Summer 1983

Interactive Computer Aided Design and Animation of Spatial Mechanisms

Emmet B. Peter
University of Central Florida

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

NOMENCLATURE

a .....input link, spherical joint position

AO .....input link, pivot (ground) position

A1 .....input link spherical joint initial position

b .....output link, spherical joint position

BO .....output link, pivot (ground) position

B1 .....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 $ØF or ØFH.

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

I. THEORY

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

For the special case of the planar mechanism in three dimensional space, analysis of a general coupler point has meaning, since for this case the coupler can be imagined to be carried by revolute joints rather than spherical ones. It was decided that the general nature of the program could be preserved by allowing for the analysis of a general coupler point and by not restricting a planar mechanism to the X-Y plane, thus figure 2 illustrates the geometry used

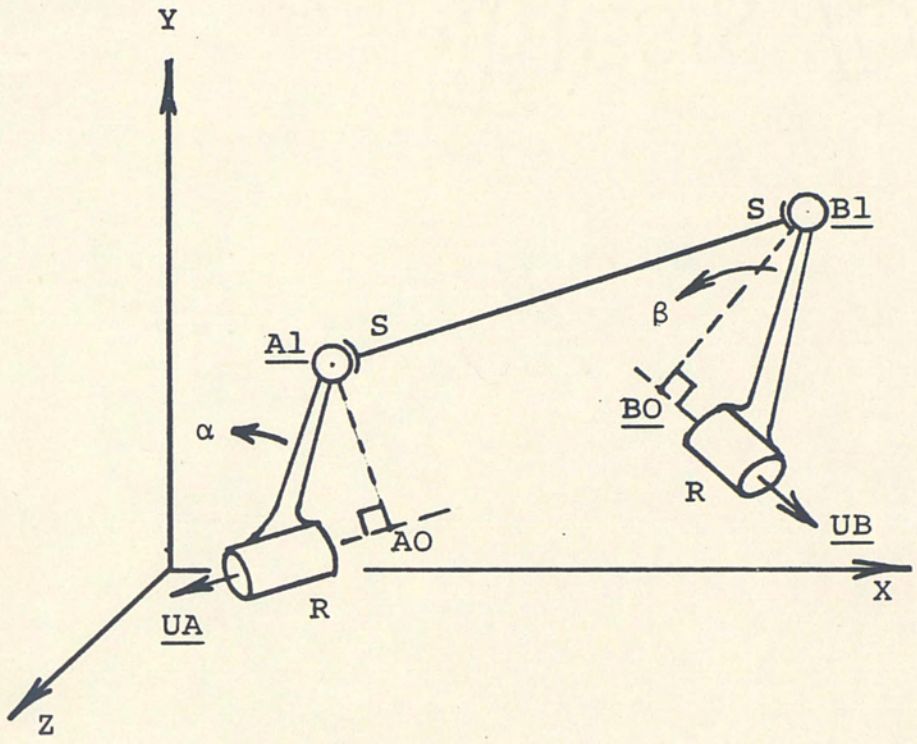


Figure 1. RSSR Conventions

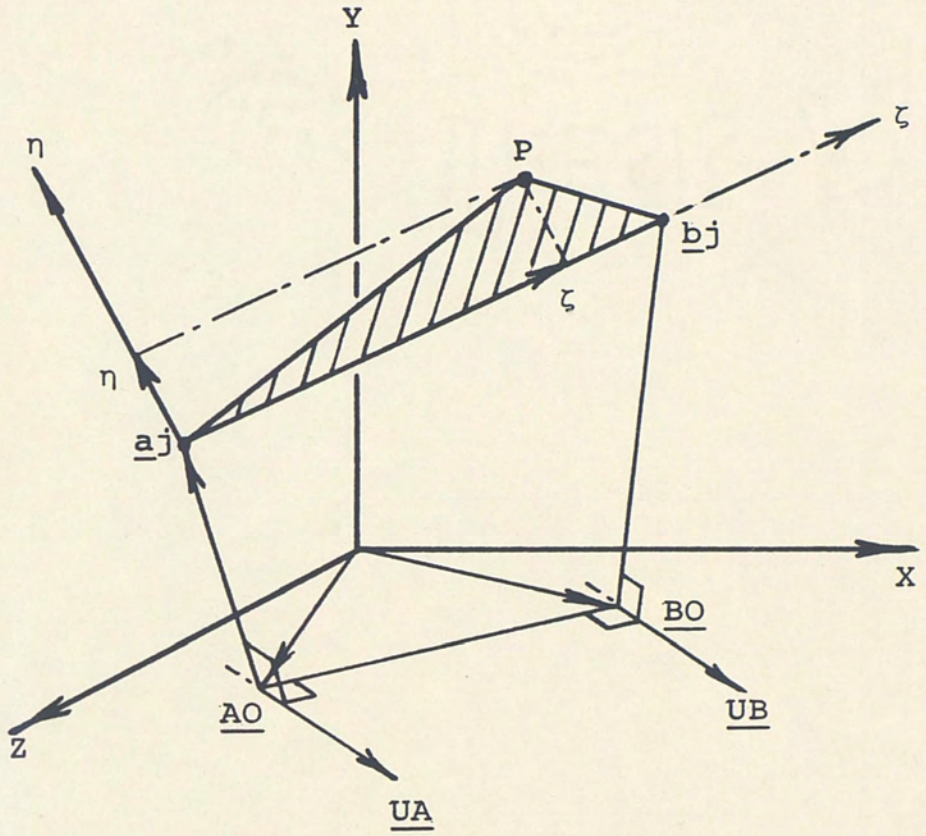


Figure 2. Planar Case

to describe the coupler point. The test for determining whether or not a mechanism is planar is shown in figure 3 and is 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 j^{th} position of the planar case,

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

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

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

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

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

The coupler point $\underline{P}$ can be written as the sum of 3 vectors:

$$\underline{P}_j = \underline{a}_j + \zeta\hat{\zeta}_j + \eta\hat{\eta}_j \quad (2)$$

Substituting (1) for η_j in (2)

$$\underline{P}_j = \underline{a}_j + \zeta\hat{\zeta}_j + \eta[R_{90,UA}]\hat{\zeta}$$

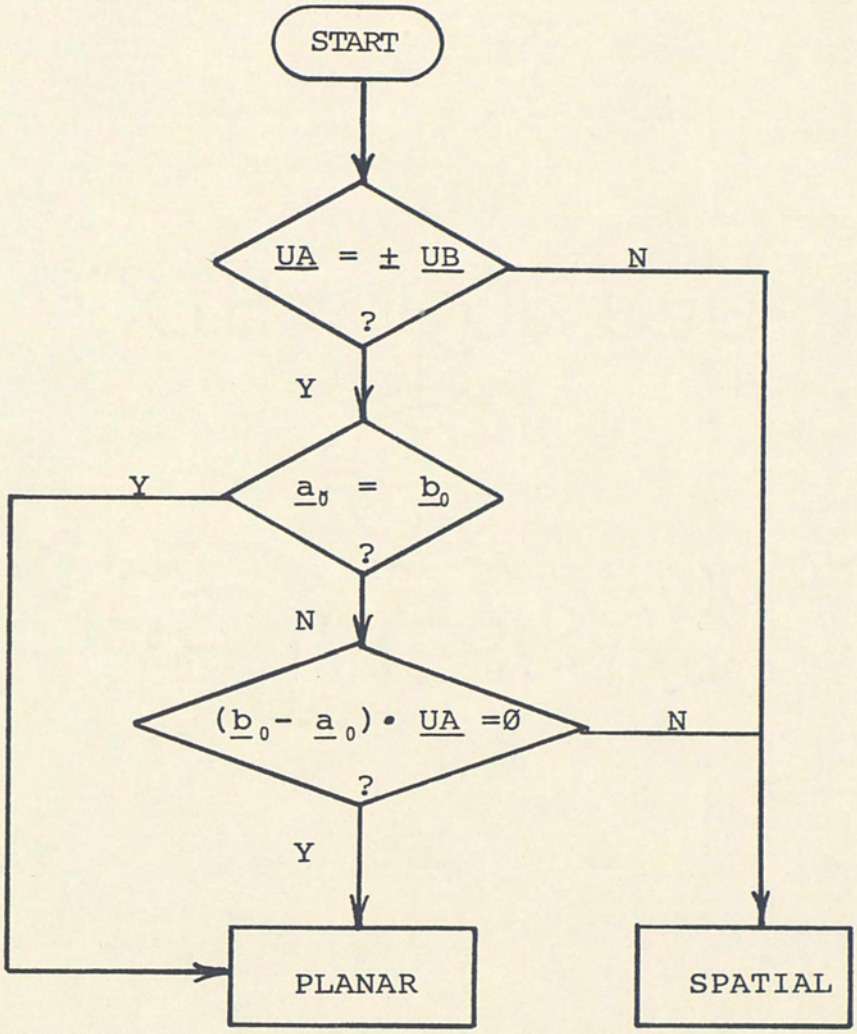


Figure 3. Testing For Planar Inputs

But $\hat{\zeta}_j$ = direction $(\underline{b}_j - \underline{a}_j)$

or $\hat{\zeta}_j = \dfrac{(\underline{b}_j - \underline{a}_j)}{|\underline{b}_j - \underline{a}_j|}$

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

$$\underline{P}_j = \underline{a}_j + \frac{\zeta}{d}(\underline{b}_j - \underline{a}_j) + \frac{\eta}{d}[R_{90,UA}](\underline{b}_j - \underline{a}_j) \tag{3}$$

Taking derivatives to obtain velocity and acceleration,

$$\dot{\underline{P}}_j = \dot{\underline{a}}_j + \frac{\zeta}{d}(\dot{\underline{b}}_j - \dot{\underline{a}}_j) + \frac{\eta}{d}[R_{90,UA}](\dot{\underline{b}}_j - \dot{\underline{a}}_j) \tag{4}$$

$$\ddot{\underline{P}}_j = \ddot{\underline{a}}_j + \frac{\zeta}{d}(\ddot{\underline{b}}_j - \ddot{\underline{a}}_j) + \frac{\eta}{d}[R_{90,UA}](\ddot{\underline{b}}_j - \ddot{\underline{a}}_j) \tag{5}$$

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

II. COMPUTER IMPLEMENTATION

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

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

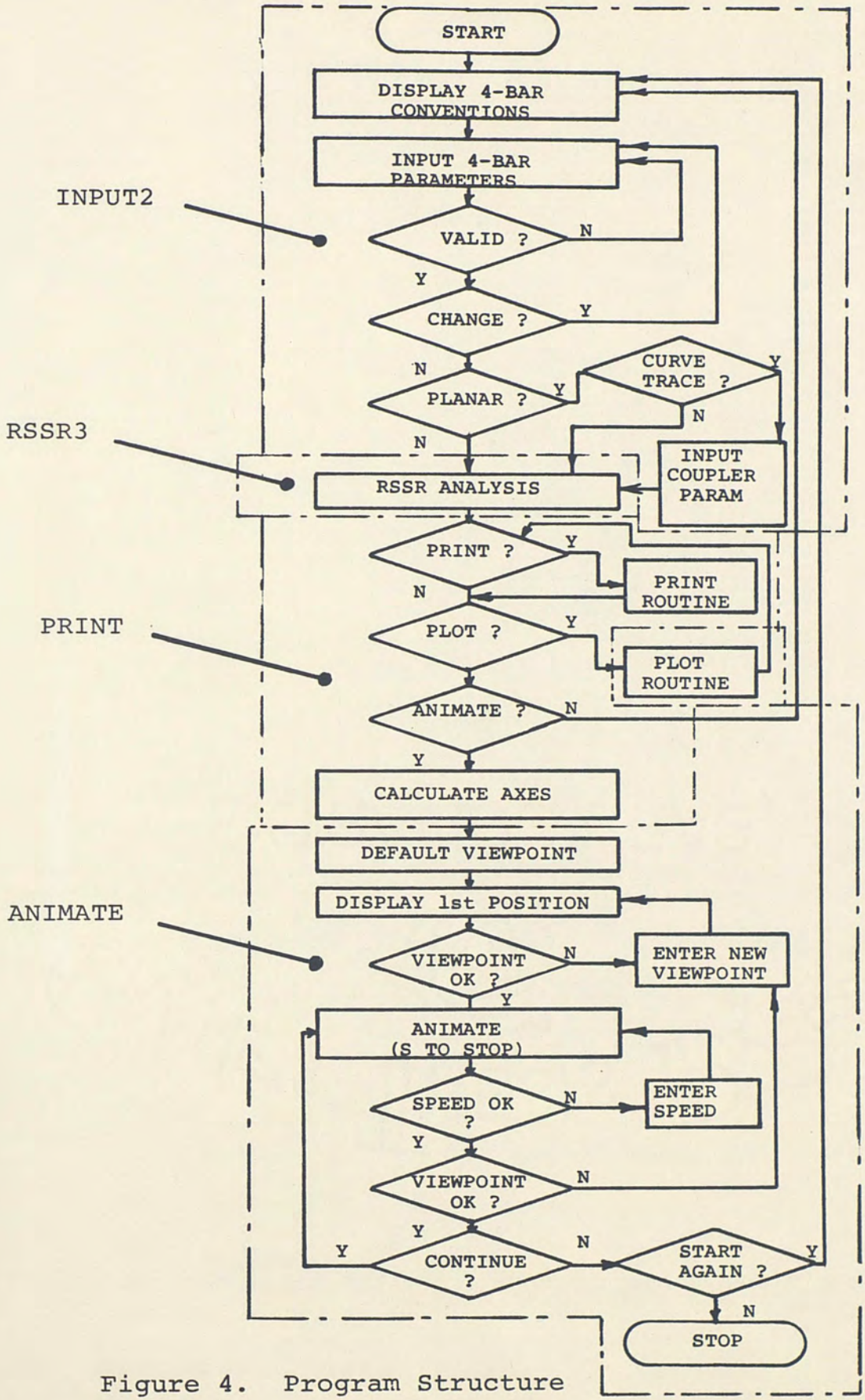


Figure 4. Program Structure

The BASIC Compiler EXPEDITER

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

High Speed Animation

Animation is a means of illustrating movement by displaying discrete, stationary pictures at a high enough

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

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

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

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

The Sublogic graphics package is a commercially available machine language program that resides in a certain location in memory. CALLing the location in BASIC activates the program and causes it to read an array (at

another specified location in memory) containing 3-D point and line information. Using pre-specified viewpoint information, the routine converts the 3-D data into 2-D lines and points and projects them on the screen on command. By creating (using the POKE command) one array in memory containing point and line information for all positions desired, and then CALLing the subroutine, an extremely rapid animation results. After interpreting the array, A2-3D2 returns control to BASIC.

