td>Secondary text page</td>
</tr>
<tr>
<td>3072 ($C00)</td>
<td>BASIC Program Animate</td>
</tr>
<tr>
<td>8191 ($2000)</td>
<td>Graphics page 1 for Animation</td>
</tr>
<tr>
<td>16384 ($4000)</td>
<td>Data for Animation</td>
</tr>
<tr>
<td>24576 ($6000)</td>
<td>A2-3D2 Machine Language Driver</td>
</tr>
<tr>
<td>33019 ($80FB)</td>
<td>Initial Animation Array</td>
</tr>
<tr>
<td>33356 ($824C)</td>
<td>Final Animation Array</td>
</tr>
<tr>
<td>38400 ($9600)</td>
<td>Disk Operating System</td>
</tr>
<tr>
<td>49152 ($C000)</td>
<td>Apple ROMs (BASIC, monitor, etc.)</td>
</tr>
<tr>
<td>65535 ($FFFFFF)</td>
<td>65535</td>
</tr>
</tbody>
</table>

- **Used by Apple**
## INITIAL ANIMATION ARRAY

<table>
<thead>
<tr>
<th>ARRAY LOC</th>
<th>COMMAND</th>
<th>DATA</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>33019 $80FB</td>
<td>07</td>
<td>83 ($53)</td>
<td>Mixed Graphics</td>
</tr>
<tr>
<td>33021 80FD</td>
<td>07</td>
<td>80 ($50)</td>
<td>Color Graphics</td>
</tr>
<tr>
<td>33023 80FF</td>
<td>07</td>
<td>87 ($57)</td>
<td>Hires Mode</td>
</tr>
<tr>
<td>33025 8101</td>
<td>07</td>
<td>84 ($54)</td>
<td>Disp Page 1</td>
</tr>
<tr>
<td>33027 8103</td>
<td>11</td>
<td>11</td>
<td>No Op</td>
</tr>
<tr>
<td>33029 8105</td>
<td>09</td>
<td>01 ($01)</td>
<td>Write Pl</td>
</tr>
<tr>
<td>33031 8107</td>
<td>05</td>
<td>x,y,z,P,B,H</td>
<td>Viewpoint</td>
</tr>
<tr>
<td>33041 8111</td>
<td>OF(F.O.V.) FF,5F,FF,7F,45,2S</td>
<td>Field of View</td>
<td></td>
</tr>
<tr>
<td>33048 8118</td>
<td>14 (SRES) 01</td>
<td>(Set Hi Res 280X192)</td>
<td></td>
</tr>
<tr>
<td>33050 811A</td>
<td>RESERVED FOR</td>
<td>Output Array Control</td>
<td></td>
</tr>
<tr>
<td>33053 811D</td>
<td></td>
<td>Line DWG Control</td>
<td></td>
</tr>
<tr>
<td>33055 811F</td>
<td></td>
<td>Interpretive Jump</td>
<td></td>
</tr>
<tr>
<td>33058 8122</td>
<td>11</td>
<td></td>
<td>No Ops</td>
</tr>
<tr>
<td>33062 8126</td>
<td>SP 01</td>
<td>xyz AO</td>
<td></td>
</tr>
<tr>
<td>33069 812D</td>
<td>CP 02</td>
<td>xyz AJ</td>
<td></td>
</tr>
<tr>
<td>33076 8134</td>
<td>RP 03</td>
<td>xyz P</td>
<td>MECHANISM</td>
</tr>
<tr>
<td>33083 813B</td>
<td>CP 02</td>
<td>xyz BJ</td>
<td>SKELETON</td>
</tr>
<tr>
<td>33090 8142</td>
<td>RP 03</td>
<td>xyz P</td>
<td></td>
</tr>
<tr>
<td>33097 8149</td>
<td>CP 02</td>
<td>xyz B0</td>
<td></td>
</tr>
<tr>
<td>33104 8150</td>
<td>11</td>
<td>No Ops - RESERVED</td>
<td></td>
</tr>
<tr>
<td>33111 8157</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>33118 815E</td>
<td>SP 01</td>
<td>0,0,0</td>
<td></td>
</tr>
<tr>
<td>33125 8165</td>
<td>CP 02</td>
<td>x,y,z ZM</td>
<td>Z = Axis</td>
</tr>
<tr>
<td>33132 816C</td>
<td>RP 03</td>
<td>xyz Z4</td>
<td></td>
</tr>
<tr>
<td>33139 8173</td>
<td>RP 03</td>
<td>xyz Z5</td>
<td></td>
</tr>
<tr>
<td>33146 817A</td>
<td>SP 01</td>
<td>xyz Z6</td>
<td>&quot;Z&quot;</td>
</tr>
<tr>
<td>33153 8181</td>
<td>CP 02</td>
<td>xyz Z7</td>
<td></td>
</tr>
<tr>
<td>33160 8188</td>
<td>CP 02</td>
<td>xyz Z9</td>
<td></td>
</tr>
<tr>
<td>33167 818F</td>
<td>CP 02</td>
<td>xyz Z8</td>
<td></td>
</tr>
<tr>
<td>33174 8196</td>
<td>SP 01</td>
<td>xyz TA</td>
<td></td>
</tr>
<tr>
<td>33181 819D</td>
<td>CP 02</td>
<td>xyz AN</td>
<td></td>
</tr>
<tr>
<td>33188 81A4</td>
<td>RP 03</td>
<td>xyz AL</td>
<td>UNIT VECTOR A</td>
</tr>
<tr>
<td>33195 81AB</td>
<td>RP 03</td>
<td>xyz AR</td>
<td></td>
</tr>
</tbody>
</table>
### INITIAL ANIMATION ARRAY, CONTINUED

