Interactive Computer Aided Design and Animation of Spatial Mechanisms

Emmet B. Peter

University of Central Florida

Part of the Engineering Commons
Find similar works at: https://stars.library.ucf.edu/rtd
University of Central Florida Libraries http://library.ucf.edu

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 STARS@ucf.edu.

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

exclusive or: 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-Kinsyn 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 (5) 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 A1 and B1, kinematic analysis of the coupler point is straightforward. For the jth position of the planar case,

\[ \hat{\zeta} = \text{direction}(b_j - a_j) \]
\[ \hat{n}_j = \hat{\zeta} \text{rotated 90 deg about } UA \text{ (or UB)} \]

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

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

Thus

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

The coupler point \( P \) can be written as the sum of 3 vectors:

\[ P_j = a_j + \zeta \hat{\zeta}_j + \hat{n}_j \quad (2) \]

Substituting (1) for \( n_j \) in (2)

\[ P_j = a_j + \zeta \hat{\zeta}_j + \hat{n}[R_{90,UA}] \hat{\zeta} \]
Figure 3. Testing For Planar Inputs
But \( \hat{\zeta}_j = \) direction \((\mathbf{b}_j - \mathbf{a}_j)\)

or \( \hat{\zeta}_j = \frac{(\mathbf{b}_j - \mathbf{a}_j)}{|\mathbf{b}_j - \mathbf{a}_j|} \)

Note that \( |\mathbf{b}_j - \mathbf{a}_j| = \)constant = length of coupler \( d \), thus we may write

\[
\mathbf{p}_j = \mathbf{a}_j + \frac{\zeta}{d} (\mathbf{b}_j - \mathbf{a}_j) + \frac{\eta}{d} \left[ R_{u_A} \right] (\mathbf{b}_j - \mathbf{a}_j) \tag{3}
\]

Taking derivatives to obtain velocity and acceleration,

\[
\dot{\mathbf{p}}_j = \dot{\mathbf{a}}_j + \frac{\zeta}{d} (\dot{\mathbf{b}}_j - \dot{\mathbf{a}}_j) + \frac{\eta}{d} \left[ R_{u_A} \right] (\dot{\mathbf{b}}_j - \dot{\mathbf{a}}_j) \tag{4}
\]

\[
\ddot{\mathbf{p}}_j = \ddot{\mathbf{a}}_j + \frac{\zeta}{d} (\ddot{\mathbf{b}}_j - \ddot{\mathbf{a}}_j) + \frac{\eta}{d} \left[ R_{u_A} \right] (\ddot{\mathbf{b}}_j - \ddot{\mathbf{a}}_j) \tag{5}
\]

Equations (3) through (5) represent the position, velocity and acceleration of an arbitrary coupler point specified by parameters \( \zeta \) 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 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. 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.
### Program Inputs

**Initial Mechanism Configuration:**

<table>
<thead>
<tr>
<th>Angle</th>
<th>Point A Unit Vector</th>
<th>Point B Unit Vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X = .28</td>
<td>X = -.28</td>
</tr>
<tr>
<td></td>
<td>Y = .947</td>
<td>Y = .947</td>
</tr>
<tr>
<td></td>
<td>Z = .16</td>
<td>Z = .16</td>
</tr>
<tr>
<td>2</td>
<td>X = 2.86</td>
<td>X = 3.54</td>
</tr>
<tr>
<td></td>
<td>Y = .807</td>
<td>Y = .807</td>
</tr>
<tr>
<td></td>
<td>Z = 3.22</td>
<td>Z = 3.22</td>
</tr>
</tbody>
</table>

**Angular 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

<table>
<thead>
<tr>
<th>Angle Alpha (Deg)</th>
<th>Position (Deg)</th>
<th>Angle Beta (Deg)</th>
<th>Velocity (RAD/SEC)</th>
<th>Acceleration (RAD/SEC/SEC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>18</td>
<td>9.74840616</td>
<td>-73.3188417</td>
</tr>
<tr>
<td>2</td>
<td>1.97471382</td>
<td>18</td>
<td>9.74840616</td>
<td>-70.7984385</td>
</tr>
<tr>
<td>4</td>
<td>3.8999628</td>
<td>18</td>
<td>9.584657</td>
<td>-68.8143135</td>
</tr>
<tr>
<td>6</td>
<td>5.77713579</td>
<td>18</td>
<td>9.2678132</td>
<td>-67.2931365</td>
</tr>
<tr>
<td>8</td>
<td>7.68729763</td>
<td>18</td>
<td>9.839662</td>
<td>-66.1753935</td>
</tr>
<tr>
<td>10</td>
<td>9.39123149</td>
<td>18</td>
<td>8.80419584</td>
<td>-65.4122051</td>
</tr>
<tr>
<td>12</td>
<td>11.1294719</td>
<td>18</td>
<td>8.57653755</td>
<td>-64.9629287</td>
</tr>
<tr>
<td>14</td>
<td>12.8223345</td>
<td>18</td>
<td>8.34995697</td>
<td>-64.7933153</td>
</tr>
<tr>
<td>16</td>
<td>14.46994</td>
<td>18</td>
<td>8.12502859</td>
<td>-63.8746682</td>
</tr>
<tr>
<td>18</td>
<td>16.8722327</td>
<td>18</td>
<td>7.8964289</td>
<td>-63.1797773</td>
</tr>
<tr>
<td>20</td>
<td>17.6290813</td>
<td>18</td>
<td>7.66798762</td>
<td>-63.5878969</td>
</tr>
<tr>
<td>22</td>
<td>19.1398917</td>
<td>18</td>
<td>7.43729577</td>
<td>-63.3781425</td>
</tr>
<tr>
<td>24</td>
<td>20.684234</td>
<td>18</td>
<td>7.28390858</td>
<td>-62.2136374</td>
</tr>
<tr>
<td>26</td>
<td>22.8220816</td>
<td>18</td>
<td>6.96745228</td>
<td>-61.2314418</td>
</tr>
<tr>
<td>28</td>
<td>23.3912953</td>
<td>18</td>
<td>6.72728095</td>
<td>-60.3681564</td>
</tr>
<tr>
<td>30</td>
<td>24.7134197</td>
<td>18</td>
<td>6.48281446</td>
<td>-69.801599</td>
</tr>
<tr>
<td>32</td>
<td>25.9856868</td>
<td>18</td>
<td>6.2392751</td>
<td>-71.9395448</td>
</tr>
<tr>
<td>34</td>
<td>27.2675565</td>
<td>18</td>
<td>5.98231345</td>
<td>-73.357728</td>
</tr>
<tr>
<td>36</td>
<td>28.378281</td>
<td>18</td>
<td>5.72147469</td>
<td>-74.8396957</td>
</tr>
<tr>
<td>38</td>
<td>29.496742</td>
<td>18</td>
<td>5.45746347</td>
<td>-76.368716</td>
</tr>
<tr>
<td>40</td>
<td>30.5618825</td>
<td>18</td>
<td>5.18866383</td>
<td>-77.9277388</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

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

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

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

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

\[R_{\phi u} = [I - Q] \cos \phi + [P] \sin \phi + [Q]\]

where

\[P_{\phi u} = \begin{bmatrix}
\phi & -u_z & u_y \\
u_z & \phi & -u_x \\
-u_y & u_x & \phi
\end{bmatrix}\]
Substituting (2) and (3) into (1) by writing \((a - b)\) as
\[ \begin{align*}
(a - b) &= (a - b_0) - [R_{\beta_{ub}}](b_1 - b_0)
\end{align*} \]
and noting that
\[ \begin{align*}
(a_1 - b_1)^T (a_1 - b_1) &= (a - b_0)^T (a - b_0) \\
+ (b_1 - b_0)^T (b_1 - b_0) - 2(a - b_0)[R_{\beta_{ub}}](b_1 - b_0)
\end{align*} \]
we get
\[ E \cos \beta + F \sin \beta + G = 0 \] (4)
where
\[ \begin{align*}
E &= (a - b_0)^T [I - Q_{ub}] (b_1 - b_0) \\
F &= (a - b_0)^T [P_{ub}] (b_1 - b_0) \\
G &= (a - b_0)^T [Q_{ub}] (b_1 - b_0) + \frac{1}{2} ((a_1 - b_1)^T (a_1 - b_1) - (a - b_0)^T (a - b_0) - (b_1 - b_0)^T(b_1 - b_0)).
\end{align*} \]
Note that (4) contains one unknown, \(\beta\), because \(a\) is known from the specified input angle \(\alpha\) (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):

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

Recall that

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

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

\[[W_{\text{qua}}]] = \frac{d}{dt} [R_{\text{qua}}] = \dot{\omega} [P_{\text{ua}}].\]

Therefore

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

Similarly,

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

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

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

**Acceleration Analysis**

Again differentiating the constraint equation,

$$(\ddot{a} - \ddot{b})^T (a - b) + (\dddot{a} - \dddot{b})^T (\dot{a} - \dot{b}) = 0 \quad (9)$$

where

$$\ddot{a} = [\ddot{W}_{\dot{a}u}] (a - a_0)$$
$$\dddot{a} = \{\dddot{a}[P_{ua}] + \dddot{a}^2[P_{ua}] [P_{ua}] \} (a - a_0). \quad (10)$$

A similar equation for $b$ can be written:

$$\ddot{b} = \{\ddot{b}[P_{ub}] + \dddot{b}^2[P_{ub}] [P_{ub}] \} (b - b_0). \quad (11)$$

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

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

And of course $\dddot{b}$ can be found by substituting $\dddot{b}$ 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, HRCG, ASCII.SET, and LANDSCAPE.SET.
APPENDIX C

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

LIST 50,820

50 HIMEM: B191
100 DIM UA(2), AO(2), AI(2), UB(2), BS(2), BI(2), A(2)
120 DIM IN(2, 9)
140 D$: "REM GNUCTRL-D"
150 PRINT D$: "LOAD RSSL CONVENTIONS"
154 CALL 24576
155 CALL 24576
160 INS$(0) = "X"; INS$(1) = "Y"; INS$(2) = "Z"
200 PRINT INPUT PIVOT A UNIT VECTOR.....
220 FOR I = 0 TO 2: PRINT INS$(1); 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 280
240 IF CF% = 1 THEN GOTO 342
260 PRINT "INPUT THE COORDINATES OF PIVOT A....."
280 FOR I = 0 TO 2: PRINT INS$(1); VTAB 23: HTAB 3: INPUT AO(I): NEXT I
300 IF CF% = 1 THEN GOTO 342
320 PRINT "INPUT THE INITIAL POSITION OF POINT A1."
340 FOR I = 0 TO 2: PRINT INS$(1); VTAB 23: HTAB 3: INPUT AI(I): NEXT I
341 REM TEST FOR INPUT COMPATIBILITY
342 FOR I = 0 TO 2: A(I) = AI(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 A1 INCOMPATIBLE WITH UA...REENTER": GOTO 280
360 IF CF% = 1 THEN GOTO 760
380 PRINT "INPUT PIVOT B UNIT VECTOR....."
400 FOR I = 0 TO 2: PRINT INS$(1); VTAB 23: HTAB 3: INPUT UB(I): NEXT I
405 B = SQR(UB(0) * UB(0) + UB(1) * UB(1) + UB(2) * UB(2))
410 IF B > .95 THEN IF B < 1.01 THEN GOTO 420
415 PRINT "NOT A VALID UNIT VECTOR...REENTER": GOTO 380
420 IF CF% = 1 THEN GOTO 522
440 PRINT "INPUT THE COORDINATES OF PIVOT B....."
460 FOR I = 0 TO 2: PRINT INS$(1); VTAB 23: HTAB 3: INPUT BS(I): NEXT I
460 IF CF% = 1 THEN GOTO 522
500 PRINT "INPUT THE INITIAL POSITION OF POINT B1"
520 FOR I = 0 TO 2: PRINT INS$(1); VTAB 23: HTAB 3: INPUT BI(I): NEXT I
521 REM TEST FOR INPUT COMPATIBILITY
522 FOR I = 0 TO 2: A(I) = BI(I) - BS(I)
524 B = 0: FOR I = 0 TO 2: B = B + A(I) * UB(I)
526 IF B > -.95 THEN IF B < 1.01 THEN GOTO 540
528 PRINT "BO AND BI INCOMPATIBLE WITH UB...REENTER": GOTO 380
540 IF CF% = 1 THEN GOTO 760
560 PRINT "INPUT THE ANGULAR VELOCITY OF LINK 2..": INPUT W2
580 IF CF% = 1 THEN GOTO 760
600 PRINT "INPUT THE ANGULAR ACCEL. OF LINK 2.....": INPUT A2
620 IF CF% = 1 THEN GOTO 760
640 PRINT "INPUT THE TOTAL TRAVEL OF THE INPUT LINK DESIRED....": INPUT ALPHA
660 IF CF% = 1 THEN GOTO 760
680 PRINT "INPUT THE TRAVEL INCREMENT .......": INPUT DOTALPHA
682 NPT = INT(ALPHA / DOTALPHA)
684 IF NPT > 71 THEN PRINT "REENTER FOR MAXIMUM OF 120 POINTS..": GOTO 680
700 IF CF% = 1 THEN GOTO 760
705 NPT = INT(ALPHA / DOTALPHA)
720 PRINT "": PRINT "": PRINT "": PRINT
740 PRINT "WOULD YOU LIKE TO CHANGE ANYTHING": INPUT II$: GOTO 760
760 PRINT "WOULD YOU LIKE TO CHANGE ANYTHING ELSE": INPUT II$: CF% = 0
780 IF LEFTS(II$, 1) = "Y" THEN GOTO 840
800 IF LEFTS(II$, 1) = "N" THEN GOTO 1280
820 GOTO 740
37