A2-3D2 has many utilities that are helpful. For example, time delays, no-op, and interpretive jump commands can be placed in the array to change the resulting animation. Another utility that proved important was the "exclusive or" line drawing feature of A2-3D2. When a scene consists of a large number of stationary lines and only a few moving ones, much faster animation can be obtained by drawing over lines to erase them rather than erasing the entire screen. This feature is used in drawing the stick figure representing the mechanism: the same figure is drawn again in the exclusive or mode to erase it and prepare for the next frame. The resulting speed for the RSSR mechanism is in excess of 1Ø 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, 1Ø 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 seperate subprogram but since it is called from PRINT, it will be discussed here. When the user desires a plot, he is allowed to choose between a number of mechanism parameters, any of which may be plotted against any other on a two-axis plot. For example, plotting the coupler point x position on the x axis and the coupler point y position on the y axis results in a plot of the coupler curve. Figure 6 illustrates a sample plot. Hardcopy of plots may be obtained if the appropriate printer and graphics "dump" routine are available.

```
****************************************
*                                      *
*              R S S R                 *
*         MECHANISM ANALYSIS           *
*              RESULTS                 *
*                                      *
****************************************
```

PROGRAM INPUTS

INITIAL MECHANISM CONFIGURATION:

POINT A UNIT VECTOR	POINT B UNIT VECTOR
X = .28	X = -.28
Y = .947	Y = .947
Z = .16	Z = .16
POINT AØ	POINT BØ
X = 4	X = 34
Y = Ø	Y = Ø
Z = 6	Z = 6
POINT A1	POINT B1
X = 2.86	X = 35.14
Y = .8Ø7	Y = .8Ø7
Z = 3.22	Z = 3.22

ANGULAR VELOCITY OF INPUT LINK..... 1Ø RAD/SEC
ANGULAR ACCELERATION OF INPUT LINK.. Ø RAD/SEC/SEC
TOTAL TRAVEL OF INPUT LINK..... 4Ø DEGREES
INCREMENT..... 2 DEGREES

RESULTS

* DENOTES NO ASSEMBLY POSSIBLE

ANGLE ALPHA (DEG)	POSITION (DEG)	ANGLE BETA VELOCITY (RAD/SEC)	ACCELERATION (RAD/SEC/SEC)
Ø	Ø	1Ø	-73.3188417
2	1.97471382	9.74840616	-7Ø.7984385
4	3.8999628	9.5Ø4657	-68.8143135
6	5.77713579	9.2670132	-67.2931365
8	7.60729763	9.Ø339662	-66.1753935
1Ø	9.39123149	8.80419584	-65.4122Ø51
12	11.1294719	8.57653755	-64.9629287
14	12.8223345	8.34995697	-64.7933153
16	14.46994	8.12353Ø29	-64.8740802
18	16.Ø722327	7.8964289	-65.1797773
2Ø	17.629ØØ13	7.66790762	-65.6878969
22	19.1398917	7.43729577	-66.3781425
24	2Ø.6Ø44234	7.2Ø399Ø58	-67.2318374
26	22.Ø22ØØ16	6.96745228	-68.2314418
28	23.3919295	6.72720Ø95	-69.36Ø1564
3Ø	24.7134197	6.48281446	-7Ø.6Ø1599
32	25.9856Ø68	6.23392751	-71.9395448
34	27.2Ø75568	5.98Ø23145	-73.357728
36	28.378281	5.72147469	-74.8396957
38	29.4967452	5.45746347	-76.368716
4Ø	3Ø.5618824	5.18Ø86Ø3	-77.9277388

Figure 5. Sample Printout

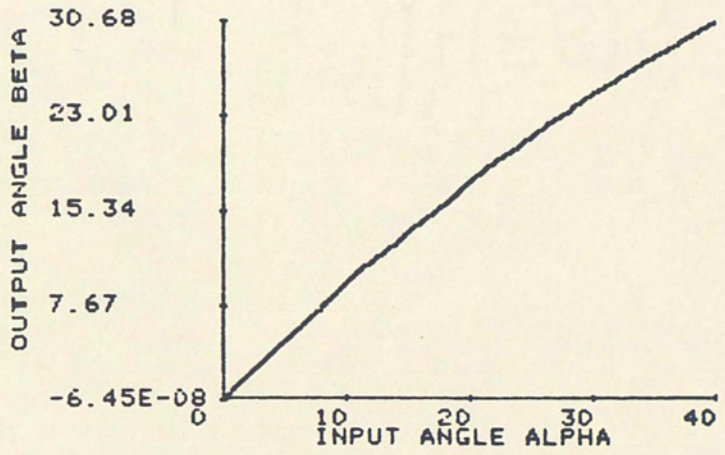


Figure 6. Sample Plot

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

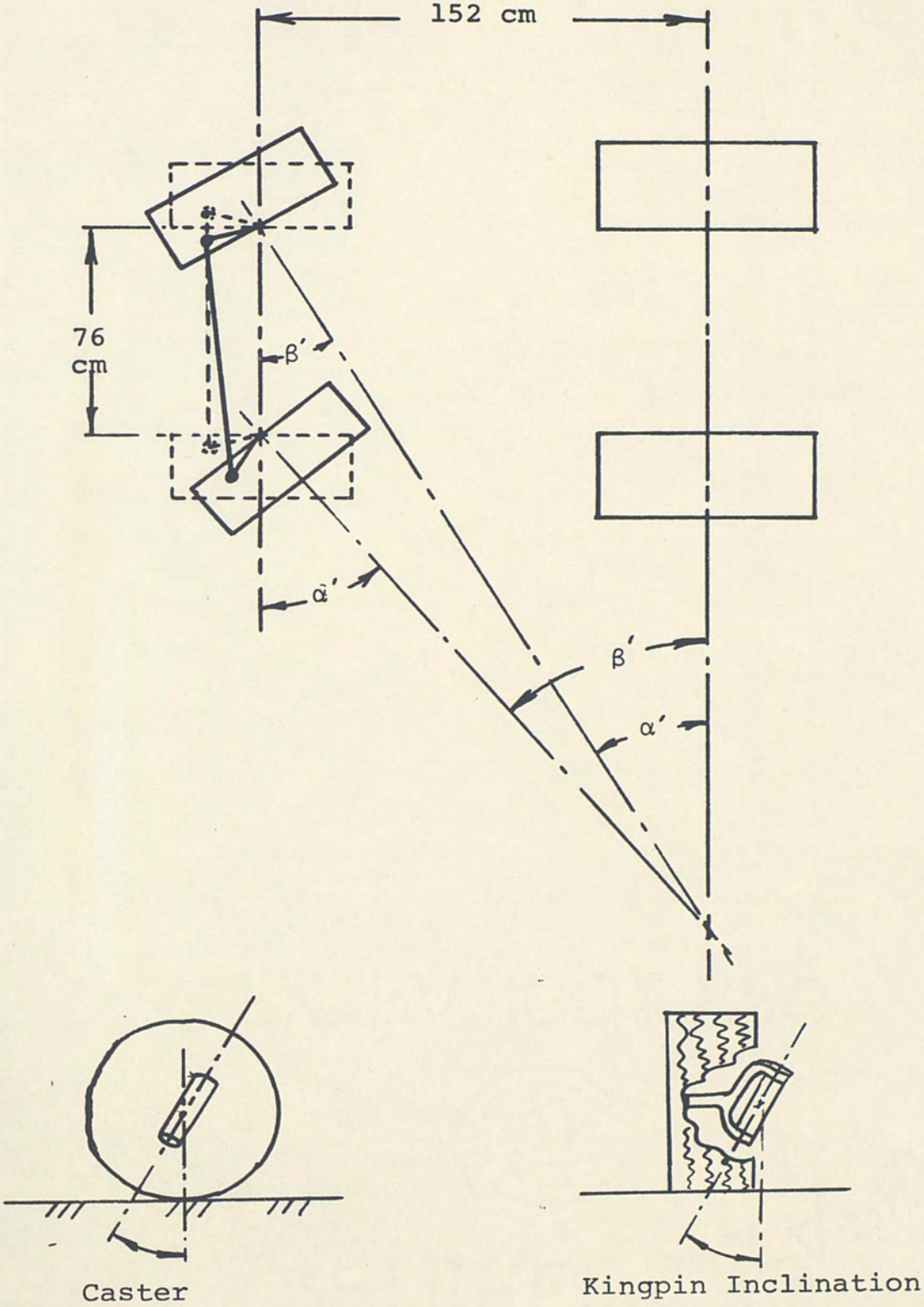


Figure 7. Ackerman Steering Geometry

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 α' and β' are not exact reflections of the mechanism conventions α and β, 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.

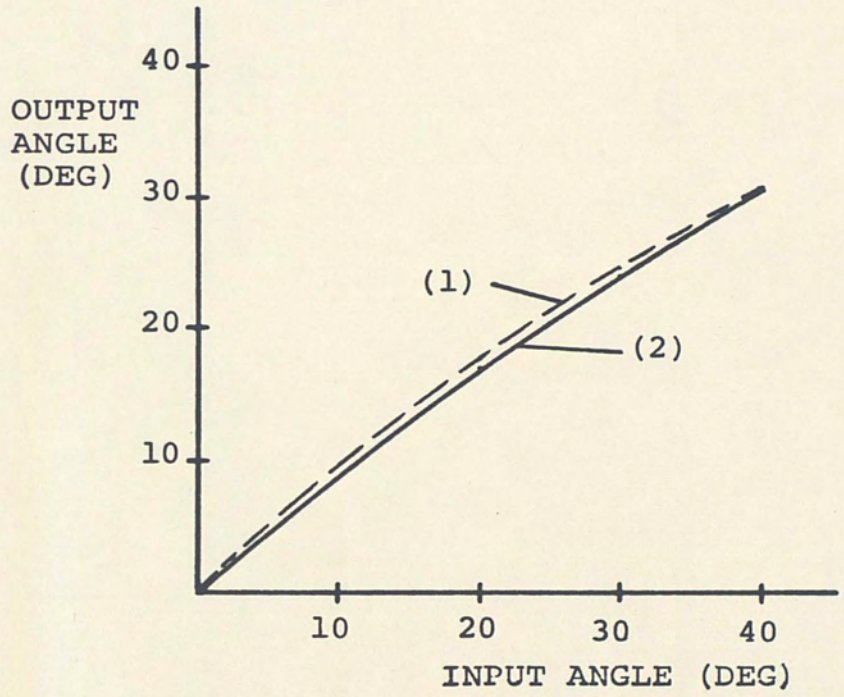


Figure 8. Actual VS Ideal Performance
(1) ACTUAL MECHANISM
(2) IDEAL (DESIGN EQUATION)

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

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:

$$(\underline{a} - \underline{b})^T (\underline{a} - \underline{b}) = (\underline{a}_1 - \underline{b}_1)^T (\underline{a}_1 - \underline{b}_1) \qquad (1)$$

where $\underline{a}$ is given in terms of the specified input angle α from

$$\underline{a} = [R_{\alpha ua}] \quad (\underline{a}_1 - \underline{a}_0) + \underline{a}_0 \qquad (2)$$

and $\underline{b}$ is a function of the unknown output angle β

$$\underline{b} = [R_{\beta ub}] \quad (\underline{b}_1 - \underline{b}_0) + \underline{b}_0 \, . \qquad (3)$$

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

$$[R_{\phi u}] = [I - Q_u] \cos \phi + [P_u] \sin \phi + [Q_u]$$

where

$$[P_u] = \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix}$$

and

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

Substituting (2) and (3) into (1) by writing $(\underline{a} - \underline{b})$ as

$$(\underline{a} - \underline{b}) = (\underline{a} - \underline{b}_0) - [R_{\beta ub}] (\underline{b}_1 - \underline{b}_0)$$

and noting that

$$(\underline{a}_1 - \underline{b}_1)^T (\underline{a}_1 - \underline{b}_1) = (\underline{a} - \underline{b}_0)^T (\underline{a} - \underline{b}_0) + (\underline{b}_1 - \underline{b}_0)^T (\underline{b}_1 - \underline{b}_0) - 2(\underline{a} - \underline{b}_0) [R_{\beta ub}] (\underline{b}_1 - \underline{b}_0)$$

we get

$$E \cos \beta + F \sin \beta + G = \emptyset \qquad (4)$$

where

$$E = (\underline{a} - \underline{b}_0)^T [I - Q_{ub}] (\underline{b}_1 - \underline{b}_0)$$

$$F = (\underline{a} - \underline{b}_0)^T [P_{ub}] (\underline{b}_1 - \underline{b}_0)$$

$$G = (\underline{a} - \underline{b}_0)^T [Q_{ub}] (\underline{b}_1 - \underline{b}_0) + \frac{1}{2} \left\{ (\underline{a}_1 - \underline{b}_1)^T (\underline{a}_1 - \underline{b}_1) - (\underline{a} - \underline{b}_0)^T (\underline{a} - \underline{b}_0) - (\underline{b}_1 - \underline{b}_0)^T (\underline{b}_1 - \underline{b}_0) \right\}.$$

Note that (4) contains one unknown, β, because $\underline{a}$ is known from the specified input angle α (see equation 2). Solution of equation (4) yields two possible values of β, 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 β 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 $\underline{b}$ utilizing this newly found β value.