<table>
<thead>
<tr>
<th>ARRAY LOC</th>
<th>COMMAND</th>
<th>DATA</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>33202</td>
<td>81B2</td>
<td>SP 01</td>
<td>xyz BT</td>
</tr>
<tr>
<td>33209</td>
<td>81B9</td>
<td>CP 02</td>
<td>xyz BN</td>
</tr>
<tr>
<td>33216</td>
<td>81C0</td>
<td>RP 03</td>
<td>xyz BR</td>
</tr>
<tr>
<td>33223</td>
<td>81C7</td>
<td>RP 03</td>
<td>xyz BL</td>
</tr>
<tr>
<td>33230</td>
<td>81CE</td>
<td>SP 01</td>
<td>xyz X9</td>
</tr>
<tr>
<td>33237</td>
<td>81D5</td>
<td>CP 02</td>
<td>xyz X7</td>
</tr>
<tr>
<td>33244</td>
<td>81DC</td>
<td>SP 01</td>
<td>xyz X8</td>
</tr>
<tr>
<td>33251</td>
<td>81E3</td>
<td>CP 02</td>
<td>xyz X6</td>
</tr>
<tr>
<td>33258</td>
<td>81EA</td>
<td>SP 01</td>
<td>xyz X5</td>
</tr>
<tr>
<td>33265</td>
<td>81F1</td>
<td>CP 02</td>
<td>xyz XM</td>
</tr>
<tr>
<td>33272</td>
<td>81F8</td>
<td>CP 02</td>
<td>xyz X4</td>
</tr>
<tr>
<td>33279</td>
<td>81FF</td>
<td>SP 01</td>
<td>xyz XM</td>
</tr>
<tr>
<td>33286</td>
<td>8206</td>
<td>CP 02</td>
<td>xyz 0,0,0</td>
</tr>
<tr>
<td>33293</td>
<td>820D</td>
<td>CP 02</td>
<td>xyz YM</td>
</tr>
<tr>
<td>33300</td>
<td>8214</td>
<td>RP 03</td>
<td>xyz Y4</td>
</tr>
<tr>
<td>33307</td>
<td>821B</td>
<td>RP 03</td>
<td>xyz Y5</td>
</tr>
<tr>
<td>33314</td>
<td>8222</td>
<td>SP 01</td>
<td>xyz Y9</td>
</tr>
<tr>
<td>33321</td>
<td>8229</td>
<td>CP 02</td>
<td>xyz Y6</td>
</tr>
<tr>
<td>33328</td>
<td>8230</td>
<td>RP 03</td>
<td>xyz Y7</td>
</tr>
<tr>
<td>33335</td>
<td>8237</td>
<td>RP 03</td>
<td>xyz Y8</td>
</tr>
<tr>
<td>33342</td>
<td>823E</td>
<td>79</td>
<td>EOF</td>
</tr>
</tbody>
</table>