840 REM WHICH CHANGE?
860 CF%= 1
880 PRINT "WHICH INPUT (TYPE NUMBER TO CHANGE)"
900 I2$ = "UA=1...AO=2...AI=3...UB=4...BO=5...BI=6...ANG VEL OF LINK 2=7...ANG A
CEL OF LINK 2=8...TOTAL TRAVEL=9...TRAVEL INCRRMT=10..."
920 VTAB 24; HTAB 1: PRINT LEFT$(I2$,39);I2$ = MID$(I2$,2) + LEFT$(I2$,1)
930 K = PEEK (-16384); IF K < 128 THEN FOR K = 1 TO 75: NEXT K: K = FRE(0): G
OTO 940
940 VTAB 24; BTAB 1: PRINT LEFT$(I2$,39);I2$ = MID$(I2$,2) + LEFT$(I2$,1)
950 INPUT CN%
960 IF CN% > 0 THEN GOTO 1020
1000 GOTO 940
1020 IF CN% < 11 THEN GOTO 1060
1040 GOTO 940
1060 IF CN% = 1 THEN GOTO 280
1100 IF CN% = 2 THEN GOTO 260
1120 IF CN% = 3 THEN GOTO 320
1140 IF CN% = 4 THEN GOTO 380
1160 IF CN% = 5 THEN GOTO 440
1180 IF CN% = 6 THEN GOTO 500
1200 IF CN% = 7 THEN GOTO 560
1220 IF CN% = 8 THEN GOTO 620
1240 IF CN% = 9 THEN GOTO 680
1260 IF CN% = 10 THEN GOTO 680
1270 REM TEST FOR PLANAR MECHANISM
1280 IF UA(0) = UB(0) THEN IF UA(1) = UB(1) THEN IF UA(2) = UB(2) THEN GOTO
1300 IF UA(0) = -UB(0) THEN IF UA(1) = -UB(1) THEN IF UA(2) = -UB(2) THEN GOTO
1320 GOTO 1560
1340 IF ao(0) = bo(0) THEN IF ao(1) = bo(1) THEN IF ao(2) = bo(2) THEN GOTO
1440 FOR i = 0 TO 2: a(i) = bo(i) - ao(i): NEXT i
1460 FOR i = 0 TO 2: b(i) = bo(i) - ao(i): NEXT i
1480 IF B = 0 THEN GOTO 1440
1500 PRINT "YOU HAVE ENTERED A PLANAR MECHANISM..."
1520 PRINT "WOULD YOU LIKE A COUPLER POINT TRACE?": INPUT ll$
1540 IF LEFT$(ll$,1) = "Y" THEN PRINT "ENTER THE COUPLER POINT PARAMETERS": PRINT "ZETA=": VTAB 23: HTAB 6: INPUT ZETA: PRINT "ETA-": VTAB 23: HTAB 5: INPUT ETA
1560 FOR I = 0 TO 2
1580 IF UA(i) = 0 THEN UA(i) = 1E -3
1600 NEXT I
1620 FOR I = 0 TO 2
1640 IF UB(i) = 0 THEN UB(i) = 1E -3
1660 NEXT I
1680 REM THE MECHANISM IS SPATIAL
1700 FOR I = 0 TO 2: IN(I,0) = UA(I): IN(I,1) = AO(I): IN(I,2) = AI(I): IN(I,3) = U
B(I): IN(I,4) = BO(I): IN(I,5) = BI(I): NEXT I
1720 PRINT D$: "OPEN INPUT FILE"
1740 PRINT D$: "WRITE INPUT FILE"
1760 FOR J = 0 TO 9: PRINT IN(J,J): PRINT IN(1,J): PRINT IN(2,J): NEXT J
1780 PRINT D$: "CLOSE INPUT FILE"
1800 HIMEM: 38399
1820 PRINT D$: "RUN RSSR3.OBJ"
1840 END
REM SUBPROGRAM RSSR3 SOURCE CODE LISTING
LIST
28 DIM UA(2),AO(2),AI(2),AJ(2,119),UB(2),BS(2),BI(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),TA(2),TB(2)
48 DIM TC(2),AS(119),UV(2),Z3(119),Z2(119),U(2),DU(2),XV(2),DB(119),DD(119),IN(2,2)
55 DS = ": REM CNTL-D
60 PRINT DS;"OPEN INPUT FILE"
70 FOR J = 0 TO 9: INPUT IN(0,J): INPUT IN(1,J): INPUT IN(2,J): NEXT J
75 PRINT DS;"CLOSE INPUT FILE"
80 FOR I = 0 TO 2:UA(I) = IN(1,0):AO(I) = IN(I,1):AI(I) = IN(I,2):UB(I) = IN(I,3):BS(I) = IN(I,4):BI(I) = IN(I,5): NEXT I
140 FOR I = 0 TO 2:TD(I) = AI(I) - AO(I):T4(I) = AI(I) - BI(I):T2(I) = BI(I) - BS(I): NEXT I
165 PM(0,0) = UB(0):PM(0,1) = UB(1):PM(1,0) = UB(0):PM(1,1) = UB(1)
170 QM(0,0) = UB(0):QM(0,1) = UB(0):QM(1,0) = UB(0):QM(1,1) = UB(0)
175 QM(1,2) = UB(1):QM(2,0) = UB(2):QM(2,1) = UB(2):QM(2,2) = UB(2)
180 FOR I = 0 TO 2: FOR K = 0 TO 2:Q1(I,K) = - QM(I,K): NEXT K: NEXT I
220 Q1(0,0) = 1 - Q1(1,0):Q1(1,1) = 1 + Q1(1,1):Q1(2,2) = 1 + Q1(2,2)
290 CV = ATN(1) / 45
300 DU = DO * CV
310 NP = INT (AL / DO)
320 BE = 0
340 FOR J = 0 TO NP
345 AS(J) = J
350 X1 = J * DU
360 IF X1 - (CV * 180) < - .028 THEN GOTO 428
370 IF X1 - (CV * 180) - .010 THEN X1 = CV * 180 - .010
420 FOR I = 0 TO 2:UA(I) = U(I): NEXT I
430 PH = X1
440 GOSUB 4008
460 FOR I = 0 TO 2:UA(I) = U(I): NEXT I
470 X1 = PH
490 FOR I = 0 TO 2:AI(I,J) = RM(I,0,1) * (AL(0) - AO(0)) + RM(I,1,1) * (AI(1) - AO(1)) + RM(I,2,1) * (AI(2) - AO(2)) + AO(1): NEXT I
520 FOR I = 0 TO 2:AT5(I) = AI(I,2) - AO(I):T3(I) = AI(I,1) - BS(I): NEXT I
690 FOR I = 0 TO 2:TD(I) = 0: FOR K = 0 TO 2:TD6(I) = TD(I) + QI(I,K) * T2(K): NEXT K: NEXT I
750 E = 0: FOR I = 0 TO 2: E = E + T3(I) * TD6(I): NEXT I
790 FOR I = 0 TO 2:TD7(I) = TD7(I) + PM(I,K) * T2(K): NEXT K: NEXT I
840 F = 0: FOR I = 0 TO 2: F = F + T3(I) * TD7(I): NEXT I
880 FOR I = 0 TO 2:TD8(I) = 0: FOR K = 0 TO 2:TD8(I) = TD8(I) + QM(I,K) * T2(K): NEXT K: NEXT I
1810 IF Y1 < 0 THEN GOTO 1148
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 * ATAN ((- 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 1138
1990 FOR I = 0 TO N: PRINT Z3(I): NEXT I
1992 FOR I = 0 TO N: PRINT Z2(I): NEXT I
1994 FOR I = 0 TO N: PRINT ZB(I): NEXT I
1996 FOR I = 0 TO N: 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(2) * 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

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),DB(71),DD(71),UA(2),A0(2),Al(2),UB(2),Be(2),Bl(2)
DIM AN(2),TA(2),BN(2),BT(2),AL(2),AR(2),BL(2),BR(2),XM(2),YM(2),ZM(2),X4(2),XS(2),X6(2),X7(2),X8(2),X9(2)
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)
DIM IN(2,9),R(2,2),P(2,71),DP(2,71),P2(2,71),S(2),PD(9,71)
DIM AN(2),TA(2),BN(2),BT(2),AL(2),AR(2),BL(2),BR(2),XM(2),YM(2),ZM(2),X4(2),XS(2),X6(2),X7(2),X8(2),X9(2)