Velocity Analysis

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

We begin by differentiating the constraint equation (1):

$$(\dot{\underline{a}} - \dot{\underline{b}})^T (\underline{a} - \underline{b}) = \emptyset . \tag{5}$$

Recall that

$$\dot{\underline{a}} = [W_{\dot{\alpha}ua}] \ (\underline{a} - \underline{a}_0)$$

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

$$[W_{\dot{\alpha}ua}] = \frac{d}{dt} [R_{\alpha ua}] = \dot{\alpha}[P_{ua}] .$$

Therefore

$$\dot{\underline{a}} = \dot{\alpha}[P_{ua}] \ (\underline{a} - \underline{a}_0) . \tag{6}$$

Similarily,

$$\dot{\underline{b}} = [W_{\dot{\beta}ub}] \ (\underline{b} - \underline{b}_0) = \dot{\beta}[P_{ub}] \ (\underline{b} - \underline{b}_0) . \tag{7}$$

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

$$\dot{\beta} = \frac{(\dot{a})^T \ (a - b)}{(a - b)^T \ [P_{ub}] \ (b - b_0)} \tag{8}$$

with $\dot{\beta}$ known, we can find $\dot{\underline{b}}$ from (7)

Acceleration Analysis

Again differentiating the constraint equation,

$$(\ddot{\underline{a}} - \ddot{\underline{b}})^T (\underline{a} - \underline{b}) + (\dot{\underline{a}} - \dot{\underline{b}})^T (\dot{\underline{a}} - \dot{\underline{b}}) = \emptyset \qquad (9)$$

where

$$\ddot{\underline{a}} = [\dot{W}_{\alpha\alpha ua}] \quad (\underline{a} - \underline{a}_0)$$

$$\ddot{\underline{a}} = \{\ddot{\alpha}[P_{ua}] + \dot{\alpha}^2[P_{ua}][P_{ua}]\} (\underline{a} - \underline{a}_0). \qquad (10)$$

A similar equation for $\underline{b}$ can be written:

$$\ddot{\underline{b}} = \{\ddot{\beta}[P_{ub}] + \dot{\beta}^2[P_{ub}][P_{ub}]\} (\underline{b} - \underline{b}_0). \qquad (11)$$

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

$$\ddot{\beta} = \frac{(\underline{a}-\underline{b})^T \{\ddot{\underline{a}} - \dot{\beta}^2 [P_{ub}][P_{ub}](\underline{b}-\underline{b}_0)\} + (\dot{\underline{a}}-\dot{\underline{b}})^T(\dot{\underline{a}}-\dot{\underline{b}})}{(\underline{a} - \underline{b})^T [P_{ub}] (\underline{b} - \underline{b}_0)}.$$