### FINAL ANIMATION ARRAY (OUTPUT ARRAY)

<table>
<thead>
<tr>
<th>ARRAY LOCATION</th>
<th>COMMAND</th>
<th>DATA</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>33356 824C (MEM)</td>
<td>06</td>
<td>2-D Lines</td>
<td>2D Lines For To Create Mechanism Skeleton For jth Position</td>
</tr>
<tr>
<td>MEM+26 - MEM+33 27 (81b)</td>
<td></td>
<td></td>
<td>No Ops Pause Command</td>
</tr>
<tr>
<td>MEM+34</td>
<td></td>
<td></td>
<td>2D Lines To Erase jth Position</td>
</tr>
<tr>
<td>MEM+35</td>
<td>06</td>
<td>2-D Lines</td>
<td>Begin 2D Lines For (J + 1)th Position</td>
</tr>
</tbody>
</table>
Product Description

Program Numbers A2-3D1 and A2-3D2

Apple II* Graphics Packages
(3D Graphics, Assembly Language Versions)

A2-3D1

Hardware Requirements
Apple II microcomputer and video monitor.

Memory Requirements
32K minimum.

Product Format
DOS 3.2 standard (muffinable).

Documentation
32 page user’s manual, 84 page technical manual, in a handsome three-ring binder.

A2-3D2

Hardware requirements
Apple II microcomputer and video monitor.

Memory Requirements
48K.

Software Requirements
A2-3D1.

Product Format
DOS 3.3 standard.

Documentation
62 page technical manual.

Shown are examples of the kinds of graphics possible with the A2-3D1 and A2-3D2 programs.

*Apple* is the registered trademark of Apple Computer Inc.
The A2-3D1 and A2-3D2 graphics packages contain sophisticated yet easy-to-use programs for 3D and 2D animation on the Apple II microcomputer. They are designed to accommodate the graphics needs of both new and experienced programmers. With either package you can:

a) View two- or three-dimensional scenes created in the standard XYZ coordinate system...

b) "Zoom" between wide-angle and telephoto fields of view...

c) Select a location in space and a direction of view.

Standard Features of the A2-3D1

The A2-3D1 program ratings "high" in a number of respects: high projection rate, high versatility, high control. And the extensive documentation makes it highly easy to use.

Resolution. You'll have 140 x 192 pixel resolution on the Apple II as your scenes appear on its screen.

Speed. Projection rates of 150 lines per second in the unclipped mode and 100 lines per second in the clipped mode are possible. A 42 millisecond screen erase subroutine is included in the package. A 20-line drawing will be presented at about 5 frames per second in the unclipped mode. If you are doing complex calculations of location and viewing angle, then the program speed will be reduced accordingly.

Versatility. You may specify your own scenes consisting of points and lines by giving XYZ coordinates of points and line end points. Coordinate values within a ±32767 unit range may be entered and stored. Viewing location (XYZ) may be specified within a ±32767 range, and you have full-circle viewing freedom of pitch, bank, and heading.

Control. A set of control programs is provided to give you unlimited utility in your educational, scientific, and game applications. These programs help you to create scenes and allow you to move dynamically through 3D space as follows:

You receive five sample data bases to view during familiarization. After familiarization, you may enter your own data bases as the "data base development" program requests. You may view your creations from different angles at any time during the development, and a "view finder program" helps find scenes as you move freely in 3D space. Scenes can be saved on cassette or disk. Data base relocation instructions let you move data bases in memory and thereby eliminate scene re-entry because of system or program change.

The movement program examples included allow you to change your location and viewing direction dynamically. Your scene will be viewed as you move through 3D space. A special subroutine is included that allows you to orbit your scene.

Utility. Two manuals which are written at different technical levels give all Apple users a quick understanding of access and uses from both assembly language and BASIC language levels.

The Load and Go Manual guides you through an orientation session with the A2-3D1 program. Load the 3D-2D transformer, load the DEVELOP program, and view the scenes waiting for you. The manual will show you how to change location and direction of view one step at a time.

The load and go manual will also guide you through data base development by discussing how to create and enter your own data bases.
The Technical Manual for advanced applications describes the 3D-2D transformer algorithm in depth. Patchpoints and methods for hardware multiply, data output, and data output are described also, as well as special features of the package.