REM SUBPROGRAM PRINT

100 REM PROGRAM PRINT
108 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),DB(71),DD(71),UA(2),A0(2),Al(2),UB(2),Be(2),Bl(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),AR(2),BL(2),BR(2),XM(2),YM(2),ZM(2),X4(2),XS(2),X6(2),X7(2),X8(2),X9(2)
117 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)
120 REM RETRIEVE VALUES FROM DISK FILES
140 D$: REM CNTRL-D
160 PRINT D$: OPEN INPUT FILE: PRINT D$: READ INPUT FILE
180 FOR J = 8 TO 9: INPUT IN(0,J): INPUT IN(1,J): INPUT IN(2,J): NEXT J
200 PRINT D$: CLOSE INPUT FILE
220 FOR I = 0 TO NP: UA(I) = IN(I,0): AO(I) = IN(I,1): AI(I) = IN(I,2): UB(I) = IN(I,3): Bl(I) = IN(I,4): B1(I) = IN(I,5): NEXT I
240 FOR I = 0 TO NP: INPUT AJ(I,0): INPUT AJ(I,1): NEXT I
260 FOR I = 0 TO NP: INPUT AN(I): NEXT I
280 FOR I = 0 TO NP: INPUT AN(I): NEXT I
300 FOR I = 0 TO NP: DP(I) = IN(I,0): ETA = IN(I,1): NEXT I
320 NP = INT (AL / DO)
340 PRINT D$: OPEN OUTPUT FILE: PRINT D$: READ OUTPUT FILE
360 FOR I = 0 TO NP: INPUT AJ(I,0): INPUT AJ(I,1): NEXT I
380 FOR I = 0 TO NP: INPUT AN(I): NEXT I
400 FOR I = 0 TO NP: INPUT AN(I): NEXT I
420 FOR I = 0 TO NP: INPUT AN(I): NEXT I
440 FOR I = 0 TO NP: INPUT AN(I): NEXT I
460 FOR I = 0 TO NP: INPUT AN(I): NEXT I
480 PRINT D$: CLOSE OUTPUT FILE
600 REM COUPLER POINT PARAMETERS
620 REM COMPUTE THE ROTATION MATRIX
640 FOR I = 0 TO 2: U(I) = UA(I): NEXT I
660 FOR I = 0 TO 2: S(I) = Bl(I) - Al(I): NEXT I
680 SU = 0: FOR I = 9 TO 2: SU = SU + S(I) * S(I): NEXT I
700 D = SQR (SU)
720 ZD = ETA / D; ED = ETA / D
730 IF ETA = 0 THEN IF ETA = S THEN GOTO 1888
740 REM LOOP FOR COUPLER POINTS
750 FOR J = 0 TO NP
760 REM POSITION OF COUPLER POINT J
780 FOR I = 0 TO 2
800 P(I,J) = AJ(I,J) + ZD * (BJ(I,J) - AJ(I,J)) + ED * (R(I,0) * (BJ(0,J) - AJ(0,J)) + R(I,1) * (BJ(1,J) - AJ(1,J)) + R(I,2) * (BJ(2,J) - AJ(2,J)))
820 NEXT I
840 REM VELOCITY OF COUPLER POINT J
860 FOR I = 0 TO 2
870 DP(I,J) = VA(I,J) + ZD * (VB(I,J) - VA(I,J)) + ED * (R(I,0) * (VB(0,J) - VA(0,J)) + R(I,1) * (VB(1,J) - VA(1,J)) + R(I,2) * (VB(2,J) - VA(2,J)))
880 NEXT I
900 REM ACCELERATION OF COUPLER POINT J
920 FOR I = 0 TO 2
940 NEXT I
960 NEXT J
1888 REM PRINT, PLOT, OR ANIMATE?
1890 PRINT "WOULD YOU LIKE A PRINT OF THE RESULTS": VTAB 23; HTAB 39: INPUT 11$
1840 IF INPUTS [118,1] = "Y" THEN GOTO 5088
1860 IF INPUTS [118,1] = "N" THEN PRINT "WOULD YOU LIKE A PRINT OF THE RESULTS"
VTAB 23; HTAB 39: INPUT 116
1000 IF LEFT$ (11$, 1) = "Y" THEN GOTO 10800
1100 IF LEFT$ (11$, 1) = "N" THEN PRINT "WOULD YOU LIKE TO SEE AN ANIMATION?"
VTAB 23; HTAB 37: INPUT 116
1120 IF LEFT$ (11$, 1) = "Y" THEN GOTO 12000
1140 IF LEFT$ (11$, 1) = "N" THEN PRINT 0$; "RUN INPUT2"
1160 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$; "PRtl"
5080 POKE 1657.80: REM AUTHOR'S SETUP
5100 PRINT "******************************************"
5120 PRINT "* R S S R *"
5130 PRINT "* MECHANISM ANALYSIS *"
5140 PRINT "* RESULTS *"
5150 PRINT "*"
5160 PRINT "******************************************"
5180 PRINT 1 PRINT
5190 PRINT "PROGRAM INPUTS"
5200 PRINT "-------------"
5210 PRINT "INITIAL MECHANISM CONFIGURATION:"
5220 PRINT "------------------"
5230 PRINT "POINT A UNIT VECTOR POINT B UNIT VECTOR"
5240 PRINT "X = "; UA(0); SPC( 53 - POS (0)); "Y = "; UB(0)
5250 PRINT "Z = "; UA(2); SPC( 53 - POS (0)); "\n5260 PRINT "X = "; UA(0); SPC( 53 - POS (0)); "Y = "; UB(1)
5270 PRINT "Z = "; UA(2); SPC( 53 - POS (0)); "\n5280 PRINT POINT A8 POINT B8"
5290 PRINT "X = "; AO(0); SPC( 53 - POS (0)); "Y = "; BO(0)
5300 PRINT "Z = "; AO(2); SPC( 53 - POS (0)); "\n5310 PRINT "X = "; AO(0); SPC( 53 - POS (0)); "Y = "; BO(1)
5320 PRINT "Z = "; AO(2); SPC( 53 - POS (0)); "\n5330 PRINT POINT A1 POINT B1"
5340 PRINT "X = "; AI(0); SPC( 53 - POS (0)); "Y = "; BI(0)
5350 PRINT "Z = "; AI(2); SPC( 53 - POS (0)); "\n5360 PRINT "X = "; AI(0); SPC( 53 - POS (0)); "Y = "; BI(1)
5370 PRINT "Z = "; AI(2); SPC( 53 - POS (0)); "\n5380 PRINT ANGULAR VELOCITY OF INPUT LINK........... ";W2; " RAD/SEC"
5390 PRINT ANGULAR ACCELERATION OF INPUT LINK...... ";A2; " RAD/SEC/SEC"
5400 PRINT TOTAL TRAVEL OF INPUT LINK............... ";AL; DEGREES"
5410 PRINT INCREMENT................................. ";DI; DEGREES"
5430 IF ZETA = 0 THEN IF ETA = 0 THEN GOTO 5490
5440 PRINT THESE INPUTS INDICATE A PLANAR MECHANISM."
5450 PRINT THE COUPLER POINT PARAMETERS ARE.......
5470 PRINT ZETA = "$;ZETA
5480 PRINT ETA = "$;ETA
5490 PRINT : PRINT : PRINT
5500 PRINT SPC( 36); "RESULTS"
5510 PRINT SPC( 35); "-----------"
5520 PRINT : PRINT
5530 PRINT SPC( 25); "* DENOTES NO ASSEMBLY POSSIBLE"
5540 PRINT : PRINT
5550 PRINT "ANGLE"; SPC( 39); "ANGLE BETA";
5560 PRINT "ALPHA"; SPC( 16); "POSITION"; SPC( 13); "VELOCITY"; SPC( 13); "ACCELERATION"
5570 PRINT "(DEG)"; SPC( 16); "(DEG)"; SPC( 16); "(RAD/SEC)"; SPC( 12); "(RAD/SEC/SEC)"
5580 PRINT "------------------"
5590 FOR I = 0 TO NP
5600 IF AS$(I) = 0 THEN PRINT Z3(I); SPC( 21 - POS (0)); "*"; SPC( 42 - POS (0))
43

PRINT Z3(I); SPC( 21 - POS (0)); Z2(I); SPC( 42 - POS (0)); DB(I); SPC ( 23 - POS (0)); DD(I)
GOTO 5620

PRINT Z3(I); SPC( 21 - POS (0)); AJ(0,I); SPC( 42 - POS (0)); AJ(1,I); SPC ( 23 - POS (0)); AJ(2,I)
GOTO 5718

PRINT Z3(I); SPC( 21 - POS (0)); BJ(0,I); SPC( 42 - POS (0)); BJ(1,I); SPC ( 23 - POS (0)); BJ(2,I)
GOTO 5878

PRINT Z3(I); SPC( 21 - POS (0)); VA(0,I); SPC( 42 - POS (0)); VA(1,I); SPC ( 23 - POS (0)); VA(2,I)
GOTO 6038

PRINT Z3(I); SPC( 21 - POS (0)); AB(0,I); SPC( 42 - POS (0)); AB(1,I); SPC ( 23 - POS (0)); AB(2,I)
GOTO 6198

PRINT Z3(I); SPC( 21 - POS (0)); P(0,I); SPC( 42 - POS (0)); P(1,I); SPC ( 23 - POS (0)); P(2,I)
GOTO 6268
44

6250 PRINT Z3(I); SPC( 21 - POS (0)); DP(I,1); SPC ( 23 - POS (0)); DP(2,1)
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)); P2(I,1); SPC ( 42 - POS (0)); P2(2,1)
6330 NEXT I
6340 GOTO 6990
6500 PRINT i; PRINT i; PRINT i
6520 PRINT •ANGLE•; SPC( 32); ST$
6540 PRINT •ALPHA•
6560 PRINT •(DEG)•; SPC( 20); •x•; SPC( 20); •Y•; SPC( 28); •z•
6690 RETURN
6800 F$ = 0: IF ASI(I) = 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$; "PR4B"
7000 PRINT "END OF PLOT ROUTINE"
7020 PRINT "WOULD YOU LIKE PLOTS OF THE RESULTS": VTAB 23: HTAB 38: INPUT 11$:
GOTO 1080
10000 REM BUILD PLOT DATA FILE
10020 FOR I = 0 TO NP
10040 PD(0,1) = Z3(I)
10060 PD(1,1) = Z2(I)
10080 PD(2,1) = BJ(0,1)
10100 PD(3,1) = BJ(1,1)
10120 PD(4,1) = BJ(2,1)
10140 PD(5,1) = DB(I)
10160 PD(6,1) = DD(I)
10180 PD(7,1) = P(0,1)
10200 PD(8,1) = P(1,1)
10220 PD(9,1) = P(2,1)
10240 NEXT I
10260 REM BUILD PLOT DESCRIPTOR FILE
10280 PRINT D$; "OPEN PLOT DESCRIPTOR"
10300 PRINT D$; "WRITE PLOT DESCRIPTOR"
10320 PRINT 9: REM NO. OF VARIABLES
10340 PRINT 71: REM MAX NO. OF POINTS
10360 PRINT NP: REM ACTUAL NO. OF POINTS
10380 PRINT "INPUT ANGLE ALPHA"
10400 PRINT "OUTPUT ANGLE BETA"
10420 PRINT "POS. OF FT B X"
10440 PRINT "POS. OF FT B Y"
10460 PRINT "POS. OF FT B Z"
10480 PRINT "OUTPUT ANGULAR VEL."
10500 PRINT "OUTPUT ANGULAR ACC."
10520 PRINT "COUPLER PT X POS."
10540 PRINT "COUPLER PT Y POS."
10560 PRINT "COUPLER PT Z POS."
10580 PRINT D$; "CLOSE PLOT DESCRIPTOR"
10600 PRINT D$; "RUN PLOT ROUTINE.C"
10620 END
12000 REM BUILD DATA FOR ANIMATION
12020 FOR I = 0 TO 2 I$: I$ = I$ + 1: NEXT I
12040 FOR I = 0 TO 2 I$: I$ = I$ + 1: NEXT I
12080 LA = SQR (SU)
12100 FOR I = 0 TO 2 I$: I$ = I$ + 1: NEXT I
LP UA(8) P(2,I): NEXT
ZM A:X5(0) •UB(B)
AaX9(0) •A:Y8(1) •8
A:X7(8) •1316e FOR
12960 X4(8) •12900 XM(0) •12780
12540 NEXT I
12330 AN(2) = - AN(2) •TA(2) = - TA(2) •BN(2) = - BN(2) •BT(2) = - BT(2)
12340 REM CALCULATE ARROW HEADS
12360 K1 = 2 * LF / 3: K2 = LF / 6
12380 REM ARROWHEAD FOR UNIT VECTOR A
12400 U(0) = UA(2):U(1) = 8:U(2) = - UA(0)
12420 GOSUB 14600
12440 FOR I = 0 TO 2
12460 AL(I) = AO(I) + UA(I) • K1 + K2 • (R(I,0) • UA(I) + R(I,1) • UA(1) + R(I,2)
I) • UA(I))
12480 AR(I) = AO(I) + UA(I) • K1 - K2 • (R(I,0) • UA(I) + R(I,1) • UA(1) + R(I,2
I) • UB(I))
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) = 8:U(2) = - UB(0)
12560 GOSUB 14600
12580 FOR I = 0 TO 2
12600 BL(I) = B0(I) + UB(I) • K1 + K2 • (R(I,0) • UB(I) + R(I,1) • UB(1) + R(I,2
I) • UB(I))
12620 BR(I) = B0(I) + UB(I) • K1 - K2 • (R(I,0) • UB(I) + R(I,1) • UB(1) + R(I,2
I) • UB(I))
12640 NEXT I
12650 BL(2) = - BL(2): BR(2) = - BR(2)
12760 REM DETERMINE AXES
12780 IF AO(0) > B0(0) THEN XM = AO(0) + LM
12800 XM = B0(0) + LM
12820 IF AO(1) > B0(1) THEN YM = AO(1) + LM
12840 YM = B0(1) + LM
12860 IF AO(2) > B0(2) THEN ZM = AO(2) + LM
12880 ZM = B0(2) + LM
12900 XM(0) = XM: YM(1) = YM: ZM(2) = - ZM
12920 REM CREATE ARROWHEADS AND LABELS
12940 A = .15: B = .3: C = .6
XM = XM - (B * XM/4(I)) = - A: X5(0) = XM - B: X5(1) = A: X6(0) = XM + B: X6(1)
A: X7(0) = XM + C: X7(1) = A: X8(0) = XM + C: X8(1) = - A: X9(0) = XM + B: X9(1)
= - A
12980 Y4(1) = XM - B: Y4(0) = A: Y5(1) = YM - B: Y5(0) = - A: Y6(1) = YM + B: Y7(1)
Y = YM + C: Y(0) = - A: Y8(1) = YM + C: Y8(0) = A: Y9(1) = YM + A
ZM + B: Z7(1) = - A: Z7(2) = -(ZM + C): Z8(1) = A: Z8(2) = -(ZM + C): Z9(1)
= A: Z9(2) = -(ZM + B)
13100 PRINT D5: "OPEN ANIMATION FILE": PRINT D5: "WRITE ANIMATION FILE" PRINT D5
13180 FOR I = 0 TO NP: PRINT A(I,1): PRINT BJ(0,1): PRINT P(0,1): PRINT - P(2,1): NEXT I
13185 FOR I = 0 TO NP: PRINT A(I,2): PRINT BJ(1,1): PRINT P(1,1): PRINT - P(2,1): NEXT I
13186 FOR I = 0 TO NP: PRINT A(I,3): NEXT I
13187 FOR I = 0 TO NP: PRINT A(I,4): NEXT I
PRINT Z9(1): NEXT I
13189 PRINT D5: "CLOSE ANIMATION FILE"
13226 PRINT D$; "RUN ANIMATE"
14006 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)
14026 R(1,0) = U(0) * U(1) + U(2); R(1,1) = U(1) * U(1)
14036 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)
14046 RETURN
REM PROGRAM ANIMATE