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

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

APPENDIX B

DISK ORGANIZATION

DISK ORGANIZATION

The following is a catalog of the Apple II diskette:

```
 A ØØ4 HELLO
 A Ø21 RSSR3
 A Ø15 INPUT2
 A Ø39 PRINT
 T ØØ2 INPUT FILE
 T 123 OUTPUT FILE
*B ØØ3 RBOOT
*B ØØ5 RLOAD
*R Ø12 HRCG
*B ØØ5 ASCII.SET
*B ØØ5 LANDSCAPE.SET
 B ØØ2 BLOAD ARRAY
 B ØØ2 BSAVE ARRAY
 A Ø13 PLOT ROUTINE.C
 A Ø68 RSSR3.OBJ
 T ØØ2 PLOT DESCRIPTOR
 B Ø27 PLOT DATA
 B Ø34 A2-3D2
 B Ø1Ø RSSR CONVENTIONS
 A Ø24 ANIMATE
 B ØØ3 RSSR SKELETON ARRAY
 T Ø56 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: 8191
100  DIM UA(2),AO(2),A1(2),UB(2),B0(2),B1(2),A(2)
120  DIM IN(2,9)
140 D$ = "": REM  CNTRL-D
150  PRINT D$;"BLOAD A2-3D2"
152  PRINT D$;"BLOAD RSSR CONVENTIONS"
153  HGR
154  CALL 24576
155  CALL 24576
160 IN$(0) = "X=":IN$(1) = "Y=":IN$(2) = "Z="
200  PRINT "INPUT PIVOT A UNIT VECTOR...."
220  FOR I = 0 TO 2: PRINT IN$(I): VTAB 23: HTAB 3: INPUT UA(I): NEXT I
225 B =  SQR (UA(0) * UA(0) + UA(1) * UA(1) + UA(2) * UA(2))
230  IF B > .95 THEN  IF B < 1.01 THEN  GOTO 240
235  PRINT "NOT A VALID UNIT VECTOR..REENTER": GOTO 200
240  IF CF% = 1 THEN  GOTO 342
260  PRINT "INPUT THE COORDINATES OF PIVOT A....."
280  FOR I = 0 TO 2: PRINT IN$(I): VTAB 23: HTAB 3: INPUT AO(I): NEXT I
300  IF CF% = 1 THEN  GOTO 342
320  PRINT "INPUT THE INITIAL POSITION OF POINT A1."
340  FOR I = 0 TO 2: PRINT IN$(I): VTAB 23: HTAB 3: INPUT A1(I): NEXT I
341  REM  TEST FOR INPUT COMPATIBILITY
342  FOR I = 0 TO 2:A(I) = A1(I) - AO(I): NEXT I
344 B = 0: FOR I = 0 TO 2:B = B + A(I) * UA(I): NEXT I
346  IF B >  - .95 THEN  IF B < 1.01 THEN  GOTO 360
348  PRINT "AO AND A1 INCOMPATIBLE WITH UA..REENTER": GOTO 200
360  IF CF% = 1 THEN  GOTO 760
380  PRINT "INPUT PIVOT B UNIT VECTOR....."
400  FOR I = 0 TO 2: PRINT IN$(I): VTAB 23: HTAB 3: INPUT UB(I): NEXT I
405 B =  SQR (UB(0) * UB(0) + UB(1) * UB(1) + UB(2) * UB(2))
410  IF B > .95 THEN  IF B < 1.01 THEN  GOTO 420
415  PRINT "NOT A VALID UNIT VECTOR...REENTER": GOTO 380
420  IF CF% = 1 THEN  GOTO 522
440  PRINT "INPUT THE COORDINATES OF PIVOT B....."
460  FOR I = 0 TO 2: PRINT IN$(I): VTAB 23: HTAB 3: INPUT B0(I): NEXT I
480  IF CF% = 1 THEN  GOTO 522
500  PRINT "INPUT THE INITIAL POSITION OF POINT B1"
520  FOR I = 0 TO 2: PRINT IN$(I): VTAB 23: HTAB 3: INPUT B1(I): NEXT I
521  REM  TEST FOR INPUT COMPATIBILITY
522  FOR I = 0 TO 2:A(I) = B1(I) - B0(I): NEXT I
524 B = 0: FOR I = 0 TO 2:B = B + A(I) * UB(I): NEXT I
526  IF B >  - .95 THEN  IF B < 1.01 THEN  GOTO 540
528  PRINT "B0 AND B1 INCOMPATIBLE WITH UB..REENTER": GOTO 380
540  IF CF% = 1 THEN  GOTO 760
560  PRINT "INPUT THE ANGULAR VELOCITY OF LINK 2..": INPUT W2
580  IF CF% = 1 THEN  GOTO 760
600  PRINT "INPUT THE ANGULAR ACCEL. OF LINK 2.....": INPUT A2
620  IF CF% = 1 THEN  GOTO 760
640  PRINT "INPUT THE TOTAL TRAVEL OF THE INPUT LINK DESIRED....": INPUT ALPHA
660  IF CF% = 1 THEN  GOTO 760
680  PRINT "INPUT THE TRAVEL INCREMENT......": INPUT DOTALPHA
682 NPT =  INT (ALPHA / DOTALPHA)
684  IF NPT > 71 THEN  PRINT "REENTER FOR MAXIMUM OF 120 POINTS..": GOTO 680
700  IF CF% = 1 THEN  GOTO 760
705 NPT =  INT (ALPHA / DOTALPHA)
720  PRINT " ": PRINT " "
740  PRINT "WOULD YOU LIKE TO CHANGE ANYTHING": INPUT I1$: GOTO 780
760  PRINT "WOULD YOU LIKE TO CHANGE ANYTHING ELSE": INPUT I1$:CF% = 0
780  IF  LEFT$ (I1$,1) = "Y" THEN  GOTO 840
800  IF  LEFT$ (I1$,1) = "N" THEN  GOTO 1280
820  GOTO 740
```

```
840  REM  WHICH CHANGE?
860 CF% = 1
880  PRINT "WHICH INPUT (TYPE NUMBER TO CHANGE)"
900  VTAB 24
920 I2$ = "UA=1...AO=2...A1=3...UB=4...BO=5...B1=6...ANG VEL OF LINK 2=7...ANG A
CCEL OF LINK 2=8...TOTAL TRAVEL=9...TRAVEL INCRIMENT=10..."
940  VTAB 24: HTAB 1: PRINT  LEFT$ (I2$,39);:I2$ =  MID$ (I2$,2) +  LEFT$ (I2$,1
):K =  PEEK ( - 16384): IF K < 128 THEN  FOR K = 1 TO 75: NEXT K:K =  FRE (0): G
OTO 940
960  INPUT CN%
980  IF CN% > 0 THEN  GOTO 1020
1000  GOTO 940
1020  IF CN% < 11 THEN  GOTO 1060
1040  GOTO 940
1060 CF% = 1
1080  IF CN% = 1 THEN  GOTO 200
1100  IF CN% = 2 THEN  GOTO 260
1120  IF CN% = 3 THEN  GOTO 320
1140  IF CN% = 4 THEN  GOTO 380
1160  IF CN% = 5 THEN  GOTO 440
1180  IF CN% = 6 THEN  GOTO 500
1200  IF CN% = 7 THEN  GOTO 560
1220  IF CN% = 8 THEN  GOTO 600
1240  IF CN% = 9 THEN  GOTO 640
1260  IF CN% = 10 THEN  GOTO 680
1270  REM  TEST FOR PLANAR MECHANISM
1280  IF UA(0) = UB(0) THEN  IF UA(1) = UB(1) THEN  IF UA(2) = UB(2) THEN  GOTO
1340
1300  IF UA(0) =  - UB(0) THEN  IF UA(1) =  - UB(1) THEN  IF UA(2) =  - UB(2) TH
EN  GOTO 1340
1320  GOTO 1560
1340  IF AO(0) = B0(0) THEN  IF AO(1) = B0(1) THEN  IF AO(2) = B0(2) THEN  GOTO
1440
1360  FOR I = 0 TO 2:A(I) = B0(I) - AO(I): NEXT I
1380 B = 0: FOR I = 0 TO 2:B = B + A(I) * UA(I): NEXT I
1400  IF B = 0 THEN  GOTO 1440
1420  GOTO 1560
1440  PRINT "YOU HAVE ENTERED A PLANAR MECHANISM.."
1460  PRINT "WOULD YOU LIKE A COUPLER POINT TRACE ?": INPUT I1$
1480  IF  LEFT$ (I1$,1) = "N" THEN  GOTO 1580
1500  IF  LEFT$ (I1$,1) = "Y" THEN  PRINT "ENTER THE COUPLER POINT PARAMETERS":
PRINT "ZETA=": VTAB 23: HTAB 6: INPUT ZETA: PRINT "ETA=": VTAB 23: HTAB 5: INPUT
 ETA
1515  FOR I = 0 TO 2
1520  IF UA(I) = 0 THEN UA(I) = 1E - 3
1525  NEXT I
1535  FOR I = 0 TO 2
1540  IF UB(I) = 0 THEN UB(I) = 1E - 3
1545  NEXT I
1560  REM  THE MECHANISM IS SPATIAL
1580  FOR I = 0 TO 2:IN(I,0) = UA(I):IN(I,1) = AO(I):IN(I,2) = A1(I):IN(I,3) = U
B(I):IN(I,4) = B0(I):IN(I,5) = B1(I): NEXT I
1600 IN(0,6) = W2:IN(0,7) = A2:IN(0,8) = ALPHA:IN(0,9) = DOTALPHA:IN(1,6) = ZETA
:IN(1,7) = ETA:IN(2,6) = NPT
1620  PRINT D$;"OPEN INPUT FILE"
1640  PRINT D$;"WRITE INPUT FILE"
1660  FOR J = 0 TO 9: PRINT IN(0,J): PRINT IN(1,J): PRINT IN(2,J): NEXT J
1680  PRINT D$;"CLOSE INPUT FILE"
1690  HIMEM: 38399
1700  PRINT D$;"RUN RSSR3.OBJ"
1720  END

]
```

```
 REM SUBPROGRAM RSSR3 SOURCE CODE LISTING

]LIST

20  DIM UA(2),AO(2),A1(2),AJ(2,119),UB(2),B0(2),B1(2),BJ(2,119),VA(2,119),VB(2,1
19),AA(2,119),AB(2,119),BB(2),PM(2,2),QM(2,2),QI(2,2),RM(2,2,1),WM(2,2,0),WD(2,2
,0),T1(2),T2(2),T3(2),T4(2),T5(2),T6(2),T7(2),T8(2),T9(2),TA(2),TB(2)
40  DIM TC(2),AS%(119),VU(2),Z3(119),Z2(119),U(2),DU(2),XV(2),DB(119),DD(119),IN
(2,9)
55 D$ = "": REM  CNTRL-D
60  PRINT D$;"OPEN INPUT FILE": PRINT D$;"READ INPUT FILE"
70  FOR J = 0 TO 9: INPUT IN(0,J): INPUT IN(1,J): INPUT IN(2,J): NEXT J
75  PRINT D$;"CLOSE INPUT FILE"
80  FOR I = 0 TO 2:UA(I) = IN(I,0):AO(I) = IN(I,1):A1(I) = IN(I,2):UB(I) = IN(I,
3):B0(I) = IN(I,4):B1(I) = IN(I,5): NEXT I:W2 = IN(0,6):A2 = IN(0,7):AL = IN(0,8
):DO = IN(0,9)
140  FOR I = 0 TO 2:T1(I) = A1(I) - AO(I):T4(I) = A1(I) - B1(I):T2(I) = B1(I) -
B0(I): NEXT I
165 PM(0,0) = 0:PM(0,1) =  - UB(2):PM(0,2) = UB(1):PM(1,0) = UB(2):PM(1,1) = 0:P
M(1,2) =  - UB(0):PM(2,0) =  - UB(1):PM(2,1) = UB(0):PM(2,2) = 0
170 QM(0,0) = UB(0) * UB(0):QM(0,1) = UB(0) * UB(1):QM(0,2) = UB(0) * UB(2):QM(1
,0) = UB(1) * UB(0):QM(1,1) = UB(1) * UB(1)
175 QM(1,2) = UB(1) * UB(2):QM(2,0) = UB(2) * UB(0):QM(2,1) = UB(2) * UB(1):QM(2
,2) = UB(2) * UB(2)
180  FOR I = 0 TO 2: FOR K = 0 TO 2:QI(I,K) =  - QM(I,K): NEXT K: NEXT I
200 QI(0,0) = 1 + QI(0,0):QI(1,1) = 1 + QI(1,1):QI(2,2) = 1 + QI(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) = 1
350 X1 = J * DU
360  IF X1 - (CV * 180) <  - .020 THEN  GOTO 420
370  IF X1 - (CV * 180) < .010 THEN X1 = CV * 180 - .010
420  FOR I = 0 TO 2:U(I) = UA(I): NEXT I
430 PH = X1
440  GOSUB 4000
460  FOR I = 0 TO 2:UA(I) = U(I): NEXT I
470 X1 = PH
490  FOR I = 0 TO 2:AJ(I,J) = RM(I,0,1) * (A1(0) - AO(0)) + RM(I,1,1) * (A1(1) -
 AO(1)) + RM(I,2,1) * (A1(2) - AO(2)) + AO(I): NEXT I
520  FOR I = 0 TO 2:T5(I) = AJ(I,J) - AO(I):T3(I) = AJ(I,J) - B0(I): NEXT I
690  FOR I = 0 TO 2:T6(I) = 0: FOR K = 0 TO 2:T6(I) = T6(I) + QI(I,K) * T2(K): N
EXT K: NEXT I
750 E = 0: FOR I = 0 TO 2:E = E + T3(I) * T6(I): NEXT I
790  FOR I = 0 TO 2:T7(I) = 0: FOR K = 0 TO 2:T7(I) = T7(I) + PM(I,K) * T2(K): N
EXT K: NEXT I
840 F = 0: FOR I = 0 TO 2:F = F + T3(I) * T7(I): NEXT I
880  FOR I = 0 TO 2:T8(I) = 0: FOR K = 0 TO 2:T8(I) = T8(I) + QM(I,K) * T2(K): N
EXT K: NEXT I
940 G1 = 0: FOR I = 0 TO 2:G1 = G1 + T3(I) * T8(I): NEXT I
950 G2 = 0: FOR I = 0 TO 2:G2 = G2 + T4(I) * T4(I): NEXT I
960 G3 = 0: FOR I = 0 TO 2:G3 = G3 + T3(I) * T3(I): NEXT I
970 G4 = 0: FOR I = 0 TO 2:G4 = G4 + T2(I) * T2(I): NEXT I
980 G = G1 + .5 * (G2 - G3 - G4)
1000 Y1 = E * E + F * F - G * G
1010  IF Y1 < 0 THEN  GOTO 1140