**Features of the A2-3D2**

You must own and be familiar with the A2-3D1 package in order to use the A2-3D2 enhanced graphics package. The concepts of 3D data bases, viewer location and rotation, and display file creation and interpretation are all described in the A2-3D1 technical manual.

The A2-3D2 graphics package contains all of the features found in the A2-3D1 (listed above). It also has a number of new features not available in the A2-3D1 package. These include:

- Color lines and high-resolution (280 x 192) line generation that is nearly as fast as the generation of white low-res lines.
- Slightly faster 2D line drawing and erasing.
- Range handling. Data bases can go right to the edge of the world and lines can start at any point in space and run to any other point (no line length restrictions).
- Trig, multiply, divide, erase, point plot, line draw, and other routines to aid in overall simulation speed.
- Independent object manipulation that allows you to "instance" an object definition anywhere in space. Instance nesting is supported so the user can define objects that remain in other objects' reference and even move around in them.
- Commands to aid in debugging and display control are provided. Skip (to skip over no-longer-used elements) and pause (to put a wait in the display file) are provided.

The most obvious new feature of the A2-3D2 is the color and high-res line generation. White-lined objects take on a new look of precision when drawn in 280 x 192 high resolution. Colors available in lower resolution include white, green, violet, blue, and orange/red.

Independent object manipulation can be used to manipulate objects on an individual basis. It also allows you to create a large number of occurrences of a single object (such as putting 25 houses on a street by calling for the same house definition), and lets you give an object moving parts (such as propellers on airplanes, wheels on cars, etc.). In addition, this feature allows elements of an overall design to be grouped together.

The independent object feature even allows you to design a number of items (such as walls of a house) in two dimensions (where designing is easy), and finally assemble the flat surfaces into a composite 3D design by simply rotating the design planes into their proper positions. Independent object files can be used to build libraries of symbols, fonts, and shading patterns.

**Special Features of the A2-3D1**

- An array generating feature lets you generate an output array of line start and end points instead of plotting on the Apple screen. This array will let you use the program with future graphic output devices.
- A "zero page restore" feature leaves all of your zero page variables intact after subroutine exit.
- The page control feature allows selective page erase, display, and draw for ping-ponging between screens. This permits smooth animation.
- The selective erase feature allows movement of objects without erasing the full screen.
- A variable field of view feature lets you adjust your field of view and "zoom in" on objects in a camera-like fashion.