50 LOMEM: 16384
70 HIMEM: 24575
180 REM PROGRAM ANIMATE
120 DIM AN(2), TA(2), BN(2), BT(2), AL(2), AR(2), BL(2), BR(2), XM(2), YM(2), ZM(2), X4(2), X5(2), X6(2), X7(2), X8(2), X9(2)
160 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), EY(2)
280 REM RETRIEVE VALUES FROM DISK FILE
220 DS = "": REM CNTRL-D
240 PRINT DS; "OPEN INPUT FILE": PRINT DS; "READ INPUT FILE"
260 FOR J = 0 TO 9: INPUT IN(0, J): INPUT IN(1, J): INPUT IN(2, J): NEXT J
280 PRINT DS; "CLOSE INPUT FILE"
300 FOR I = 0 TO 2: UA(I) = IN(1, 0): AO(I) = IN(1, 1): AL(I) = IN(1, 2): UB(I) = IN(0, 0): U(I) = IN(0, 1): UL(I) = IN(0, 2): V(I) = IN(0, 3): V(I) = IN(0, 4): W(I) = IN(0, 5): NEXT I
320 FOR I = 0 TO 2: BV(I, 0, 0) = IN(I, 0): BV(I, 0, 1) = IN(I, 1): BV(I, 0, 2) = IN(I, 2): BV(I, 0, 3) = IN(I, 3): BV(I, 0, 4) = IN(I, 4): NEXT I
340 FOR I = 0 TO 2: F(I) = IN(I, 5): NEXT I
360 FOR I = 0 TO 2: SF(1, 0, I) = IN(I, 6): SF(1, I, I) = IN(I, 7): SF(1, I, I) = IN(I, 8): SF(1, I, I) = IN(I, 9): NEXT I
380 FOR I = 0 TO 2: SP(I) = IN(I, 10): NEXT I
400 FOR I = 0 TO 2: M(I) = IN(I, 11): NEXT I
420 FOR J = 0 TO 2: I = IN(J, 12): NEXT J
440 FOR J = 0 TO 2: I = IN(J, 13): NEXT J
460 FOR J = 0 TO 2: I = IN(J, 14): NEXT J
480 FOR J = 0 TO 2: I = IN(J, 15): NEXT J
500 FOR J = 0 TO 2: I = IN(J, 16): NEXT J
520 FOR J = 0 TO 2: I = IN(J, 17): NEXT J
540 FOR J = 0 TO 2: I = IN(J, 18): NEXT J
560 PRINT DS; "CLOSE ANIMATION FILE"
580 REM BRING IN SKELETON ARRAY AND LOAD THE GRAPHICS GENERATOR
600 PRINT DS; "LOAD RSR SKELETON ARRAY"
620 PRINT DS; "CLOSE ANIMATION FILE"
640 REM STUFF THE SKELETON ARRAY WITH FIXED POINTS (AXES, ETC.)
660 REM STUFF THE SKELETON ARRAY WITH FIXED POINTS (AXES, ETC.)
680 REM STUFF THE SKELETON ARRAY WITH FIXED POINTS (AXES, ETC.)
700 REM STUFF THE SKELETON ARRAY WITH FIXED POINTS (AXES, ETC.)
720 REM STUFF THE SKELETON ARRAY WITH FIXED POINTS (AXES, ETC.)
740 REM STUFF THE SKELETON ARRAY WITH FIXED POINTS (AXES, ETC.)
760 REM STUFF THE SKELETON ARRAY WITH FIXED POINTS (AXES, ETC.)
780 REM STUFF THE SKELETON ARRAY WITH FIXED POINTS (AXES, ETC.)
800 REM STUFF THE SKELETON ARRAY WITH FIXED POINTS (AXES, ETC.)
820 REM STUFF THE SKELETON ARRAY WITH FIXED POINTS (AXES, ETC.)
840 REM STUFF THE SKELETON ARRAY WITH FIXED POINTS (AXES, ETC.)
860 REM STUFF THE SKELETON ARRAY WITH FIXED POINTS (AXES, ETC.)
880 REM STUFF THE SKELETON ARRAY WITH FIXED POINTS (AXES, ETC.)
900 REM STUFF THE SKELETON ARRAY WITH FIXED POINTS (AXES, ETC.)
920 REM STUFF THE SKELETON ARRAY WITH FIXED POINTS (AXES, ETC.)
940 REM STUFF THE SKELETON ARRAY WITH FIXED POINTS (AXES, ETC.)
960 REM STUFF THE SKELETON ARRAY WITH FIXED POINTS (AXES, ETC.)
980 REM STUFF THE SKELETON ARRAY WITH FIXED POINTS (AXES, ETC.)
1000 REM STUFF THE SKELETON ARRAY WITH FIXED POINTS (AXES, ETC.)
1826 MEM = 33174: FOR I = 0 TO 2:BV% = INT (TA(I) * SF%): GOSUB 6000: NEXT I
1840 MEM = 33181: FOR I = 0 TO 2:BV% = INT (AM(I) * SF%): GOSUB 6000: NEXT I
1856 MEM = 33188: FOR I = 0 TO 2:BV% = INT (AL(I) * SF%): GOSUB 6000: NEXT I
1860 MEM = 33195: FOR I = 0 TO 2:BV% = INT (AR(I) * SF%): GOSUB 6000: NEXT I
1100 MEM = 33062: FOR I = 0 TO 2:BV% = INT (AO(I) * SF%): GOSUB 6000: NEXT I
1120 MEM = 33097: FOR I = 0 TO 2:BV% = INT (BR(I) * SF%): GOSUB 6000: NEXT I
1140 MEM = 33282: FOR I = 0 TO 2:BV% = INT (BR(1) * SF%): GOSUB 6000: NEXT I
1160 MEM = 33289: FOR I = 0 TO 2:BV% = INT (BN(I) * SF%): GOSUB 6000: NEXT I
1180 MEM = 33223: FOR I = 0 TO 2:BV% = INT (BL(I) * SF%): GOSUB 6000: NEXT I
1200 MEM = 33216: FOR I = 0 TO 2:BV% = INT (BR(I) * SF%): GOSUB 6000: NEXT I
1220 MEM = 33230: FOR I = 0 TO 2:BV% = INT (BR(1) * SF%): GOSUB 6000: NEXT I
1240 MEM = 33237: FOR I = 0 TO 2:BV% = INT (XT(I) * SF%): GOSUB 6000: NEXT I
1260 MEM = 33244: FOR I = 0 TO 2:BV% = INT (XB(I) * SF%): GOSUB 6000: NEXT I
1280 MEM = 33251: FOR I = 0 TO 2:BV% = INT (X6(I) * SF%): GOSUB 6000: NEXT I
1292 MEM = 33258: FOR I = 0 TO 2:BV% = INT (X5(I) * SF%): GOSUB 6000: NEXT I
1324 MEM = 33265: FOR I = 0 TO 2:BV% = INT (XM(I) * SF%): GOSUB 6000: NEXT I
1334 MEM = 33272: FOR I = 0 TO 2:BV% = INT (X4(I) * SF%): GOSUB 6000: NEXT I
1280 MEM = 33069: FOR I = 0 TO 2:LO% = AJ%(I,0,0):HI% = AJ%(I,1,J): GOSUB 7000: NEXT I
1290 IF ETA = 0 THEN IF ETA = 0 THEN MEM = 33076: FOR I = 0 TO 2:LO% = AJ%(I,0,0):HI% = AJ%(I,1,J): GOSUB 7000: NEXT I: GOTO 1293
1291 MEM = 33076: FOR I = 0 TO 2:LO% = P%(I,0,0):HI% = P%(I,1,J): GOSUB 7000: NEXT I
1293 MEM = 33083: FOR I = 0 TO 2:LO% = BJ%(I,0,0):HI% = BJ%(I,1,J): GOSUB 7000: NEXT I
1294 IF ETA = 0 THEN IF ETA = 0 THEN MEM = 33098: FOR I = 0 TO 2:LO% = BJ%(I,0,0):HI% = BJ%(I,1,J): GOSUB 7000: NEXT I
1295 MEM = 33098: FOR I = 0 TO 2:LO% = P%(I,0,0):HI% = P%(I,1,J): GOSUB 7000: NEXT I
1296 IF F% = 1 THEN GOTO 1335
1300 EY(0) = XM(0) / 3: EY(1) = YM(1) / 4: EY(2) = - 2 * XM(0)
1310 P1 = $:BA = $:HE = $
1315 IF = 1
1320 MEM = 33031: FOR I = 0 TO 2:BV% = INT (EY(I) * SF%): GOSUB 6000: NEXT I
1330 POKE 33030,P1: POKE 33039,BA: POKE 33048,HE
1335 CALL 24576: IF I% = 1 THEN 11$ = "N": GOTO 1380
1340 INPUT "EYE POSITION SATISFACTORY (Y/M)?:";11$
1360 IF LEFT$(11$,1) = "Y" THEN GOTO 3000
1380 IF LEFT$(11$,1) = "N" THEN INPUT "NEW EYE POSITION (X,Y,Z)";EY(0),EY(1),EY(2)
1390 EY(2) = - EY(2):I% = 0
1400 INPUT "NEW PITCH, BANK, AND HEADING... (ENTER PSEUDODEGREES; 256=FULL CIRCL E)";P1,BA,HE: PRINT: PRINT: PRINT: PRINT: GOTO 1320
3000 REM THE ANIMATION LOOP
3020 POKE 33027,17: POKE 33028,17
3045 POKE 33104,121: POKE 33059,13
3050 MI = 33556:MF = NP * 70 + 71 + MI
3055 FOR M = MI TO MF: POKE M,17: NEXT M
3060 A = $:B = NP+C = 1:BP% = $
3120 FOR J = A TO B STEP C
3140 M = 33556 + J * 76
3220 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 3260
3240 MEM = 33076: FOR I = 0 TO 2:LO% = P%(I,0,J):HI% = P%(I,1,J): GOSUB 7000: NEXT I
3260 MEM = 33083: FOR I = 0 TO 2:LO% = BJ%(I,0,J):HI% = BJ%(I,1,J): GOSUB 7000: NEXT I
3270 IF ETA = 0 THEN IF ETA = 0 THEN MEM = 33098: FOR I = 0 TO 2:LO% = BJ%(I,0,J):HI% = BJ%(I,1,J): GOSUB 7000: NEXT I: GOTO 3300
3280 MEM = 33098: FOR I = 0 TO 2:LO% = P%(I,0,J):HI% = P%(I,1,J): GOSUB 7000: NEXT I
3300 MEM = 33059:O$=BV% = M = 65536: GOSUB 6000
3320 CALL 24576
3330 FOR I = M + 34 TO M STEP - 1: IF PEEK (I) = 121 THEN POKE 1,17: GOTO 33
3335 NEXT I
3338 POKE M + 33,27: POKE M + 34,SP$ 3340 MEM = 33059:I = $BV: (M + 35) - 65536: GOSUB 6000
3340 CALL 24576
3340 FOR I = M + 69 TO M + 35 STEP - 1: IF PEEK (I) = 121 THEN POKE I,17: GO TO 3380
3370 NEXT I
3380 NEXT J
3385 HGR s REM CLEAR GRAPHICS SCREEN
3390 POKE 33059,26: POKE 33060,43: POKE 33061,0
3395 CALL 24576
3400 REM ANIMATION CALL
3400 PRINT: PRINT: PRINT "PRESS ANY KEY TO STOP ANIMATION"
3400 POKE 33059,17: POKE 33060,17: POKE 33061,17
3400 POKE 33055,11:MEM = 33055:I = $BV: = -32188: GOSUB 6000
3400 POKE 33053,12: POKE 33054,01
3400 CALL 24576
3400 IF PEEK (-16384) > 127 THEN : GOTO 3500
3400 GOTO 3480
3500 INPUT "SPEED O.K.? (Y/N)";II$: IF II$ = "N" THEN INPUT "SELECT SPEED. (5 SLOWEST, 0 FASTEST)";SP$: FOR I = 33390 TO MF STEP 78: POKE I,SP$: NEXT I: GOTO 3480
3500 INPUT "VIEWPOINT O.K.? (Y/N)";II$: IF II$ = "N" THEN P$ = 1: POKE 33054,0: FOR I = 33855 TO 33857: POKE I,17: NEXT I: POKE 33104,17: POKE 33027,6: POKE 33028,0: GOTO 3490
3500 INPUT "CONTINUE WITH ANIMATION? (Y/N)";II$: IF II$ = "Y" THEN GOTO 3400
3500 INPUT "TRY A DIFFERENT MECHANISM? (Y/N)";II$: IF II$ = "Y" THEN PRINT D$: RUN INPUT2 3500 END
3500 END
6800 HI% = BV% / 256
6801 IF BV% < 0 THEN HI% = HI% + 1
6802 LO% = BV% - HI% * 256
6800 IF BV% > -1 THEN GOSUB 7000: RETURN
6803 HI% = 255 + HI%
6804 LO% = 256 + LO%
6805 IF LO% < 256 THEN GOSUB 7000: RETURN
6806 HI% = HI% + 1
6807 LO% = 0
6800 GOSUB 7000: RETURN
7000 POKE (MEM + 2 * I) + 1,LO%
7010 POKE (MEM + 2 * I) + 2,HI% 7040 RETURN
APPENDIX D