1020 Y1 =  SQR (Y1)
1030 Z1 = G - E
1040  IF  ABS (Z1) < 1E - 10 THEN Z1 = 1E - 10
1050 A1 = 2 *  ATN (( - F - Y1) / Z1)
1060 XA = 2 *  ATN (( - F + Y1) / Z1)
1070 T1 =  ABS (A1 - BE)
1080 T2 =  ABS (XA - BE)
1090 BE = A1
1100  IF T2 < T1 THEN  GOTO 1120
1110  GOTO 1130
```

```
1120 BE = XA
1130  GOTO 1160
1140 AS$(J) = Ø
1150  GOTO 1970
1160  FOR I = Ø TO 2:U(I) = UB(I): NEXT I:PH = BE
1170  FOR I = Ø TO 2:U(I) = UB(I): NEXT I:PH = BE
1180  GOSUB 4000
1200  FOR I = Ø TO 2:UB(I) = U(I): NEXT I:BE = PH
1220  FOR I = Ø TO 2
1230 BJ(I,J) = RM(I,Ø,1) * (B1(Ø) - BØ(Ø)) + RM(I,1,1) * (B1(1) - BØ(1)) + RM(I,
2,1) * (B1(2) - BØ(2)) + BØ(I)
1240  NEXT I
1250 Z2(J) = BE / CV:Z3(J) = X1 / CV
1300  FOR I = Ø TO 2:U(I) = UA(I): NEXT I:VP = W2
1310  GOSUB 5000
1330  FOR I = Ø TO 2:UA(I) = U(I): NEXT I:W2 = VP
1350  FOR I = Ø TO 2:VA(I,J) = Ø: FOR K = Ø TO 2:VA(I,J) = WM(I,K,Ø) * T5(K) + V
A(I,J): NEXT K: NEXT I
1380  FOR I = Ø TO 2
1390 DU(I) = Ø:XV(I) = Ø
1400 TB(I) = AJ(I,J) - BJ(I,J)
1410 T9(I) = BJ(I,J) - BØ(I)
1420  NEXT I
1440  FOR I = Ø TO 2:TA(I) = Ø: FOR K = Ø TO 2:TA(I) = TA(I) + PM(I,K) * T9(K):
NEXT K: NEXT I
1480 VX = Ø: FOR I = Ø TO 2:VX = VX + VA(I,J) * TB(I): NEXT I
1490 TX = Ø: FOR I = Ø TO 2:TX = TX + TB(I) * TA(I): NEXT I
1500 DB(J) = VX / TX
1520  FOR I = Ø TO 2:U(I) = UB(I): NEXT I:VP = DB(J)
1530  GOSUB 5000
1540  FOR I = Ø TO 2:UB(I) = U(I): NEXT I:DB(J) = VP
1560  FOR I = Ø TO 2:VB(I,J) = Ø: FOR K = Ø TO 2:VB(I,J) = WM(I,K,Ø) * T9(K) + V
B(I,J): NEXT K: NEXT I
1590  FOR I = Ø TO 2:U(I) = UA(I):VU(I) = DU(I): NEXT I:VP = W2:AP = A2
1600  GOSUB 7000
1610  FOR I = Ø TO 2:UA(I) = U(I):DU(I) = VU(I): NEXT I:W2 = VP:A2 = AP
1630  FOR I = Ø TO 2:AA(I,J) = Ø: FOR K = Ø TO 2:AA(I,J) = WD(I,K,Ø) * T5(K) + A
A(I,J): NEXT K: NEXT I
1650  FOR I = Ø TO 2:T6(I) = Ø: FOR K = Ø TO 2:T6(I) = T6(I) + QI(I,K) * T9(K):
NEXT K: NEXT I
1670  FOR I = Ø TO 2:T7(I) = Ø: FOR K = Ø TO 2:T7(I) = T7(I) + PM(I,K) * T9(K):
NEXT K: NEXT I
1680  FOR I = Ø TO 2:TC(I) = VA(I,J) - VB(I,J): NEXT I
1720 Z4 = Ø: FOR I = Ø TO 2:Z4 = Z4 + AA(I,J) * TB(I): NEXT I
1730 Z5 = Ø: FOR I = Ø TO 2:Z5 = Z5 + T6(I) * TB(I): NEXT I
1740 Z6 = Ø: FOR I = Ø TO 2:Z6 = Z6 + TC(I) * TC(I): NEXT I
1750 Z7 = Ø: FOR I = Ø TO 2:Z7 = Z7 + TB(I) * T7(I): NEXT I
1760 X7 = Z4 + DB(J) * DB(J) * Z5 + Z6
1770 X8 = Z7
1780 DD(J) = X7 / X8
1800  FOR I = Ø TO 2:U(I) = UB(I):VU(I) = XV(I): NEXT I:VP = DB(J):AP = DD(J)
1810  GOSUB 7000
1820  FOR I = Ø TO 2:UB(I) = U(I):XV(I) = VU(I): NEXT I:DB(J) = VP:DD(J) = AP
1840  FOR I = Ø TO 2:AB(I,J) = Ø: FOR K = Ø TO 2:AB(I,J) = WD(I,K,Ø) * T9(K) + A
B(I,J): NEXT K: NEXT I
1960  PRINT "YOU ARE ";J;"/";NPT;" OF THE WAY THERE"
1970  NEXT J
1972  PRINT D$;"OPEN OUTPUT FILE"
1974  PRINT D$;"WRITE OUTPUT FILE"
1976  FOR I = Ø TO NP: PRINT AJ(Ø,I): PRINT AJ(1,I): PRINT AJ(2,I): NEXT I
1978  FOR I = Ø TO NP: PRINT BJ(Ø,I): PRINT BJ(1,I): PRINT BJ(2,I): NEXT I
1980  FOR I = Ø TO NP: PRINT VA(Ø,I): PRINT VA(1,I): PRINT VA(2,I): NEXT I
1982  FOR I = Ø TO NP: PRINT VB(Ø,I): PRINT VB(1,I): PRINT VB(2,I): NEXT I
1984  FOR I = Ø TO NP: PRINT AA(Ø,I): PRINT AA(1,I): PRINT AA(2,I): NEXT I
1986  FOR I = Ø TO NP: PRINT AB(Ø,I): PRINT AB(1,I): PRINT AB(2,I): NEXT I
1988  FOR I = Ø TO NP: PRINT AS$(I): NEXT I
```

```
1990  FOR I = Ø TO NP: PRINT Z3(I): NEXT I
1992  FOR I = Ø TO NP: PRINT Z2(I): NEXT I
1994  FOR I = Ø TO NP: PRINT DB(I): NEXT I
1996  FOR I = Ø TO NP: PRINT DD(I): NEXT I
1998  PRINT D$;"CLOSE OUTPUT FILE"
2000  PRINT D$;"RUN PRINT"
4000 C =  COS (PH):S =  SIN (PH):V = 1 - C
4040 RM(Ø,Ø,1) = U(Ø) * U(Ø) * V + C
4050 RM(Ø,1,1) = U(Ø) * U(1) * V - U(2) * S
4060 RM(Ø,2,1) = U(Ø) * U(2) * V + U(1) * S
4070 RM(1,Ø,1) = U(Ø) * 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(Ø) * S
4100 RM(2,Ø,1) = U(Ø) * U(2) * V - U(1) * S
4110 RM(2,1,1) = U(1) * U(2) * V + U(Ø) * S
4120 RM(2,2,1) = U(2) * U(2) * V + C
4130  RETURN
5000 WM(Ø,Ø,Ø) = Ø:WM(1,1,Ø) = Ø:WM(2,2,Ø) = Ø
5010 WM(Ø,Ø,Ø) = Ø:WM(1,1,Ø) = Ø:WM(2,2,Ø) = Ø
5020 WM(Ø,1,Ø) =  - U(2) * VP
5030 WM(Ø,2,Ø) = U(1) * VP
5040 WM(1,Ø,Ø) =  - WM(Ø,1,Ø)
5050 WM(1,2,Ø) =  - U(Ø) * VP
5060 WM(2,Ø,Ø) =  - WM(Ø,2,Ø)
5070 WM(2,1,Ø) =  - WM(1,2,Ø)
5080  RETURN
7000 WD(Ø,Ø,Ø) = (U(Ø) * U(Ø) - 1) * VP * VP
7010 WD(Ø,Ø,Ø) = (U(Ø) * U(Ø) - 1) * VP * VP
7020 WD(Ø,1,Ø) = U(Ø) * U(1) * VP * VP - VU(2) * VP - U(2) * AP
7030 WD(Ø,2,Ø) = U(2) * U(Ø) * VP * VP + VU(1) * VP + U(1) * AP
7040 WD(1,Ø,Ø) = U(Ø) * U(1) * VP * VP + VU(2) * VP + U(2) * AP
7050 WD(1,1,Ø) = (U(1) * U(1) - 1) * VP * VP
7060 WD(1,2,Ø) = U(1) * U(2) * VP * VP - VU(Ø) * VP - U(Ø) * AP
7070 WD(2,Ø,Ø) = U(2) * U(Ø) * VP * VP - VU(1) * VP - U(1) * AP
7080 WD(2,1,Ø) = U(2) * U(1) * VP * VP - VU(Ø) * VP + U(Ø) * AP
7090 WD(2,2,Ø) = (U(2) * U(2) - 1) * VP * VP
7100  RETURN

]
```

```
 REM SUBPROGRAM PRINT

]LIST

90  REM  PROGRAM PRINT
100  DIM AJ(2,71),BJ(2,71),VA(2,71),VB(2,71),AA(2,71),AB(2,71),AS$(71),Z3(71),Z2
(71),DB(71),DD(71),UA(2),AO(2),A1(2),UB(2),B0(2),B1(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)
,X5(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 = 0 TO 9: INPUT IN(0,J): INPUT IN(1,J): INPUT IN(2,J): NEXT J
200  PRINT D$;"CLOSE INPUT FILE"
220  FOR I = 0 TO 2:UA(I) = IN(I,0):AO(I) = IN(I,1):A1(I) = IN(I,2):UB(I) = IN(I
,3):B0(I) = IN(I,4):B1(I) = IN(I,5): NEXT I:W2 = IN(0,6):A2 = IN(0,7):AL = IN(0,
8):DO = IN(0,9):ZETA = IN(1,6):ETA = IN(1,7)
230 NP =  INT (AL / DO)
240  PRINT D$;"OPEN OUTPUT FILE": PRINT D$;"READ OUTPUT FILE"
260  FOR I = 0 TO NP: INPUT AJ(0,I): INPUT AJ(1,I): INPUT AJ(2,I): NEXT I
280  FOR I = 0 TO NP: INPUT BJ(0,I): INPUT BJ(1,I): INPUT BJ(2,I): NEXT I
300  FOR I = 0 TO NP: INPUT VA(0,I): INPUT VA(1,I): INPUT VA(2,I): NEXT I
320  FOR I = 0 TO NP: INPUT VB(0,I): INPUT VB(1,I): INPUT VB(2,I): NEXT I
340  FOR I = 0 TO NP: INPUT AA(0,I): INPUT AA(1,I): INPUT AA(2,I): NEXT I
360  FOR I = 0 TO NP: INPUT AB(0,I): INPUT AB(1,I): INPUT AB(2,I): NEXT I
380  FOR I = 0 TO NP: INPUT AS$(I): NEXT I
400  FOR I = 0 TO NP: INPUT Z3(I): NEXT I
420  FOR I = 0 TO NP: INPUT Z2(I): NEXT I
440  FOR I = 0 TO NP: INPUT DB(I): NEXT I
460  FOR I = 0 TO NP: INPUT DD(I): NEXT I
480  PRINT D$;"CLOSE OUTPUT FILE"
600  REM  COUPLER POINT PARAMETERS
620  REM  COMPUTE THE ROTATION MATRIX
630  FOR I = 0 TO 2:U(I) = UA(I): NEXT I
640  GOSUB 14000
660  FOR I = 0 TO 2:S(I) = B1(I) - A1(I): NEXT I
680 SU = 0: FOR I = 0 TO 2:SU = SU + S(I) * S(I): NEXT I
700 D =  SQR (SU)
720 ZD = ZETA / D:ED = ETA / D
730  IF ZETA = 0 THEN  IF ETA = 0 THEN  GOTO 1000
740  REM  LOOP FOR COUPLER POINTS
750  FOR J = 0 TO NP
760  REM  POSITION OF COUPLER POINT J
780  FOR I = 0 TO 2
800 P(I,J) = AJ(I,J) + ZD * (BJ(I,J) - AJ(I,J)) + ED * (R(I,0) * (BJ(0,J) - AJ(0
,J)) + R(I,1) * (BJ(1,J) - AJ(1,J)) + R(I,2) * (BJ(2,J) - AJ(2,J)))
820  NEXT I
840  REM  VELOCITY OF COUPLER POINT J
860  FOR I = 0 TO 2
870 DP(I,J) = VA(I,J) + ZD * (VB(I,J) - VA(I,J)) + ED * (R(I,0) * (VB(0,J) - VA(
0,J)) + R(I,1) * (VB(1,J) - VA(1,J)) + R(I,2) * (VB(2,J) - VA(2,J)))
880  NEXT I
900  REM  ACCELERATION OF COUPLER POINT J
920  FOR I = 0 TO 2
930 P2(I,J) = AA(I,J) + ZD * (AB(I,J) - AA(I,J)) + ED * (R(I,0) * (AB(0,J) - AA(
0,J)) + R(I,1) * (AB(1,J) - AA(1,J)) + R(I,2) * (AB(2,J) - AA(2,J)))
940  NEXT I
960  NEXT J
1000  REM  PRINT, PLOT, OR ANIMATE?
1020  PRINT "WOULD YOU LIKE A PRINT OF THE RESULTS": VTAB 23: HTAB 39: INPUT I1$

1040  IF  LEFT$ (I1$,1) = "Y" THEN  GOTO 5000
1060  IF  LEFT$ (I1$,1) = "N" THEN  PRINT "WOULD YOU LIKE A PLOT OF THE RESULTS"
```

```
: VTAB 23: HTAB 39: INPUT I1$
1080  IF  LEFT$ (I1$,1) = "Y" THEN  GOTO 10000
1100  IF  LEFT$ (I1$,1) = "N" THEN  PRINT "WOULD YOU LIKE TO SEE AN ANIMATION":
VTAB 23: HTAB 37: INPUT I1$
1120  IF  LEFT$ (I1$,1) = "Y" THEN  GOTO 12000
1140  IF  LEFT$ (I1$,1) = "N" THEN  PRINT D$;"RUN INPUT2
1160  GOTO 1040
5000  REM  PRINT ROUTINE
5020  REM  PRINTER IN SLOT 1,AUTO LINE FEED ON, 80 COL.
5045 D$ = "": REM  CNTRL-D
5060  PRINT D$;"PR#1"
5080  POKE 1657,80: REM  AUTHOR'S SETUP ONLY
5100  PRINT "                    ****************************************"
5120  PRINT "                    *                                      *"
5130  PRINT "                    *                R S S R               *"
5140  PRINT "                    *          MECHANISM ANALYSIS          *"
5150  PRINT "                    *                RESULTS               *"
5160  PRINT "                    *                                      *"