A special demonstration program is included in the A2-3D1 package.
<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>A2-3D1</th>
<th>A2-3D2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interpretative Functions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pure Point (140 x 192)</td>
<td>0.x,y,z</td>
<td>0.x,y.z</td>
</tr>
<tr>
<td>Start Point (140 x 192)</td>
<td>1.x,y,z</td>
<td>1.x,y.z</td>
</tr>
<tr>
<td>Continue Point (140 x 192)</td>
<td>2.x,y,z</td>
<td>2.x,y.z</td>
</tr>
<tr>
<td>Ray Point (140 x 192)</td>
<td>3.x,y,z</td>
<td>3.x,y.z</td>
</tr>
<tr>
<td>Clipper Control</td>
<td>4.on/off</td>
<td>4.on/off</td>
</tr>
<tr>
<td>Viewer Position D.P.</td>
<td>5.x,y,z,p,b,h</td>
<td>5.x,y,z,p,b,h</td>
</tr>
<tr>
<td>Pseudodegrees</td>
<td>6.x,y,x,y</td>
<td>6.x,y,x,y</td>
</tr>
<tr>
<td>Draw 2D Line on Screen</td>
<td>7.code</td>
<td>7.code</td>
</tr>
<tr>
<td>Display Screen Select</td>
<td>8.code</td>
<td>8.code</td>
</tr>
<tr>
<td>Erase Screen / Fill Screen</td>
<td>9.code</td>
<td>9.code</td>
</tr>
<tr>
<td>Plot a 2D White Point</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interprettive Jump</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Set Line Drawing Mode</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Turn on Output Array</td>
<td>10.x,y</td>
<td>10.x,y</td>
</tr>
<tr>
<td>Screen Size Select</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Field of View Select</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Easy Initialize</td>
<td>11.adrlsb.adrmsb</td>
<td>11.adrlsb.adrmsb</td>
</tr>
<tr>
<td>No Operation</td>
<td>12.mode</td>
<td>12.mode</td>
</tr>
<tr>
<td>Set Color Mode</td>
<td>13.adrlsb.adrmsb</td>
<td>13.adrlsb.adrmsb</td>
</tr>
<tr>
<td>Independent Object Call</td>
<td>14.w,h,cx, cy</td>
<td>14.w,h,cx, cy</td>
</tr>
<tr>
<td>Set Resolution</td>
<td>15.axr, ayr, azr</td>
<td>15.axr, ayr, azr</td>
</tr>
<tr>
<td>Hi-Res (280 x 192) Line 2D</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>Hi-Res Bias</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hi-Res (x = 256 Limited) Line 2D</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>Hi-Res (280 x 192) Point Plot 2D</td>
<td>18</td>
<td>-</td>
</tr>
<tr>
<td>Hi-Res (x = 256 Limited) Point Plot 2D</td>
<td>19</td>
<td>-</td>
</tr>
<tr>
<td>Skip Segment</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pause for n/5ths of a Second</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Set 3D to 3D Array Gen Address</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>End of File</td>
<td>20.x,y,z,p,b,h</td>
<td>20.x,y,z,p,b,h</td>
</tr>
<tr>
<td>Callable Functions</td>
<td>21.x,y,x,y</td>
<td>21.x,y,x,y</td>
</tr>
<tr>
<td>Sine/Cosine Calls</td>
<td>22.x,y,x,y</td>
<td>22.x,y,x,y</td>
</tr>
<tr>
<td>Multiply (SP and DP)</td>
<td>23.x,y,x,y</td>
<td>23.x,y,x,y</td>
</tr>
<tr>
<td>Divide (DP)</td>
<td>24.x,y,x,y</td>
<td>24.x,y,x,y</td>
</tr>
<tr>
<td>Erase (hi/low page)</td>
<td>25.x,y,x,y</td>
<td>25.x,y,x,y</td>
</tr>
<tr>
<td>Hi-Res Point Plot</td>
<td>26.size, status</td>
<td>26.size, status</td>
</tr>
<tr>
<td>Color Point Plot</td>
<td>27.time</td>
<td>27.time</td>
</tr>
<tr>
<td>Color Line Draw</td>
<td>28.adrlsb.adrmsb</td>
<td>28.adrlsb.adrmsb</td>
</tr>
<tr>
<td>Color Line Draw</td>
<td>29.status</td>
<td>29.status</td>
</tr>
<tr>
<td>Set Display Resolution</td>
<td>30.x,y,z,p,b,h</td>
<td>30.x,y,z,p,b,h</td>
</tr>
<tr>
<td>General Features</td>
<td>31.x,y,z,p,b,h</td>
<td>31.x,y,z,p,b,h</td>
</tr>
<tr>
<td>Initialize Input Buffer Ptr.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Line Length Limit</td>
<td>32767 max</td>
<td>113508 max</td>
</tr>
<tr>
<td>World Movement</td>
<td>limited by overflow</td>
<td>unlimited**</td>
</tr>
<tr>
<td>Program Location(s)</td>
<td>2048 x 24576</td>
<td>24576</td>
</tr>
<tr>
<td>Program Length</td>
<td>4864 bytes</td>
<td>8443 bytes</td>
</tr>
</tbody>
</table>

*Distance from -32767.32767.32767 to 32767.32767.32767.
**As long as the value of -32768 is avoided in data bases and eye position.

Ordering Information
See your dealer or order directly from SubLOGIC.
The A2-3D1 with A2-3D2 Enhancement is $84.90 on disk. You may update to A2-3D2 at any time. Contact SubLOGIC for details.
Shipping weight of the packages is approximately five pounds.
Shipping charges: US and Canada add $6.50 for first class mail. $3.00 for UPS. $4.50 for COD (UPS) orders. Foreign add $20.00 ($27.00 Australia) for airmail. Illinois residents add 5% sales tax. MasterCard and Visa accepted.

© 1979 SUBLOGIC COMMUNICATIONS CORPORATION
P.O. BOX 461, CHAMPAIGN, ILL. 61820
217-359-8482

The engineering and graphics experts, opening a new era in computer simulation.
BIBLIOGRAPHY