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

100 REM SSSSSSSSSSSSSSSSSSSSSSSSSSSSSS
110 REM S INITIALIZE S
120 REM SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS
130 REM * SET LOMEM ABOVE *
135 REM * GRAPHICS *
140 LOMEM: 16384
150 REM * SAVE HIMEM *
160 REM * SET LOHEM ABOVE *
135 REM * LOAD HRCG, ASCII.SET *
140 LOMEM: 16384
150 REM * SAVE BIMEM *
160 HL' PEEK (llS) 1HH' PEEK (116)
170 REM * LOAD HRCG, ASCII.SET *
180 REM * LOAD LANDSCAPE.SET *
190 ONERR GOTO 440
200 TEXT • HOME • HDR • ADRS • 8
210 PRINT CHR$(4); "LOAD RBOOT"; CALL 520
220 ADRS • USR (0), HRcG •
230 POKE 216,8: REM RESET ONERR
240 IF ADRS < 0 THEN ADRS = ADRS + 65536
250 CS = ADRS = 2 • 768: HIMEM: CH
260 CH = INT (CS / 256): CL = CH • CH • 256
270 POKE ADRS + 7,CL: POKE ADRS + 8,CH
280 D$ = CHR$(4): REM CTRL-D
290 PRINT D$; "LOAD ASCII.SET, A"; CS
300 PRINT D$; "LOAD LANDSCAPE.SET, A"; CS + 768
310 CALL ADRS + 3
320 REM * CTRL CHARACTERS *
330 CP$ = 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$(15): REM OPTIONS
365 CN$ = CHR$(14): REM NORMAL VIDEO
370 CS$ = CHR$(19): REM SHIFT
380 CA$ = CHR$(1): REM SELECT CHR SET OR PG 1
390 REM * PLOT DENSITY *
400 HD = 196,1D = 168
410 PRINT CI$
420 PRINT "AFTER PLOT IS FINISHED PRESS S TO STORE IMAGE OF PLOT OR ANY KEY TO CONTINUE"
425 PRINT CH$
430 GOTO 2000
440 TEXT
450 PRINT "ERROR IN RLOAD OF RBOOT"
460 POKE 216,8
470 END
1000 REM SSSSSSSSSSSSSSSSSSSSSSSSSSSS
1010 REM S READ PLOT S
1015 REM S DESCRIPTOR FILE S
1020 REM SSSSSSSSSSSSSSSSSSSSSSSSSSS
1030 PRINT D$; "OPEN PLOT DESCRIPTOR"
1040 PRINT D$; "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 = 0 TO NV%
1125 NEXT I
1130 INPUT HAS(I)
1140 NEXT I
1150 PRINT D$; "CLOSE PLOT DESCRIPTOR"
1160 RETURN
2000 REM SSSSSSSSSSSSSSSSSSSSSSSSSSSSSS
2010 REM & READ PLOT DATA FILE $  
2020 REM &$SSHSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS
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$(Y%)) - I, 1)
5200 NEXT I
5205 PRINT CA$; 1: REM ASCII.SET
5210 TA = INT (HD / 14) - INT (LEN (NA$(X%)) / 2)
5220 IF TA% < 1 THEN TA% = 1
5230 PRINT NA$(X%)
5240 REM * ANNOTATE THE AXIS *
5250 REM * WITH AXIS INCREMENTS *
5260 FOR J = 0 TO 4
5270 Y TICK = YN + J * YD
5280 X TICK = XN + J * XD
5290 NUM = Y TICK: GOSUB 7000: Y TICK$ = NUM$
5300 NUM = X TICK: GOSUB 7000: X TICK$ = NUM$
5310 FOR I = 1 TO OP
5320 XP = VTAB INT ((PD(X%, I) - XN) / XF) + 279 - BD
5330 BPLOT TO XP, YP
5340 NEXT I
5350 FOR I = 1 TO LEN (XTICK$)
5360 PRINT XTICK$
5370 NEXT J
5380 RETURN
6000 REM SSSSSSSSSSSSSSSSSSSSSS
6010 REM S PLOT THE POINTS S
6020 REM SSSSSSSSSSSSSSSSSSSSSS
6030 YP = VD + 4 - INT ((PD(Y%, 0) - YN) / YF)
6040 XP = INT ((PD(X%, 0) - XN) / XF) + 279 - BD
6050 HPLOT TO XP, YP
6060 FOR I = 1 TO OP
6070 YP = VD + 4 - INT ((PD(Y%, I) - YN) / YF)
6080 XP = INT ((PD(X%, I) - XN) / XF) + 279 - BD
6090 HPLOT TO XP, YP
6100 NEXT I
6110 RETURN
7000 REM SSSSSSSSSSSSSSSSSSSSSS
7010 REM S PLOT NUMBERS S
7020 REM SSSSSSSSSSSSSSSSSSSSSS
7030 REM SET DECIMAL PLACES.
7040 D = 2
7050 REM SSSSSSSSSSSSSSSSSSSSSS
7060 REM S PASTE STR$
7070 REM SSSSSSSSSSSSSSSSSSSSSS
7080 REM S PASTE CHARS
7090 REM
7100 REM CHECK FOR EXPONENT.
7110 FOR I = 1 TO LEN (NUM$)
7120 IF MID$ (NUM$, I, 1) < > "E" THEN NEXT I
7130 REM I IS NOW AT EXPONENT OR END.
7140 REM
7150 REM CHECK FOR DECIMAL.
7160 FOR K = 1 TO I - 1
7170 IF MID$ (NUM$, K, 1) < > "." THEN NEXT K
7180 REM K IS NOW AT DECIMAL OR END.
7190 REM
7200 REM DO DIGITS EXIST TO THE RIGHT OF THE DECIMAL?
7210 IF K + D < 1 - 1 THEN LP$ = K + D: GOTO 7250: REM YES
7220 LP$ = I - 1: REM NO, SO PRINT ALL.
7230 REM
7240 REM ROUND OFF MANTISSA.
7250 MAN = VAL (LEFT$ (NUM$, LP$ + 1)): P = 185: MAN = INT (MAN * P + .5) / P: MAN$ = STR$ (MAN)
7260 REM
7270 REM CONCAT EXPONENT PART AND RETURN.
7275 NUMS$ = MANS$ + MIDS$ (NUMS$)!
7276 RETURN
8000 REM MMMMMMMMMMMMMMMMMMMMMMM
8010 REM M
8020 REM M GENERAL PURPOSE M
8030 REM M PLOTTING ROUTINE M
8040 REM M
8050 REM M UPDATED: 05/24/82 M
8060 REM M
8070 REM MMMMMMMMMMMMMMMMMMM
8080 REM
8130 REM AAAAAAAAAAAAAAAAAAAAA
8140 REM A READ PLOT A
8150 REM A DESCRIPTOR FILE A
8160 REM AAAAAAAAAAAAAAAAAAAAAAAAA
8170 GOSUB 1000
8180 REM AAAAAAAAAAAAAAAAAAAAA
8190 REM A READ PLOT DATA FILE A
8200 REM AAAAAAAAAAAAAAAAAAAAAAAAA
8210 GOSUB 2000
8220 REM AAAAAAAAAAAAAAAAAAAAA
8230 REM A GET X & Y VARIABLES A
8240 REM AAAAAAAAAAAAAAAAAAAAAAAAA
8250 GOSUB 3000
8260 REM AAAAAAAAAAAAAAAAAAAAA
8270 REM A FIND MINS, MAXS AND A
8280 REM A SCALE FACTOR A
8290 REM AAAAAAAAAAAAAAAAAAAAA
8300 GOSUB 4000
8310 IF YF$ = "" THEN PRINT "THE DEPENDENT VARIABLE IS CONSTANT ";PD(Y%,0)"."
GOTO B470
8320 REM AAAAAAAAAAAAAAAAAAAAA
8330 REM A DRAW & LABEL AXIS A
8340 REM AAAAAAAAAAAAAAAAAAAAAAAAA
8350 GOSUB 5800
8360 REM AAAAAAAAAAAAAAAAAAAAA
8370 REM A PLOT THE POINTS A
8380 REM AAAAAAAAAAAAAAAAAAAAA
8390 GOSUB 6800
8400 REM * WAIT FOR KEYPRESS *
8410 GET A$.
8420 REM * IF CHR=$ THEN STORE *
8430 REM * PLOT ON DISK *
8440 IF A$ < > "E" THEN GOTO B470
8440 REM DISCONNECT HRCC SO AS TO NOT MESS UP PLOT
8444 POKE 54,248: POKE 55,253: CALL 1802: PRINT
8452 REM RECONNECT HRCC
8454 POKE 54,24: 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$: PRINT
8505 REM NORMAL VIDEO
8510 IF LEFT$(A$,1) = "N" THEN GOTO B540
8530 GOTO B220
8540 REM
8550 REM CLEAN UP & EXIT.
8560 REM
8570 REM RESET CSW & KSW FOR NORMAL I/O.
8580 POKE 54,248: POKE 55,253: POKE 56,27: POKE 57,253
8590 REM RECONNECT DOS.
8600 CALL 1002
8610 REM RESET HIMEM
8620 POKE 115,HL$: 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 CHARS. 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. *

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

ORG $2E5

SET UP THE & VECTOR AND RETURN.