5170  PRINT "                    ****************************************"
5180  PRINT : PRINT
5190  PRINT "                                  PROGRAM INPUTS"
5200  PRINT "                                 ---------------"
5210  PRINT
5220  PRINT "                          INITIAL MECHANISM CONFIGURATION:"
5230  PRINT
5240  PRINT "            POINT A UNIT VECTOR                    POINT B UNIT VEC
TOR"
5250  PRINT  SPC( 14);"X = "UA(0); SPC( 53 -  POS (0));"X = ";UB(0)
5260  PRINT  SPC( 14);"Y = "UA(1); SPC( 53 -  POS (0));"Y = ";UB(1)
5270  PRINT  SPC( 14);"Z = "UA(2); SPC( 53 -  POS (0));"Z = ";UB(2)
5280  PRINT
5290  PRINT "            POINT A0                               POINT B0"
5300  PRINT  SPC( 14);"X = "AO(0); SPC( 53 -  POS (0));"X = ";B0(0)
5310  PRINT  SPC( 14);"Y = "AO(1); SPC( 53 -  POS (0));"Y = ";B0(1)
5320  PRINT  SPC( 14);"Z = "AO(2); SPC( 53 -  POS (0));"Z = ";B0(2)
5330  PRINT
5340  PRINT "            POINT A1                               POINT B1"
5350  PRINT  SPC( 14);"X = "A1(0); SPC( 53 -  POS (0));"X = ";B1(0)
5360  PRINT  SPC( 14);"Y = "A1(1); SPC( 53 -  POS (0));"Y = ";B1(1)
5370  PRINT  SPC( 14);"Z = "A1(2); SPC( 53 -  POS (0));"Z = ";B1(2)
5380  PRINT
5390  PRINT "          ANGULAR VELOCITY OF INPUT LINK..... ";W2;" RAD/SEC"
5400  PRINT "          ANGULAR ACCELERATION OF INPUT LINK.. ";A2;" RAD/SEC/SEC"
5410  PRINT "          TOTAL TRAVEL OF INPUT LINK......... ";AL;" DEGREES"
5420  PRINT "          INCREMENT.......................... ";DO;" DEGREES"
5430  IF ZETA = 0 THEN  IF ETA = 0 THEN  GOTO 5490
5440  PRINT
5450  PRINT "          THESE INPUTS INDICATE A PLANAR MECHANISM."
5460  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);"ACCELER
ATION"
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 (
```

```
Ø));"*"; SPC( 23 -  POS (Ø));"*": GOTO 562Ø
5610  PRINT Z3(I); SPC( 21 -  POS (Ø));Z2(I); SPC( 42 -  POS (Ø));DB(I); SPC( 23
 -  POS (Ø));DD(I)
5620  NEXT I
5630 ST$ = "POINT A1 POSITION"
5640  GOSUB 6500
5680  FOR I = Ø TO NP
5690  GOSUB 6800
5695  IF F% = 1 THEN  GOTO 5710
5700  PRINT Z3(I); SPC( 21 -  POS (Ø));AJ(Ø,I); SPC( 42 -  POS (Ø));AJ(1,I); SPC
( 23 -  POS (Ø));AJ(2,I)
5710  NEXT I
5715 ST$ = "POINT B1 POSITION"
5720  GOSUB 6500
5755  FOR I = Ø TO NP
5760  GOSUB 6800
5765  IF F% = 1 THEN  GOTO 5780
5770  PRINT Z3(I); SPC( 21 -  POS (Ø));BJ(Ø,I); SPC( 42 -  POS (Ø));BJ(1,I); SPC
( 23 -  POS (Ø));BJ(2,I)
5780  NEXT I
5790 ST$ = "POINT A1 VELOCITY"
5800  GOSUB 6500
5840  FOR I = Ø TO NP
5850  GOSUB 6800
5855  IF F% = 1 THEN  GOTO 5870
5860  PRINT Z3(I); SPC( 21 -  POS (Ø));VA(Ø,I); SPC( 42 -  POS (Ø));VA(1,I); SPC
( 23 -  POS (Ø));VA(2,I)
5870  NEXT I
5875 ST$ = "POINT B1 VELOCITY"
5880  GOSUB 6500
5915  FOR I = Ø TO NP
5920  GOSUB 6800
5925  IF F% = 1 THEN  GOTO 5940
5930  PRINT Z3(I); SPC( 21 -  POS (Ø));VB(Ø,I); SPC( 42 -  POS (Ø));VB(1,I); SPC
( 23 -  POS (Ø));VB(2,I)
5940  NEXT I
5950 ST$ = "POINT A1 ACCELERATION"
5960  GOSUB 6500
6000  FOR I = Ø TO NP
6010  GOSUB 6800
6015  IF F% = 1 THEN  GOTO 6030
6020  PRINT Z3(I); SPC( 21 -  POS (Ø));AA(Ø,I); SPC( 42 -  POS (Ø));AA(1,I); SPC
( 23 -  POS (Ø));AA(2,I)
6030  NEXT I
6035 ST$ = "POINT B1 ACCELERATION"
6040  GOSUB 6500
6075  FOR I = Ø TO NP
6080  GOSUB 6800
6085  IF F% = 1 THEN  GOTO 6100
6090  PRINT Z3(I); SPC( 21 -  POS (Ø));AB(Ø,I); SPC( 42 -  POS (Ø));AB(1,I); SPC
( 23 -  POS (Ø));AB(2,I)
6100  NEXT I
6105  IF ZETA = Ø THEN  IF ETA = Ø THEN  GOTO 6990
6110 ST$ = "COUPLER POINT POSITION"
6120  GOSUB 6500
6160  FOR I = Ø TO NP
6170  GOSUB 6800
6175  IF F% = 1 THEN  GOTO 6190
6180  PRINT Z3(I); SPC( 21 -  POS (Ø));P(Ø,I); SPC( 42 -  POS (Ø));P(1,I); SPC(
23 -  POS (Ø));P(2,I)
6190  NEXT I
6195 ST$ = "COUPLER POINT VELOCITY"
6200  GOSUB 6500
6235  FOR I = Ø TO NP
6240  GOSUB 6800
6245  IF F% = 1 THEN  GOTO 6260
```

```
6250  PRINT Z3(I); SPC( 21 -  POS (Ø));DP(Ø,I); SPC( 42 -  POS (Ø));DP(1,I); SPC
( 23 -  POS (Ø));DP(2,I)
6260  NEXT I
6265 ST$ = "COUPLER POINT ACCELERATION"
6270  GOSUB 6500
6305  FOR I = Ø TO NP
6310  GOSUB 6800
6315  IF F% = 1 THEN  GOTO 6330
6320  PRINT Z3(I); SPC( 21 -  POS (Ø));P2(Ø,I); SPC( 42 -  POS (Ø));P2(1,I); SPC
( 23 -  POS (Ø));P2(2,I)
6330  NEXT I
6340  GOTO 6990
6500  PRINT : PRINT : PRINT
6520  PRINT "ANGLE"; SPC( 32);ST$
6540  PRINT "ALPHA"
6560  PRINT "(DEG)"; SPC( 20);"X"; SPC( 20);"Y"; SPC( 20);"Z"
6580  PRINT "-------------------------------------------------------------------
-------------"
6600  RETURN
6800 F% = Ø: IF AS%(I) = Ø THEN  PRINT Z3(I); SPC( 21 -  POS (Ø));"*"; SPC( 42 -
  POS (Ø));"*"; SPC( 23 -  POS (Ø));"*":F% = 1: RETURN
6820  RETURN
6990  PRINT D$;"PR#Ø"
7000  PRINT "END OF PRINT ROUTINE"
7020  PRINT "WOULD YOU LIKE PLOTS OF THE RESULTS": VTAB 23: HTAB 38: INPUT I1$:
GOTO 1080
10000  REM  BUILD PLOT DATA FILE
10020  FOR I = Ø TO NP
10040 PD(Ø,I) = Z3(I)
10060 PD(1,I) = Z2(I)
10080 PD(2,I) = BJ(Ø,I)
10100 PD(3,I) = BJ(1,I)
10120 PD(4,I) = BJ(2,I)
10140 PD(5,I) = DB(I)
10160 PD(6,I) = DD(I)
10180 PD(7,I) = P(Ø,I)
10200 PD(8,I) = P(1,I)
10220 PD(9,I) = P(2,I)
10260  NEXT I
11000  REM  BUILD PLOT DESCRIPTOR FILE
11020  PRINT D$;"OPEN PLOT DESCRIPTOR"
11040  PRINT D$;"WRITE PLOT DESCRIPTOR"
11060  PRINT 9: REM  NO. OF VARIABLES
11080  PRINT 71: REM   MAX NO. OF              POINTS
11100  PRINT NP: REM  ACTUAL NO. OF POINTS
11120  PRINT "INPUT ANGLE ALPHA"
11140  PRINT "OUTPUT ANGLE BETA"
11160  PRINT "POS. OF PT B X"
11180  PRINT "POS. OF PT B Y"
11200  PRINT "POS. OF PT B Z"
11220  PRINT "OUTPUT ANGULAR VEL"
11240  PRINT "OUTPUT ANGULAR ACC"
11260  PRINT "COUPLER PT X POS"
11280  PRINT "COUPLER PT Y POS"
11300  PRINT "COUPLER PT Z POS"
11340  PRINT D$;"CLOSE PLOT DESCRIPTOR"
11360  PRINT D$;"BRUN BSAVE ARRAY"
11380  &  SAVE PD"PLOT DATA"
11390  PRINT D$;"RUN PLOT ROUTINE.C"
11400  END
12000  REM  BUILD DATA FOR ANIMATION
12010  REM  FIND SHORTEST LINK
12020  FOR I = Ø TO 2:S(I) = A1(I) - AO(I): NEXT I
12040 SU = Ø: FOR I = Ø TO 2:SU = SU + S(I) * S(I): NEXT I
12060 LA =  SQR (SU)
12080  FOR I = Ø TO 2:S(I) = B1(I) - BØ(I): NEXT I
```

```
12100 SU = 0: FOR I = 0 TO 2:SU = SU + S(I) * S(I): NEXT I
12120 LB =  SQR (SU)
12140 LM = D: IF LA < LM THEN LM = LA
12160  IF LB < LM THEN LM = LB
12180 LF = LM / 2:LA = LM / 4
12200  REM  CALCULATE THE ARROW BODIES
12220  FOR I = 0 TO 2
12240 AN(I) = AO(I) + UA(I) * LF
12260 TA(I) = AO(I) - UA(I) * LF
12280 BN(I) = B0(I) + UB(I) * LF
12300 BT(I) = B0(I) - UB(I) * LF
12320  NEXT I
12330 AN(2) =  - AN(2):TA(2) =  - TA(2):BN(2) =  - BN(2):BT(2) =  - BT(2)
12340  REM  CALCULATE ARROWHEADS
12360 K1 = 2 * LF / 3:K2 = LF / 6
12380  REM  ARROWHEAD FOR UNIT VECTOR A
12400 U(0) = UA(2):U(1) = 0:U(2) =  - UA(0)
12420  GOSUB 14000
12440  FOR I = 0 TO 2
12460 AL(I) = AO(I) + UA(I) * K1 + K2 * (R(I,0) * UA(0) + R(I,1) * UA(1) + R(I,2
) * UA(2))
12480 AR(I) = AO(I) + UA(I) * K1 - K2 * (R(I,0) * UA(0) + R(I,1) * UA(1) + R(I,2
) * UA(2))
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) = 0:U(2) =  - UB(0)
12560  GOSUB 14000
12580  FOR I = 0 TO 2
12600 BL(I) = B0(I) + UB(I) * K1 + K2 * (R(I,0) * UB(0) + R(I,1) * UB(1) + R(I,2
) * UB(2))
12620 BR(I) = B0(I) + UB(I) * K1 - K2 * (R(I,0) * UB(0) + R(I,1) * UB(1) + R(I,2
) * UB(2))
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
12960 X4(0) = XM - B:X4(1) =  - 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) = YM - B:Y4(0) = A:Y5(1) = YM - B:Y5(0) =  - A:Y6(1) = YM + B:Y7(1)
= YM + C:Y7(0) =  - A:Y8(1) = YM + C:Y8(0) = A:Y9(1) = YM + A
13000 Z4(1) = A:Z4(2) = B - ZM:Z5(1) =  - A:Z5(2) = B - ZM:Z6(1) =  - A:Z6(2) =
 - (ZM + B):Z7(1) =  - A:Z7(2) =  - (ZM + C):Z8(1) = A:Z8(2) =  - (ZM + C):Z9(1)
 = A:Z9(2) =  - (ZM + B)
13040  PRINT D$;"OPEN ANIMATION FILE": PRINT D$;"WRITE ANIMATION FILE"
13060  FOR I = 0 TO NP: PRINT AJ(0,I): PRINT AJ(1,I): PRINT  - AJ(2,I): NEXT I
13080  FOR I = 0 TO NP: PRINT BJ(0,I): PRINT BJ(1,I): PRINT  - BJ(2,I): NEXT I
13100  FOR I = 0 TO NP: PRINT P(0,I): PRINT P(1,I): PRINT  - P(2,I): NEXT I
13120  FOR I = 0 TO NP: PRINT AS%(I): NEXT I
13140  FOR I = 0 TO 2: PRINT AN(I): PRINT TA(I): PRINT BN(I): PRINT BT(I): PRINT
 AL(I): PRINT AR(I): PRINT BL(I): PRINT BR(I): NEXT I
13160  FOR I = 0 TO 2: PRINT XM(I): PRINT YM(I): PRINT ZM(I): PRINT X4(I): PRINT
 X5(I): PRINT X6(I): PRINT X7(I): PRINT X8(I): PRINT X9(I): NEXT I
13180  FOR I = 0 TO 2: PRINT Y4(I): PRINT Y5(I): PRINT Y6(I): PRINT Y7(I): PRINT
 Y8(I): PRINT Y9(I): PRINT Z4(I): PRINT Z5(I): PRINT Z6(I): PRINT Z7(I): PRINT Z
8(I): PRINT Z9(I): NEXT I
13200  PRINT D$;"CLOSE ANIMATION FILE"
```

```
13220  PRINT D$;"RUN ANIMATE"
14000 R(0,0) = U(0) * U(0):R(0,1) = U(0) * U(1) - U(2):R(0,2) = U(0) * U(2) + U(
1):R(1,0) = U(0) * U(1) + U(2):R(1,1) = U(1) * U(1)
14020 R(1,2) = U(1) * U(2) - U(0):R(2,0) = U(0) * U(2) - U(1):R(2,1) = U(1) * U(
2) + U(0):R(2,2) = U(2) * U(2)
14030  RETURN

]
```

```
REM SUBPROGRAM ANIMATE

]LIST

50  LOMEM: 16384
70  HIMEM: 24575
100  REM  PROGRAM ANIMATE
120  DIM AJ%(2,1,71),BJ%(2,1,71),AS%(71),UA(2),AO(2),A1(2),UB(2),B0(2),B1(2),IN(
2,9),P%(2,1,71)
160  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)
180  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)
200  REM  RETRIEVE VALUES FROM DISK FILE
220 D$ = "": REM  CNTRL-D
240  PRINT D$;"OPEN INPUT FILE": PRINT D$;"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 D$;"CLOSE INPUT FILE"
300  FOR I = 0 TO 2:UA(I) = IN(I,0):AO(I) = IN(I,1):A1(I) = IN(I,2):UB(I) = IN(I
,3):B0(I) = IN(I,4):B1(I) = IN(I,5): NEXT I:W2 = IN(0,6):A2 = IN(0,7):AL = IN(0,
8):DO = IN(0,9):ZETA = IN(1,6):ETA = IN(1,7)
315 A1(2) =  - A1(2):B1(2) =  - B1(2):AO(2) =  - AO(2):B0(2) =  - B0(2)
320 NP =  INT (AL / DO)
330 SF% = 250: REM  SCALE FACTOR
340  PRINT D$;"OPEN ANIMATION FILE": PRINT D$;"READ ANIMATION FILE"
360  FOR I = 0 TO NP: INPUT X: INPUT Y: INPUT Z:AJ%(0,0,I) = X * SF%:AJ%(1,0,I)
= Y * SF%:AJ%(2,0,I) = Z * SF%: NEXT I
380  FOR I = 0 TO NP: INPUT X: INPUT Y: INPUT Z:BJ%(0,0,I) = X * SF%:BJ%(1,0,I)
= Y * SF%:BJ%(2,0,I) = Z * SF%: NEXT I
460  FOR I = 0 TO NP: INPUT X: INPUT Y: INPUT Z:P%(0,0,I) = X * SF%:P%(1,0,I) =
Y * SF%:P%(2,0,I) = Z * SF%: NEXT I
480  FOR I = 0 TO NP: INPUT AS%(I): NEXT I
500  FOR I = 0 TO 2: INPUT AN(I): INPUT TA(I): INPUT BN(I): INPUT BT(I): INPUT A
L(I): INPUT AR(I): INPUT BL(I): INPUT BR(I): NEXT I
520  FOR I = 0 TO 2: INPUT XM(I): INPUT YM(I): INPUT ZM(I): INPUT X4(I): INPUT X
5(I): INPUT X6(I): INPUT X7(I): INPUT X8(I): INPUT X9(I): NEXT I
540  FOR I = 0 TO 2: INPUT Y4(I): INPUT Y5(I): INPUT Y6(I): INPUT Y7(I): INPUT Y
8(I): INPUT Y9(I): INPUT Z4(I): INPUT Z5(I): INPUT Z6(I): INPUT Z7(I): INPUT Z8(
I): INPUT Z9(I): NEXT I
560  PRINT D$;"CLOSE ANIMATION FILE"
580  REM  BRING IN SKELETON ARRAY AND LOAD THE GRAPHICS GENERATOR
600  PRINT D$;"BLOAD RSSR SKELETON ARRAY"
620  PRINT D$;"BLOAD A2-3D2"
635 MEM = 38327
640  FOR J = 0 TO NP: FOR I = 0 TO 2:BV% = AJ%(I,0,J): GOSUB 6000:AJ%(I,0,J) = L
O%:AJ%(I,1,J) = HI%: NEXT I: NEXT J
650  FOR J = 0 TO NP: FOR I = 0 TO 2:BV% = BJ%(I,0,J): GOSUB 6000:BJ%(I,0,J) = L
O%:BJ%(I,1,J) = HI%: NEXT I: NEXT J
660  FOR J = 0 TO NP: FOR I = 0 TO 2:BV% = P%(I,0,J): GOSUB 6000:P%(I,0,J) = LO%
:P%(I,1,J) = HI%: NEXT I: NEXT J
690  REM  STUFF THE SKELETON ARRAY WITH FIXED POINTS (AXES, ETC.)
700 MEM = 33279: FOR I = 0 TO 2:BV% =  INT (XM(I) * SF%): GOSUB 6000: NEXT I
740 MEM = 33293: FOR I = 0 TO 2:BV% =  INT (YM(I) * SF%): GOSUB 6000: NEXT I
760 MEM = 33300: FOR I = 0 TO 2:BV% =  INT (Y4(I) * SF%): GOSUB 6000: NEXT I
780 MEM = 33307: FOR I = 0 TO 2:BV% =  INT (Y5(I) * SF%): GOSUB 6000: NEXT I
800 MEM = 33314: FOR I = 0 TO 2:BV% =  INT (Y9(I) * SF%): GOSUB 6000: NEXT I
820 MEM = 33321: FOR I = 0 TO 2:BV% =  INT (Y6(I) * SF%): GOSUB 6000: NEXT I
840 MEM = 33328: FOR I = 0 TO 2:BV% =  INT (Y7(I) * SF%): GOSUB 6000: NEXT I
860 MEM = 33335: FOR I = 0 TO 2:BV% =  INT (Y8(I) * SF%): GOSUB 6000: NEXT I
880 MEM = 33125: FOR I = 0 TO 2:BV% =  INT (ZM(I) * SF%): GOSUB 6000: NEXT I
900 MEM = 33132: FOR I = 0 TO 2:BV% =  INT (Z4(I) * SF%): GOSUB 6000: NEXT I
920 MEM = 33139: FOR I = 0 TO 2:BV% =  INT (Z5(I) * SF%): GOSUB 6000: NEXT I
940 MEM = 33146: FOR I = 0 TO 2:BV% =  INT (Z6(I) * SF%): GOSUB 6000: NEXT I
960 MEM = 33153: FOR I = 0 TO 2:BV% =  INT (Z7(I) * SF%): GOSUB 6000: NEXT I
980 MEM = 33160: FOR I = 0 TO 2:BV% =  INT (Z9(I) * SF%): GOSUB 6000: NEXT I
1000 MEM = 33167: FOR I = 0 TO 2:BV% =  INT (Z8(I) * SF%): GOSUB 6000: NEXT I
```

```
1020 MEM = 33174: FOR I = 0 TO 2:BV% =  INT (TA(I) * SF%): GOSUB 6000: NEXT I
1040 MEM = 33181: FOR I = 0 TO 2:BV% =  INT (AN(I) * SF%): GOSUB 6000: NEXT I
1060 MEM = 33188: FOR I = 0 TO 2:BV% =  INT (AL(I) * SF%): GOSUB 6000: NEXT I
1080 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 (B0(I) * SF%): GOSUB 6000: NEXT I
1140 MEM = 33202: FOR I = 0 TO 2:BV% =  INT (BT(I) * SF%): GOSUB 6000: NEXT I
1160 MEM = 33209: 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 (X9(I) * SF%): GOSUB 6000: NEXT I
1240 MEM = 33237: FOR I = 0 TO 2:BV% =  INT (X7(I) * SF%): GOSUB 6000: NEXT I
1260 MEM = 33244: FOR I = 0 TO 2:BV% =  INT (X8(I) * SF%): GOSUB 6000: NEXT I
1280 MEM = 33251: FOR I = 0 TO 2:BV% =  INT (X6(I) * SF%): GOSUB 6000: NEXT I
1282 MEM = 33258: FOR I = 0 TO 2:BV% =  INT (X5(I) * SF%): GOSUB 6000: NEXT I
1284 MEM = 33265: FOR I = 0 TO 2:BV% =  INT (XM(I) * SF%): GOSUB 6000: NEXT I
1286 MEM = 33272: FOR I = 0 TO 2:BV% =  INT (X4(I) * SF%): GOSUB 6000: NEXT I
1288 MEM = 33069: FOR I = 0 TO 2:LO% = AJ%(I,0,0):HI% = AJ%(I,1,0): GOSUB 7000:
NEXT I
1290  IF ZETA = 0 THEN  IF ETA = 0 THEN MEM = 33076: FOR I = 0 TO 2:LO% = AJ%(I,
0,0):HI% = AJ%(I,1,0): GOSUB 7000: NEXT I: GOTO 1293
1291 MEM = 33076: FOR I = 0 TO 2:LO% = P%(I,0,0):HI% = P%(I,1,0): GOSUB 7000: NE
XT I
1293 MEM = 33083: FOR I = 0 TO 2:LO% = BJ%(I,0,0):HI% = BJ%(I,1,0): GOSUB 7000:
NEXT I
1294  IF ZETA = 0 THEN  IF ETA = 0 THEN MEM = 33090: FOR I = 0 TO 2:LO% = BJ%(I,
0,0):HI% = BJ%(I,1,0): GOSUB 7000: NEXT I: GOTO 1296
1295 MEM = 33090: FOR I = 0 TO 2:LO% = P%(I,0,0):HI% = P%(I,1,0): GOSUB 7000: NE
XT 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 PI = 0:BA = 0:HE = 0
1315 I% = 1
1320 MEM = 33031: FOR I = 0 TO 2:BV% =  INT (EY(I) * SF%): GOSUB 6000: NEXT I
1330  POKE 33038,PI: POKE 33039,BA: POKE 33040,HE
1335  CALL 24576: IF I% = 1 THEN I1$ = "N": GOTO 1380
1340  INPUT "EYE POSITION SATISFACTORY (Y/N)?";I1$
1360  IF  LEFT$ (I1$,1) = "Y" THEN  GOTO 3000
1380  IF  LEFT$ (I1$,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)";PI,BA,HE: 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 = 33356:MF = NP * 70 + 71 + MI
3055  FOR M = MI TO MF: POKE M,17: NEXT M
3060 A = 0:B = NP:C = 1:SP% = 0
3120  FOR J = A TO B STEP C
3140 M = 33356 + J * 70
3220 MEM = 33069: FOR I = 0 TO 2:LO% = AJ%(I,0,J):HI% = AJ%(I,1,J): GOSUB 7000:
NEXT I
3230  IF ZETA = 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: NE
XT 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 ZETA = 0 THEN  IF ETA = 0 THEN MEM = 33090: FOR I = 0 TO 2:LO% = BJ%(I,
0,J):HI% = BJ%(I,1,J): GOSUB 7000: NEXT I: GOTO 3300
3280 MEM = 33090: FOR I = 0 TO 2:LO% = P%(I,0,J):HI% = P%(I,1,J): GOSUB 7000: NE
XT I
3300 MEM = 33059:I = 0:BV% = M - 65536: GOSUB 6000
3320  CALL 24576
3330  FOR I = M + 34 TO M STEP  - 1: IF  PEEK (I) = 121 THEN  POKE I,17: GOTO 33
```

```
38
3335  NEXT I
3338  POKE M + 33,27: POKE M + 34,SP%
3340 MEM = 33059:I = 0:BV% = (M + 35) - 65536: GOSUB 6000
3360  CALL 24576
3370  FOR I = M + 69 TO M + 35 STEP  - 1: IF  PEEK (I) = 121 THEN  POKE I,17: GO
TO 3380
3375  NEXT I
3380  NEXT J
3385  HGR : REM  CLEAR GRAPHICS SCREEN
3390  POKE 33059,26: POKE 33060,43: POKE 33061,0
3395  CALL 24576
3400  REM  ANIMATION CALL
3410  PRINT : PRINT : PRINT "PRESS ANY KEY TO STOP ANIMATION"
3420  POKE 33059,17: POKE 33060,17: POKE 33061,17
3440  POKE 33055,11:MEM = 33055:I = 0:BV% =  - 32180: GOSUB 6000
3460  POKE 33053,12: POKE 33054,01
3480  CALL 24576
3485  IF  PEEK ( - 16384) > 127 THEN : GOTO 3500
3490  GOTO 3480
3500  INPUT "SPEED O.K. ? (Y/N)..";I1$: IF I1$ = "N" THEN  INPUT "SELECT SPEED.(
5 SLOWEST,0 FASTEST)...";SP%: FOR I = 33390 TO MF STEP 70: POKE I,SP%: NEXT I: G
OTO 3480
3520  INPUT "VIEWPOINT O.K. ? (Y/N)..";I1$: IF I1$ = "N" THEN F% = 1: POKE 33054
,0: FOR I = 33055 TO 33057: POKE I,17: NEXT I: POKE 33104,17: POKE 33027,8: POKE
 33028,0: GOTO 1290
3540  INPUT "CONTINUE WITH ANIMATION ? (Y/N)";I1$: IF I1$ = "Y" THEN  GOTO 3480
3550  INPUT "TRY A DIFFERENT MECHANISM ? (Y/N)";I1$: IF I1$ = "Y" THEN  PRINT D$
;"RUN INPUT2
3560  END
6000 HI% = BV% / 256
6005  IF BV% < 0 THEN HI% = HI% + 1
6010 LO% = BV% - HI% * 256
6020  IF BV% >  - 1 THEN  GOSUB 7000: RETURN
6030 HI% = 255 + HI%
6040 LO% = 256 + LO%
6050  IF LO% < 256 THEN  GOSUB 7000: RETURN
6060 HI% = HI% + 1
6070 LO% = 0
6080  GOSUB 7000: RETURN
7000  POKE (MEM + 2 * I) + 1,LO%
7020  POKE (MEM + 2 * I) + 2,HI%
7040  RETURN

]
```