*BEGINNING OF DOS BSAVE MSG.

BLOADMSG

EXTERNAL SUBROUTINES:

CHGET = $B1 ;ASOFT CHGET S/R CALL -

CHGOT = $B7 ;GETS NEXT SEQUENTIAL CHR

OR TOKEN - LOADS A-REG FROM LOCN SPECIFIED

"BSAVE ARRAY" AND IS DESIGNED *

CARRY IS CLRd IF *

CHR IS NUMERIC OTHERWISE SET -

Z-FLAG SET IF CHR IS 0 (EOL) OR : (EOS),

OTHERWISE Z-FLAG CLRd -

CHR; CHRGOT LEAVES TXTPTlr UNCHANGED.

ERROR = $D412 ;ASOFT ERROR PROCESSING -

CHECKS ERFLG AND JMS TO

02E5: A9 F9
02E7: 8D F6 03
02EA: A9 02
02EC: 8D F7 03
02EF: 60

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

BLOADMSG

BEX 8D84

ASC "LOAD "

GETS NEXT SEQUENTIAL CHR

CHR IS NUMERIC OTHERWISE SET -

Z-FLAG SET IF CHR IS 0 (EOL) OR : (EOS),

OTHERWISE Z-FLAG CLRd -

CHR; CHRGOT LEAVES TXTPTlr UNCHANGED.

ERROR = $D412 ;ASOFT ERROR PROCESSING -

CHECKS ERFLG AND JMS TO
NAME = $06 ; PGM STORAGE OF
ARRAY NAME GOTTEN FROM ASOFT TEXT.
ARYTAB = $6B ; ASOFT PTR TO
BEGINNING OF ARRAY SPACE.
STREND = $6D ; ASOFT PTR TO
END OF NUMERIC STORAGE.
ARYPTR = $08 ; PGM POINTER USED
TO INCREMENT THROUGH MEMORY.
ARYLEN = $EC ; PGM VARIABLE FOR
LENGTH OF ARRAY.
ARYDIM = $ED ; PGM VARIABLE FOR
NO. OF DIMENSIONS OF ARRAY.
REMSTK = $F8 ; ASOFT STACK PTR SAVED
BEFORE EACH STATEMENT.
ERRNUM = $DE ; ASOFT ERROR CODE STORE

*EXTERNAL PARAMETER STORAGE

* PROGRAM BEGINNING

BLOADPGM JSR SAVE ; SAVE ALL REGS
LDA $0 ; INITIALIZE NAME
STA NAME
STA NAME+1 ; CHECK FOR TOKENIZED LOAD
JSR SYNCHR
GET THE ARRAY NAME
LDA $182
STA CHRGOT
BEQ SYNTAX
AT TXTPTR
STA NAME
JSR CHRGOT
BEQ SYNTAX
CMP $S22
BEQ FINDARRY
JMP IF ONE CHR ARRAYNAME
STA NAME+1
ELSE STORE 2ND CHR
FINDQUOT JSR CHRGOT
AND MOVE TXTPTR TO QUOTE
FINDARRY LDY $0
STA ARYPTR
LDY $0
STA ARYPTR+1
STA ARYPTR
58

032C: B1 00 129 CMPNAME LDA (ARYPTR),Y
032E: C5 06 130 CMP NAME
0330: D0 26 131 BNE NOMATCH
0332: C8 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 ;GET OFFSET TO
033A: B1 08 140 LDA (ARYPTR),Y ;NEXT ARRAY
033C: 18 141 CLC ;NAME AND ADD
033D: 65 08 142 ADC ARYPTR ;TO ARYPTR
033F: 65 EB 143 STA TEMP
0341: CB 144 INY
0342: B1 08 145 LDA (ARYPTR),Y
0344: 65 09 146 ADC ARYPTR+1
0346: 65 09 147 STA ARYPTR+1
0348: A5 EB 148 LDA TEMP
034A: 85 08 149 STA ARYPTR
034C: C5 6D 150 CMP STREND ;CHECK FOR
034E: A5 09 151 LDA ARYPTR+1 ;END OF
0350: E5 6E 152 SBC STREND+1 ;ARRAY STORAGE
0352: B0 07 153 BGE SYNTAX
0354: A0 00 154 LDY #0
0356: F0 D4 155 BEQ CMPNAME ;UNCOND JMP
156
157 *HERE IF 1ST CHR DOESNT MATCH
158
0358: C8 159 NOMATCH INY ;BUMP Y PAST 2ND CHR OF NAME
0359: D0 DE 160 BNE OFFSET ;UNCOND JMP
161
162 *HERE IF ERROR IN STATEMENT SYNTAX
163
035A: A6 F8 164 SYNTAX LDX REMSTK ;RESTORE STACK FROM BEFORE THE &
035B: 9A 165 TXS
035C: A2 10 166 LDX $16 ;SYNTAX ERROR CODE
0360: 4C 12 167 JMP ERROR ;REPORT THE ERROR
168
169 *HERE WHEN ARRAY FOUND
170
0363: A2 00 171 FOUND LDX #0 ;SAVE ARRAY LENGTH
0365: C8 172 FOUNDLP INY ;AND NO. OF DIMENSIONS.
0366: B1 08 173 LDA (ARYPTR),Y
0368: 95 EC 174 STA ARYLEN,X
036A: E8 175 INX
036B: ED 03 176 CPX #3
036D: 90 F6 177 BLT FOUNDLP
036F: A9 F0 178 LDA $BLOADMSG ;SEND BLOAD TO DOS.
0371: A0 02 179 LDY $BLOADMSG
0373: 20 3A DB 180 JSR STROUT
0376: A9 EA 181 LDA $#8EA ;MODIFY CHRGET TO
0377: 85 EA 182 STA $#8C ;ALLOW SPACES
0379: 85 C1 183 STA $#C1
037B: 25 B1 00 184 SENDFILE JSR CHRGET ;GET FILENAME
0377: F0 11 185 BEQ ENDS ENDSEND ;AT TXTPTR
0378: C9 22 186 CMP $#22 ;CMP TO QUOTE
0379: F0 08 187 BEQ FINXTX
0385: 09 80 188 ORA $#80 ;NEG ASCII
0386: E0 149 JSR COUT ;AND SEND
038A: 18 190 CLC ;TO DOS
038B: 90 EF 191 BCC SENDFILE ;UNCOND JMP
038D: 20 B1 00 192 FINXTX JSR CHRGET ;FINISH OUT
0390: D0 F8 193 BNE FINXTX ;TXT TO EOL
0392: A9 F0 194 ENDS ENDSEND LDA $#F0 ;RESTORE CHRGET
U-\text{CU} \quad \text{DIA} \quad \text{DU}

0396: \text{A9} \quad \text{EF} \quad 196 \quad \text{LDA} \quad \text{&&EF}
0398: \text{B5} \quad \text{C1} \quad 197 \quad \text{STA} \quad \text{SC1}
039A: \text{A9} \quad \text{AC} \quad 198 \quad \text{SENDADDR} \quad \text{LDA} \quad \text{\#"A"}; \text{SEND COMMA}
039C: \text{20} \quad \text{ED} \quad \text{FD} \quad 199 \quad \text{JSR} \quad \text{COUT}
039F: \text{A9} \quad \text{C1} \quad 200 \quad \text{LDA} \quad \text{\#A} \quad \text{SEND A}
03A1: \text{20} \quad \text{ED} \quad \text{FD} \quad 201 \quad \text{JSR} \quad \text{COUT}
03A4: \text{A5} \quad \text{08} \quad 202 \quad \text{LDA} \quad \text{ARYPTR} \quad \text{SEND ADDR VIA LINPRT}
03A6: \text{AA} \quad 203 \quad \text{TAX}
03A7: \text{A5} \quad \text{09} \quad 204 \quad \text{LDA} \quad \text{ARYPTR+1}
03A9: \text{20} \quad \text{24} \quad \text{ED} \quad 205 \quad \text{JSR} \quad \text{LINPRT}
03AC: \text{20} \quad \text{8E} \quad \text{FD} \quad 206 \quad \text{JSR} \quad \text{CROUT} \quad \text{SEND CR.}
03AF: \text{A2} \quad \text{00} \quad 207 \quad \text{LDX} \quad \text{\#0} \quad \text{CMP ARRAY LENGTH}
03B1: \text{A0} \quad \text{02} \quad 208 \quad \text{LDY} \quad \text{\#2} \quad \text{AND DIMENSIONS TO}
03B3: \text{B1} \quad \text{08} \quad 209 \quad \text{CMPARYLP} \quad \text{LDA} \quad \text{(ARYPTR),Y}; \text{THOSE SAVEN BEFORE}
03B5: \text{D5} \quad \text{EC} \quad 210 \quad \text{CMP} \quad \text{ARYLEN,X}; \text{THE FILE LOAD.}
03B7: \text{DD} \quad \text{A2} \quad 211 \quad \text{BNE} \quad \text{SYNTAX}
03B9: \text{CB} \quad 212 \quad \text{INX}
03BA: \text{EB} \quad 213 \quad \text{INX}
03BB: \text{E0} \quad \text{03} \quad 214 \quad \text{CPX} \quad \text{\#3}
03BD: \text{90} \quad \text{F4} \quad 215 \quad \text{BLT} \quad \text{CMPARYLP}
03BF: \text{A0} \quad \text{00} \quad 216 \quad \text{LDY} \quad \text{\#0}; \text{NOW RESTORE THE}
03C1: \text{A5} \quad \text{06} \quad 217 \quad \text{LDA} \quad \text{NAME}; \text{ARRAY NAME FROM}
03C3: \text{91} \quad \text{08} \quad 218 \quad \text{STA} \quad \text{(ARYPTR),Y}; \text{BEFORE THE}
03C5: \text{CB} \quad 219 \quad \text{INY} \quad \text{FILE LOAD}
03C6: \text{A5} \quad \text{07} \quad 220 \quad \text{LDA} \quad \text{NAME+1}
03C8: \text{91} \quad \text{08} \quad 221 \quad \text{STA} \quad \text{(ARYPTR),Y}
03CA: \text{4C} \quad \text{3F} \quad \text{FF} \quad 222 \quad \text{JMP} \quad \text{RESTORE}; \text{RETURN VIA RESTORE}
03CB: 223 \quad \text{RUTINE TO RESTORE REGS.}
03CC: 224 \quad \text{NOTE: CODE MUST END BEFORE S3DO}
03CD: 225 \quad \text{BYTES (DOS VECTORS)}.

---END ASSEMBLY---

ERRORS: 0

SYMBOL TABLE - ALPHABETICAL ORDER:

? ARYDIM = $ED ARYLEN = $EC ARYPTR = $08 ARYTAB = $6B
BLOADMSG = $02F0 BLOADPGM = $02F9 CHRGOT = $B1 CHRGOT = $87
CMPARYLP = $03B3 CMPNAME = $032C COUT = $FDED CROUT = $FDBE
ENDSEND = $0392 ? ERNUM = $DE ERROR = $D412 FINDARRY = $0322
FINDQOUT = $0319 FINTXT = $038D FOUND = $0363 FOUNDLP = $0365
LINFRT = $ED24 NAME = $06 NOMATCH = $0356 OFFSET = $0339
RESTK = $F8 RESTORE = $FF3F SAVE = $FF4A ? SENDADDR = $039A
SENDFILE = $037C STREND = $6D STROUT = $DB3A SYNCHR = $DECO
SYNTAX = $035B TEMP = $EB

SYMBOL TABLE - NUMERICAL ORDER:

NAME = $06 ARYPTR = $88 ARYTAB = $6B STREND = $6D
CHRGOT = $B1 CHRGOT = $E7 ? ERNUM = $DE TEMP = $EB
ARYLEN = $EC ? ARYDIM = $ED REMSTK = $F8 BLOADMSG = $02F0
BLOADPGM = $02F9 FINDQOUT = $0319 FINDARRY = $0322 CMPNAME = $032C
OFFSET = $0339 NOMATCH = $0358 SYNTAX = $035B FOUND = $0363
FOUNDLP = $0365 SENDFILE = $037C FINTXT = $038D ENDSSEND = $0392
? SENDADDR = $039A CMPARYLP = $03B3 ERROR = $D412 STROUT = $DB3A
SYNCHR = $DECO LINPRT = $ED24 CROUT = $FDBE 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 CHAR. 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 ;CR,CTRL-D
JEOL
CBRGET $81 1ASOFT
END
RTS

* EXTERNAL SUBROUTINES:

CHRGET = $B1 ;ASOFT CHRGET S/R CALL -
CHRGOT = $B7 ;GETS NEXT SEQUENTIAL CHR
OR TOKEN - LOADS A-REG FROM LOCN SPECIFIED
BY TXTPTR ($B8-$B9) - CARRY IS CLR IF
CHR IS NUMERIC OTHERWISE SET -
Z-FLAG GET IF CHR IS 0 (EOL) OR : (EOS),
OTHERWISE Z-FLAG CLR -
CHRGET INCREMENTS TXTPTR BEFORE GETTING
CHR; CHRGOT LEAVES TXTPTR UNCHANGED.
ERROR = $D412 ;ASOFT ERROR PROCESSING -
CHECKS ERRFLG AND JMP'S TO
HANDLELL IF ONERR ACTIVE
OTHERWISE PRINTS ERROR MSG
030E: 20 4A FF 100 BSAVEPGM JSR SAVE ;SAVE ALL REGS
0311: A9 00 101 LDA $0 ;INITIALIZE NAME
0313: BS 06 102 STA NAME
0315: BS 07 103 STA NAME+1
0317: A9 B7 104 LDA $183 ;CHECK FOR TOKENIZED SAVE
0319: 20 C0 DE 105 JSR SYNCR
031C: 20 B7 00 106 JSR CHRGOT ;GET THE ARRAY NAME
031F: F0 4F 107 BEQ SYNTAX ;AT TXTPTR
0321: BS 06 108 STA NAME
0323: 20 B1 00 109 JSR CHRGET
0326: F0 48 110 BEQ SYNTAX
0328: C9 22 111 CMP $522 ;CMP TO QUOTE
032A: F0 0B 112 BEO FINDARRY ;JMP IF ONE CHR ARRAYNAME
032C: BS 07 113 STA NAME+1 ;ELSE STORE 2ND CHR
032E: 20 B1 00 114 FINDQUOT JSR CHRGET ;AND MOVE TXTPTR TO QUOTE
0331: F0 3D 115 BEQ SYNTAX
0333: C9 22 116 CMP $522 ;CMP TO QUOTE
0335: D0 F7 117 BNE FINDQUOT
0337: A0 00 118 FINDARRY LDY $0 ;ARYPTR <-- ARYTAB
0339: A5 6B 119 LDA ARYTAB
033B: BS 08 120 STA ARYPTR
033D: A5 6C 121 LDA ARYTAB+1
033F: BS 09 122 STA ARYPTR+1
0341: B1 08 123 CMPNAME LDA (ARYPTR),Y
0343: C5 06 124 CMP NAME
0345: D0 26 125 BNE NOMATCH
0347: C8 126 INY
0348: B1 08 127 LDA (ARYPTR),Y
034A: C5 07 128 CMP NAME+1
START HERE IF NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

OFFSET TO

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS

NAME NAME COMPARE FAILS
LDA $-L-

;SEND LENGTH VIA LINPRT

LDA (ARYPTR),Y

TAX

INY

LDA (ARYPTR),Y

JSR LINPRT

JSR CROUT ;SEND CR

JMP RESTORE ;RETURN VIA RESTORE

205 * ROUTINE TO RESTORE REG.

206 *NOTE: CODE MUST END BEFORE $3D0

207 *(DOS VECTORS).

**END ASSEMBLY**

**ERRORS:** 0

**214 BYTES**

**SYMBOL TABLE - ALPHABETICAL ORDER:**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARYPTR</td>
<td>$08</td>
</tr>
<tr>
<td>CBRGET</td>
<td>$B1</td>
</tr>
<tr>
<td>CROUT</td>
<td>$FD8E</td>
</tr>
<tr>
<td>FINDARRY</td>
<td>$0337</td>
</tr>
<tr>
<td>LINPRT</td>
<td>$ED24</td>
</tr>
<tr>
<td>REMSTK</td>
<td>$F8</td>
</tr>
<tr>
<td>SENDFILE</td>
<td>$0385</td>
</tr>
<tr>
<td>SYNTAX</td>
<td>$0370</td>
</tr>
<tr>
<td>ARYPTR</td>
<td>$08</td>
</tr>
<tr>
<td>ARYTAB</td>
<td>$6B</td>
</tr>
<tr>
<td>BSAVEMSG</td>
<td>$0305</td>
</tr>
<tr>
<td>BSAPGM</td>
<td>$030E</td>
</tr>
<tr>
<td>CMPNAME</td>
<td>$0341</td>
</tr>
<tr>
<td>COUT</td>
<td>$FDED</td>
</tr>
<tr>
<td>ENDSSEND</td>
<td>$039B</td>
</tr>
<tr>
<td>ERROR</td>
<td>$D412</td>
</tr>
<tr>
<td>FINTXT</td>
<td>$0396</td>
</tr>
<tr>
<td>NOMATCH</td>
<td>$036D</td>
</tr>
<tr>
<td>OFFSET</td>
<td>$034E</td>
</tr>
<tr>
<td>SENDADDR</td>
<td>$03A3</td>
</tr>
<tr>
<td>SAVE</td>
<td>$FF4A</td>
</tr>
<tr>
<td>STREND</td>
<td>$6D</td>
</tr>
<tr>
<td>STROUT</td>
<td>$DB3A</td>
</tr>
<tr>
<td>SYNCHR</td>
<td>$DECO</td>
</tr>
</tbody>
</table>

**SYMBOL TABLE - NUMERICAL ORDER:**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>$06</td>
</tr>
<tr>
<td>CBRGET</td>
<td>$B1</td>
</tr>
<tr>
<td>CMPNAME</td>
<td>$0341</td>
</tr>
<tr>
<td>ENDSSEND</td>
<td>$039B</td>
</tr>
<tr>
<td>ERRNUM</td>
<td>$DE</td>
</tr>
<tr>
<td>ERROR</td>
<td>$D412</td>
</tr>
<tr>
<td>FINTXT</td>
<td>$0396</td>
</tr>
<tr>
<td>NOMATCH</td>
<td>$036D</td>
</tr>
<tr>
<td>OFFSET</td>
<td>$034E</td>
</tr>
<tr>
<td>SENDADDR</td>
<td>$03A3</td>
</tr>
<tr>
<td>BSAPGM</td>
<td>$030E</td>
</tr>
<tr>
<td>BSAPGM</td>
<td>$030E</td>
</tr>
<tr>
<td>CMPNAME</td>
<td>$0341</td>
</tr>
<tr>
<td>ENDSSEND</td>
<td>$039B</td>
</tr>
<tr>
<td>ERROR</td>
<td>$D412</td>
</tr>
<tr>
<td>FINDARRY</td>
<td>$0337</td>
</tr>
<tr>
<td>FINTXT</td>
<td>$0396</td>
</tr>
<tr>
<td>NOMATCH</td>
<td>$036D</td>
</tr>
<tr>
<td>OFFSET</td>
<td>$034E</td>
</tr>
<tr>
<td>SENDADD</td>
<td>$03A3</td>
</tr>
<tr>
<td>BSAPGM</td>
<td>$030E</td>
</tr>
<tr>
<td>BSAPGM</td>
<td>$030E</td>
</tr>
<tr>
<td>CMPNAME</td>
<td>$0341</td>
</tr>
<tr>
<td>ENDSSEND</td>
<td>$039B</td>
</tr>
<tr>
<td>ERROR</td>
<td>$D412</td>
</tr>
<tr>
<td>FINDARRY</td>
<td>$0337</td>
</tr>
<tr>
<td>FINTXT</td>
<td>$0396</td>
</tr>
<tr>
<td>NOMATCH</td>
<td>$036D</td>
</tr>
<tr>
<td>OFFSET</td>
<td>$034E</td>
</tr>
<tr>
<td>SENDADD</td>
<td>$03A3</td>
</tr>
<tr>
<td>BSAPGM</td>
<td>$030E</td>
</tr>
<tr>
<td>BSAPGM</td>
<td>$030E</td>
</tr>
<tr>
<td>CMPNAME</td>
<td>$0341</td>
</tr>
<tr>
<td>ENDSSEND</td>
<td>$039B</td>
</tr>
<tr>
<td>ERROR</td>
<td>$D412</td>
</tr>
<tr>
<td>FINDARRY</td>
<td>$0337</td>
</tr>
<tr>
<td>FINTXT</td>
<td>$0396</td>
</tr>
<tr>
<td>NOMATCH</td>
<td>$036D</td>
</tr>
<tr>
<td>OFFSET</td>
<td>$034E</td>
</tr>
<tr>
<td>SENDADD</td>
<td>$03A3</td>
</tr>
<tr>
<td>BSAPGM</td>
<td>$030E</td>
</tr>
<tr>
<td>BSAPGM</td>
<td>$030E</td>
</tr>
<tr>
<td>CMPNAME</td>
<td>$0341</td>
</tr>
<tr>
<td>ENDSSEND</td>
<td>$039B</td>
</tr>
<tr>
<td>ERROR</td>
<td>$D412</td>
</tr>
<tr>
<td>FINDARRY</td>
<td>$0337</td>
</tr>
<tr>
<td>FINTXT</td>
<td>$0396</td>
</tr>
<tr>
<td>NOMATCH</td>
<td>$036D</td>
</tr>
<tr>
<td>OFFSET</td>
<td>$034E</td>
</tr>
<tr>
<td>SENDADD</td>
<td>$03A3</td>
</tr>
<tr>
<td>BSAPGM</td>
<td>$030E</td>
</tr>
<tr>
<td>BSAPGM</td>
<td>$030E</td>
</tr>
<tr>
<td>CMPNAME</td>
<td>$0341</td>
</tr>
<tr>
<td>ENDSSEND</td>
<td>$039B</td>
</tr>
<tr>
<td>ERROR</td>
<td>$D412</td>
</tr>
<tr>
<td>FINDARRY</td>
<td>$0337</td>
</tr>
<tr>
<td>FINTXT</td>
<td>$0396</td>
</tr>
<tr>
<td>NOMATCH</td>
<td>$036D</td>
</tr>
<tr>
<td>OFFSET</td>
<td>$034E</td>
</tr>
<tr>
<td>SENDADD</td>
<td>$03A3</td>
</tr>
<tr>
<td>BSAPGM</td>
<td>$030E</td>
</tr>
<tr>
<td>BSAPGM</td>
<td>$030E</td>
</tr>
<tr>
<td>CMPNAME</td>
<td>$0341</td>
</tr>
<tr>
<td>ENDSSEND</td>
<td>$039B</td>
</tr>
<tr>
<td>ERROR</td>
<td>$D412</td>
</tr>
<tr>
<td>FINDARRY</td>
<td>$0337</td>
</tr>
<tr>
<td>FINTXT</td>
<td>$0396</td>
</tr>
<tr>
<td>NOMATCH</td>
<td>$036D</td>
</tr>
<tr>
<td>OFFSET</td>
<td>$034E</td>
</tr>
<tr>
<td>SENDADD</td>
<td>$03A3</td>
</tr>
<tr>
<td>BSAPGM</td>
<td>$030E</td>
</tr>
<tr>
<td>BSAPGM</td>
<td>$030E</td>
</tr>
<tr>
<td>CMPNAME</td>
<td>$0341</td>
</tr>
<tr>
<td>ENDSSEND</td>
<td>$039B</td>
</tr>
<tr>
<td>ERROR</td>
<td>$D412</td>
</tr>
<tr>
<td>FINDARRY</td>
<td>$0337</td>
</tr>
<tr>
<td>FINTXT</td>
<td>$0396</td>
</tr>
<tr>
<td>NOMATCH</td>
<td>$036D</td>
</tr>
<tr>
<td>OFFSET</td>
<td>$034E</td>
</tr>
<tr>
<td>SENDADD</td>
<td>$03A3</td>
</tr>
<tr>
<td>BSAPGM</td>
<td>$030E</td>
</tr>
<tr>
<td>BSAPGM</td>
<td>$030E</td>
</tr>
<tr>
<td>CMPNAME</td>
<td>$0341</td>
</tr>
<tr>
<td>ENDSSEND</td>
<td>$039B</td>
</tr>
<tr>
<td>ERROR</td>
<td>$D412</td>
</tr>
<tr>
<td>FINDARRY</td>
<td>$0337</td>
</tr>
<tr>
<td>FINTXT</td>
<td>$0396</td>
</tr>
</tbody>
</table>
APPENDIX E

MEMORY MAP AND ARRAY FOR SUBPROGRAM ANIMATE

SUBLOGIC DATA SHEET
(Courtesy Sublogic Corp., Champaign, IL)
<table>
<thead>
<tr>
<th>Apple ROMs (BASIC, monitor, etc.)</th>
<th>65535 ($FFFF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk Operating System</td>
<td>49152 ($C000)</td>
</tr>
<tr>
<td>Final Animation Array</td>
<td>38400 ($9600)</td>
</tr>
<tr>
<td>Initial Animation Array</td>
<td>33356 ($824C)</td>
</tr>
<tr>
<td>A2-3D2 Machine Language Driver</td>
<td>33019 ($80FB)</td>
</tr>
<tr>
<td>Data for Animation</td>
<td>24576 ($6000)</td>
</tr>
<tr>
<td>Graphics page 1 for Animation</td>
<td>16384 ($4000)</td>
</tr>
<tr>
<td>BASIC Program Animate</td>
<td>8191 ($2000)</td>
</tr>
<tr>
<td>Secondary text page</td>
<td>3072 ($C00)</td>
</tr>
<tr>
<td>Used by Apple</td>
<td>2048 ($800)</td>
</tr>
<tr>
<td>Used by Apple</td>
<td>0 ($0)</td>
</tr>
</tbody>
</table>
## 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.)</td>
<td>FF,5F,FF,7F,45,25</td>
<td>Field of View</td>
</tr>
<tr>
<td>33048 $8118</td>
<td>14 (SRES)</td>
<td>01</td>
<td>(Set Hi Res 280X192)</td>
</tr>
<tr>
<td>33050 $811A</td>
<td>RESERVED FOR</td>
<td></td>
<td>Output Array Control</td>
</tr>
<tr>
<td>33053 $811D</td>
<td></td>
<td></td>
<td>Line DWG Control</td>
</tr>
<tr>
<td>33055 $811F</td>
<td></td>
<td></td>
<td>Interpretive Jump</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 BO</td>
<td></td>
</tr>
<tr>
<td>33104 $8150</td>
<td>11</td>
<td></td>
<td>No Ops - RESERVED</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></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 ZB</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></td>
</tr>
<tr>
<td>33195 $81AB</td>
<td>RP 03</td>
<td>xyz AR</td>
<td>UNIT VECTOR A</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>SP 01</td>
<td>xyz BT</td>
<td></td>
</tr>
<tr>
<td>33209</td>
<td>CP 02</td>
<td>xyz BN</td>
<td>UNIT VECTOR B</td>
</tr>
<tr>
<td>33216</td>
<td>RP 03</td>
<td>xyz BR</td>
<td></td>
</tr>
<tr>
<td>33223</td>
<td>CP 03</td>
<td>xyz BL</td>
<td></td>
</tr>
<tr>
<td>33230</td>
<td>SP 01</td>
<td>xyz X9</td>
<td></td>
</tr>
<tr>
<td>33237</td>
<td>CP 02</td>
<td>xyz X7</td>
<td>&quot;X&quot;</td>
</tr>
<tr>
<td>33244</td>
<td>SP 01</td>
<td>xyz X8</td>
<td></td>
</tr>
<tr>
<td>33251</td>
<td>CP 02</td>
<td>xyz X6</td>
<td>X ARROWHEAD</td>
</tr>
<tr>
<td>33258</td>
<td>SP 01</td>
<td>xyz X5</td>
<td></td>
</tr>
<tr>
<td>33265</td>
<td>CP 02</td>
<td>xyz XM</td>
<td></td>
</tr>
<tr>
<td>33272</td>
<td>CP 02</td>
<td>xyz X4</td>
<td>X-Y AXES</td>
</tr>
<tr>
<td>33279</td>
<td>SP 01</td>
<td>xyz XM</td>
<td></td>
</tr>
<tr>
<td>33286</td>
<td>CP 02</td>
<td>xyz 0,0,0</td>
<td></td>
</tr>
<tr>
<td>33293</td>
<td>CP 02</td>
<td>xyz YM</td>
<td>X-Y AXES</td>
</tr>
<tr>
<td>33300</td>
<td>RP 03</td>
<td>xyz Y4</td>
<td></td>
</tr>
<tr>
<td>33307</td>
<td>RP 03</td>
<td>xyz Y5</td>
<td></td>
</tr>
<tr>
<td>33314</td>
<td>SP 01</td>
<td>xyz Y9</td>
<td></td>
</tr>
<tr>
<td>33321</td>
<td>CP 02</td>
<td>xyz Y6</td>
<td>&quot;Y&quot;</td>
</tr>
<tr>
<td>33328</td>
<td>RP 03</td>
<td>xyz Y7</td>
<td></td>
</tr>
<tr>
<td>33335</td>
<td>RP 03</td>
<td>xyz Y8</td>
<td></td>
</tr>
<tr>
<td>33342</td>
<td>79</td>
<td></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</td>
</tr>
<tr>
<td>MEM+26 - MEM+33</td>
<td>27 (81b)</td>
<td>2-D Lines</td>
<td>To Create Mechanism Skeleton For jth Position</td>
</tr>
<tr>
<td>MEM+34</td>
<td></td>
<td>No Ops</td>
<td></td>
</tr>
<tr>
<td>MEM+35</td>
<td>06</td>
<td>Pause Command</td>
<td></td>
</tr>
<tr>
<td>MEM+70</td>
<td>06</td>
<td>2-D Lines</td>
<td>2D Lines To Erase jth Position Of Mechanism Skeleton</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Begin 2D Lines For (J + 1)th Position</td>
<td></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...

![A point in 3D space](image1)

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

![A line in 3D space](image2)

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

Standard Features of the A2-3D1
The A2-3D1 program rates "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 ±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.

**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.

---

**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.
## FUNCTION

### Interpretative Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>A2-3D1</th>
<th>A2-3D2</th>
</tr>
</thead>
<tbody>
<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. Pseudodegrees</td>
<td>5.x.y.z.p.b.h</td>
<td>5.x.y.z.p.b.h</td>
</tr>
<tr>
<td>Display Screen Select</td>
<td>6.x.y.x.y</td>
<td>6.x.y.x.y</td>
</tr>
<tr>
<td>Erase Screen / Fill Screen</td>
<td>7.code</td>
<td>7.code</td>
</tr>
<tr>
<td>Write Screen Select</td>
<td>8.code</td>
<td>8.code</td>
</tr>
<tr>
<td>Plot a 2D White Point</td>
<td>9.code</td>
<td>9.code</td>
</tr>
<tr>
<td>Interpretive Jump</td>
<td>10.x.y</td>
<td>10.x.y</td>
</tr>
<tr>
<td>Set Line Drawing Mode</td>
<td>11.adrlsb.adrmsb</td>
<td>11.adrlsb.adrmsb</td>
</tr>
<tr>
<td>Turn on Output Array</td>
<td>12.mode</td>
<td>12.mode</td>
</tr>
<tr>
<td>Screen Size Select</td>
<td>13.adrlsb.adrmsb</td>
<td>13.adrlsb.adrmsb</td>
</tr>
<tr>
<td>Field of View Select</td>
<td>14.w,h,cx,cy</td>
<td>14.w,h,cx,cy</td>
</tr>
<tr>
<td>Easy Initialize</td>
<td>15.axr,ayr,azr</td>
<td>15.axr,ayr,azr</td>
</tr>
<tr>
<td>No Operation</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>Set Color Mode</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>Independent Object Call</td>
<td>18.col</td>
<td>18.col</td>
</tr>
<tr>
<td>Set Resolution</td>
<td>19.statloc.addr</td>
<td>19.statloc.addr</td>
</tr>
<tr>
<td>Hi-Res (280 x 192) Line 2D</td>
<td>20.res</td>
<td>20.res</td>
</tr>
<tr>
<td>Hi-Res Bias</td>
<td>21.xl,xh,y,xl,xh,y</td>
<td>21.xl,xh,y,xl,xh,y</td>
</tr>
<tr>
<td>Hi-Res (x=256 Limited) Line 2D</td>
<td>22.xl,xh,y</td>
<td>22.xl,xh,y</td>
</tr>
<tr>
<td>Hi-Res (280 x 192) Point Plot 2D</td>
<td>23.x,y,z</td>
<td>23.x,y,z</td>
</tr>
<tr>
<td>Hi-Res (x=256 Limited) Point Plot 2D</td>
<td>25.x,x,y</td>
<td>25.x,x,y</td>
</tr>
<tr>
<td>Skip Segment</td>
<td>26.size,stat</td>
<td>26.size,stat</td>
</tr>
<tr>
<td>Pause for n/5ths of a Second</td>
<td>27.time</td>
<td>27.time</td>
</tr>
<tr>
<td>Set 3D to 3D Array Gen Address</td>
<td>28.adrlsb.adrmsb</td>
<td>28.adrlsb.adrmsb</td>
</tr>
<tr>
<td>End of File</td>
<td>29.stat</td>
<td>29.stat</td>
</tr>
<tr>
<td>Callable Functions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sine/Cosine Calls</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Multiply (SP and DP)</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Divide (DP)</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Erase (In/low page)</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Hi-Res Point Plot</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Color Point Plot</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Hi-Res Line Draw</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Color Line Draw</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Set Display Resolution</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

### General Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>A2-3D1</th>
<th>A2-3D2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialize Input Buffer Pointer</td>
<td>32767 max</td>
<td>113508 max*</td>
</tr>
<tr>
<td>Line Length Limit</td>
<td>limited by overflow</td>
<td>unlimited**</td>
</tr>
<tr>
<td>World Movement</td>
<td>2048 + 24576</td>
<td>24576</td>
</tr>
<tr>
<td>Program Location(s)</td>
<td>4864 bytes</td>
<td>8443 bytes</td>
</tr>
</tbody>
</table>

*Distance from -32767 to 32767 is unlimited 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.

---

**subLOGIC**

Communications Corp.
713 Edgebrook Drive
Champaign, IL 61820
(217) 359-8482
Telex: 206995

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