APPENDIX D

PLOTTING AND BINARY DISK STORAGE PROGRAMS

```
 REM PLOT ROUTINE UNCOMPRESSED...WILL NOT RUN...

]LIST

100  REM  SSSSSSSSSSSSSSSSSSSSSS
110  REM  S INITIALIZE           S
120  REM  SSSSSSSSSSSSSSSSSSSSSSS
130  REM  * SET LOMEM ABOVE      *
135  REM  * GRAPHICS             *
140  LOMEM: 16384
150  REM  * SAVE HIMEM           *
160 HL% =  PEEK (115):HH% =  PEEK (116)
170  REM  * LOAD HRCG, ASCII.SET *
180  REM  * AND LANDSCAPE.SET    *
190  ONERR  GOTO 440
200  TEXT : HOME : HGR :ADRS = 0
210  PRINT  CHR$ (4);"BLOAD RBOOT": CALL 520
220 ADRS =  USR (0),"HRCG"
230  POKE 216,0: REM   RESET ONERR
240  IF ADRS < 0 THEN ADRS = ADRS + 65536
250 CS = ADRS - 2 * 768: HIMEM: CS
260 CH =  INT (CS / 256):CL = CS - CH * 256
270  POKE ADRS + 7,CL: POKE ADRS + 8,CH
280 D$ =  CHR$ (4): REM   CTRL-D
290  PRINT D$;"BLOAD ASCII.SET,A";CS
300  PRINT D$;"BLOAD LANDSCAPE.SET,A"CS + 768
310  CALL ADRS + 3
320  REM  * CTRL CHARACTERS      *
330 CP$ =  CHR$ (16): REM   CLR PAGE
340 CL$ =  CHR$ (12): REM  LOWER CASE
350 CK$ =  CHR$ (11): REM  UPPER CASE
355 CI$ =  CHR$ (9): REM  INVERSE VIDEO
360 CO$ =  CHR$ (15): REM  OPTIONS
365 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:VD = 160
410  PRINT CI$
420  PRINT "AFTER PLOT IS FINISHED PRESS S TO STORE IMAGE OF PLOT OR ANY KEY TO
CONTINUE"
425  PRINT CN$
430  GOTO 8000
440  TEXT
450  PRINT "ERROR IN RLOAD OF RBOOT"
460  POKE 216,0
470  END
1000  REM  SSSSSSSSSSSSSSSSSSSSS
1010  REM  S READ PLOT           S
1015  REM  S DESCRIPTOR FILE     S
1020  REM  SSSSSSSSSSSSSSSSSSSSS
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 NX%
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%
1130  INPUT NA$(I)
1140  NEXT I
1150  PRINT D$;"CLOSE PLOT DESCRIPTOR"
1160  RETURN
2000  REM  SSSSSSSSSSSSSSSSSSSSS
```

```
2010  REM  S READ PLOT DATA FILE S
2020  REM  SSSSSSSSSSSSSSSSSSSSS
2030  REM
2040  REM INIT MACHINE CODE ROUTINE TO LOAD IN ARRAY.
2050  PRINT D$;"BRUN BLOAD ARRAY"
2060  REM
2070  REM DIMENSION THE ARRAY
2080  DIM PD(NV%,NX%)
2090  REM  AND READ IT IN.
2100  &  LOAD PD"PLOT DATA"
2140  RETURN
3000  REM  SSSSSSSSSSSSSSSSSSSSS
3010  REM  S GET X & Y VARIABLES S
3020  REM  SSSSSSSSSSSSSSSSSSSSS
3030  PRINT CP$: REM   CLR PG
3040  PRINT CS$;"SELECT THE Y-AXIS VARIABLE :": PRINT CK$: PRINT
3050  FOR I = Ø TO NV%
3060  HTAB 9: PRINT I;".)";NA$(I): PRINT
3070  NEXT I
3080  GET Y%: PRINT
3085  IF Y% < Ø OR Y% > NV% THEN  GOTO 3000
3090  HTAB 1: VTAB 1
3100  PRINT CS$;"SELECT THE X-AXIS VARIABLE :": PRINT CK$: PRINT : HTAB 1: VTAB
I * 2 + 1
3110  GET X%: PRINT
3120  IF X% < Ø OR X% > NV% THEN  GOTO 3090
3130  RETURN
4000  REM  SSSSSSSSSSSSSSSSSSSSS
4010  REM  S FIND MINS, MAXS     S
4015  REM  S & SCALE FACTOR      S
4020  REM  SSSSSSSSSSSSSSSSSSSSS
4030 YN = PD(Y%,Ø):YM = PD(Y%,Ø):XN = PD(X%,Ø):XM = PD(X%,Ø)
4040  FOR I = Ø TO OP
4050  IF PD(Y%,I) < YN THEN YN = PD(Y%,I)
4060  IF PD(Y%,I) > YM THEN YM = PD(Y%,I)
4070  IF PD(X%,I) < XN THEN XN = PD(X%,I)
4080  IF PD(X%,I) > XM THEN XM = PD(X%,I)
4090  NEXT I
4100  REM  * CALCULATE SCALE FAC.*
4110 YF =  ABS (YM - YN) / VD
4120 XF =  ABS (XM - XN) / HD
4130  REM  * CALCULATE DELTAS    *
4140 YD =  ABS (YM - YN) / 4
4150 XD =  ABS (XM - XN) / 4
4160  REM  * FIND VERTICAL LOC.  *
4165  REM  * OF X-AXIS           *
4170 LX% = VD: IF YN < Ø AND YF <  > Ø THEN LX% =  INT (VD + YN / YF)
4180 LX% = LX% + 4
4190  RETURN
5000  REM  SSSSSSSSSSSSSSSSSSSSS
5010  REM  S DRAW & LABEL AXES   S
5020  REM  SSSSSSSSSSSSSSSSSSSSS
5030  PRINT CP$: REM   CLR PG
5040  HCOLOR= 3
5050  REM  * DRAW AXES           *
5060  HPLOT 279 - HD,4 TO 279 - HD,VD + 4
5070  HPLOT 279 - HD,LX% TO 279,LX%
5080  REM  * DRAW TICK MARKS     *
5090 YTICK = VD / 4:XTICK = HD / 4
5100  FOR I = Ø TO 4
5110  HPLOT 278 - HD,I * YTICK + 4 TO 280 - HD,I * YTICK + 4
5120  HPLOT (279 - HD) + XTICK * I,LX% - 1 TO (279 - HD) + XTICK * I,LX% + 1
5130  NEXT I
5140  REM  * LABEL AXES          *
5145  PRINT CA$;2: REM   LANDSCAPE.SET
5150 TA% =  INT (VD / 16) -  INT ( LEN (NA$(Y%)) / 2)
```

```
5160  IF TA% < 1 THEN TA% = 1
5170  VTAB TA%: HTAB 1
5180  FOR I = Ø 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  VTAB 23: HTAB  INT ((279 - HD) / 7 + .5) + TA%
5240  PRINT NA$(X%)
5250  REM  * ANNOTATE THE AXIS   *
5260  REM  * WITH AXIS INCREMENTS*
5270  FOR J = Ø TO 4
5280 YTICK = YN + J * YD
5290 XTICK = XN + J * XD
5300  REM  * FORMAT YTICK & XTICK*
5310 NUM = YTICK: GOSUB 7000:YTICK$ = NUM$
5320 NUM = XTICK: GOSUB 7000:XTICK$ = NUM$
5330  VTAB  INT (VD / 8 - J * 5 + 1)
5340  HTAB 3
5350  PRINT YTICK$
5360  VTAB  INT (VD / 8 + .5) + 2
5370  HTAB  INT ((279 - HD) / 7 + J * 7 -  LEN (XTICK$) / 2 + .5)
5380  PRINT XTICK$
5390  NEXT J
5400  RETURN
6000  REM  SSSSSSSSSSSSSSSSSSSSS
6010  REM  S PLOT THE POINTS     S
6020  REM  SSSSSSSSSSSSSSSSSSSSS
6030 YP% = VD + 4 -  INT ((PD(Y%,Ø) - YN) / YF)
6040 XP% =  INT ((PD(X%,Ø) - XN) / XF) + 279 - HD
6050  HPLOT 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 - HD
6090  HPLOT  TO XP%,YP%
6100  NEXT I
6110  RETURN
7000  REM  SSSSSSSSSSSSSSSSSSSSS
7010  REM  S FORMAT NUMBERS      S
7020  REM  SSSSSSSSSSSSSSSSSSSSS
7030  REM
7040  REM SET DECIMAL PLACES.
7050 D = 2
7060  REM
7070  REM CONVERT NUM TO STRING.
7080 NUM$ =  STR$ (NUM)
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$,K,1) <  > "." THEN  NEXT K
7180  REM K IS NOW AT DECIMAL OR END.
7190  REM
7200  REM DO D DIGITS EXIST TO THE RIGHT OF THE DECIMAL?
7210  IF K + D <  = I - 1 THEN LP% = K + D: GOTO 7250: REM YES
7220 LP% = I - 1: REM NO, SO PRINT ALL.
7230  REM
7240  REM ROUND OFF MANTISSA.
7250 MAN =  VAL ( LEFT$ (NUM$,LP% + 1)):P = 100:MAN =  INT (MAN * P + .5) / P:MA
N$ =  STR$ (MAN)
7260  REM
```

```
7270  REM CONCAT EXPONENT PART AND RETURN.
7280 NUM$ = MAN$ +  MID$ (NUM$,I)
7290  RETURN
8000  REM  MMMMMMMMMMMMMMMMMMMMM
8010  REM  M                   M
8020  REM  M   GENERAL PURPOSE  M
8030  REM  M   PLOTTING ROUTINE M
8040  REM  M                   M
8050  REM  M UPDATED: 05/24/82  M
8060  REM  M                   M
8070  REM  MMMMMMMMMMMMMMMMMMMMM
8080  REM
8130  REM  AAAAAAAAAAAAAAAAAAAAA
8140  REM  A READ PLOT          A
8150  REM  A DESCRIPTOR FILE    A
8160  REM  AAAAAAAAAAAAAAAAAAAAA
8170  GOSUB 1000
8180  REM  AAAAAAAAAAAAAAAAAAAAA
8190  REM  A READ PLOT DATA FILE A
8200  REM  AAAAAAAAAAAAAAAAAAAAA
8210  GOSUB 2000
8220  REM  AAAAAAAAAAAAAAAAAAAAA
8230  REM  A GET X & Y VARIABLES A
8240  REM  AAAAAAAAAAAAAAAAAAAAA
8250  GOSUB 3000
8260  REM  AAAAAAAAAAAAAAAAAAAAA
8270  REM  A FIND MINS, MAXS AND A
8280  REM  A SCALE FACTOR       A
8290  REM  AAAAAAAAAAAAAAAAAAAAA
8300  GOSUB 4000
8310  IF YF = 0 THEN  PRINT "THE DEPENDENT VARIABLE IS CONSTANT ";PD(Y%,0);" .":
 GOTO 8470
8320  REM  AAAAAAAAAAAAAAAAAAAAA
8330  REM  A DRAW & LABEL AXIS  A
8340  REM  AAAAAAAAAAAAAAAAAAAAA
8350  GOSUB 5000
8360  REM  AAAAAAAAAAAAAAAAAAAAA
8370  REM  A PLOT THE POINTS    A
8380  REM  AAAAAAAAAAAAAAAAAAAAA
8390  GOSUB 6000
8400  REM  * WAIT FOR KEYPRESS  *
8410  GET A$
8420  REM  * IF CHR=S THEN STORE *
8430  REM  * PLOT ON DISK       *
8440  IF A$ <  > "S" THEN  GOTO 8470
8442  REM DISCONNECT HRCG SO AS TO NOT MESS UP PLOT
8444  POKE 54,240: POKE 55,253: CALL 1002: PRINT
8450  PRINT D$;"BSAVE ";NA$(Y%);".PIC,A$2000,L$2000
8452  REM RECONNECT HRCG
8454  POKE 54,24: POKE 55,143: CALL 1002
8460  REM  * CLEAR PAGE         *
8470  PRINT CP$: PRINT
8480  VTAB 12
8490  PRINT CI$: REM  INVERSE VIDEO
8500  PRINT "WOULD YOU LIKE ANOTHER PLOT ?"
8505  GET A$: PRINT
8507  PRINT CN$: REM   NORMAL VIDEO
8510  IF  LEFT$ (A$,1) = "N" THEN  GOTO 8540
8530  GOTO 8220
8540  REM
8550  REM CLEAN UP & EXIT.
8560  REM
8570  REM RESET CSW & KSW FOR NORMAL I/O.
8580  POKE 54,240: 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"

]
```

```
                        1    *************************
                        2    *                       *
                        3    *      BLOAD ARRAY      *
                        4    *                       *
                        5    *      W. M. KOOS JR.   *
                        6    *                       *
                        7    */ LAST UPDATE: 6/16/82 *
                        8    *                       *
                        9    *************************
                       10
                       11    * THIS PROGRAM DOES A BINARY
                       12    *LOAD OF ARRAYS PREVIOUSLY SAVED
                       13    *WITH THE COMPANION PROGRAM
                       14    *"BSAVE ARRAY" AND IS DESIGNED
                       15    *TO BE USED FROM WITHIN AN ASOFT
                       16    *PROGRAM VIA THE AMPERSAND CALL.
                       17
                       18    * THE PROPER SYNTAX IS
                       19    *   &LOAD ARRAYNAME "FILENAME"
                       20    *WHERE ARRAYNAME IS A VALID
                       21    *PREVIOUSLY DIMENSIONED ARRAY
                       22    *(ONLY FIRST TWO CHARS. USED AS
                       23    *IN ASOFT) AND FILENAME IS THE
                       24    *DISK FILE YOU WISH THE ARRAY TO
                       25    *BE RETRIEVED FROM.
                       26    * NOTE THAT ARRAYNAME IS ONLY
                       27    *THE NAME OF THE ARRAY WITH NO
                       28    *PARENTHESIS OR DIMENSIONS.
                       29
                       30    * PREPARATIONS FOR USE:
                       31    *BRUN BLOAD ARRAY FROM THE
                       32    *CALLING PROGRAM.
                       33
                       34             ORG  $2E5
                       35
                       36    *SET UP THE & VECTOR AND RETURN.
                       37
02E5: A9 F9            38             LDA  #BLOADPGM  ;START OF THIS PGM.
02E7: 8D F6 03         39             STA  $3F6
02EA: A9 02            40             LDA  #>BLOADPGM
02EC: 8D F7 03         41             STA  $3F7
02EF: 60               42             RTS
                       43
                       44    *BEGINNING OF DOS BSAVE MSG.
                       45
02F0: 8D 84            46    BLOADMSG HEX  8D84       ;CR,CTRL-D
02F2: C2 CC CF
02F5: C1 C4 A0         47             ASC  "BLOAD "
02F8: 00               48             HEX  00          ;EOL
                       49
                       50    * EXTERNAL SUBROUTINES:
                       51
                       52    CHRGET   =    $B1        ;ASOFT CHRGET S/R CALL -
                       53    CHRGOT   =    $B7        ;GETS NEXT SEQUENTIAL CHR
                       54    *   OR TOKEN - LOADS A-REG FROM LOCN SPECIFIED
                       55    *   BY TXTPTR ($B8-$B9) - CARRY IS CLRD IF
                       56    *   CHR IS NUMERIC OTHERWISE SET -
                       57    *   Z-FLAG SET IF CHR IS 0 (EOL) OR : (EOS),
                       58    *   OTERWISE Z-FLAG CLRD -
                       59    *   CHRGET INCREMENTS TXTPTR BEFORE GETTING
                       60    *   CHR; CHRGOT LEAVES TXTPTR UNCHANGED.
                       61    ERROR    =    $D412      ;ASOFT ERROR PROCESSING -
                       62    *   CHECKS ERRFLG AND JMPS TO
```

```
                        63    *   HANDLERR IF ONERR ACTIVE
                        64    *   OTHERWISE PRINTS ERROR MSG
                        65    *   BASED ON CODE IN X-REG.
                        66    *   SEE ASOFT REF MANUAL FOR CODES.
                        67    LINPRT   =    $ED24       ;ASOFT PRINT 2-BYTE
                        68    *   UNSIGNED INTGER IN X-REG (LSB) & A-REG (MSB).
                        69    STROUT   =    $DB3A       ;ASOFT PRINT STRING
                        70    *   POINTED TO BY Y-REG (MSB) &
                        71    *   A-REG (LSB); STRING MUST END
                        72    *   WITH A ZERO OR A QUOTE.
                        73    COUT     =    $FDED       ;MON CHR OUTPUT ROUTINE
                        74    CROUT    =    $FD8E       ;MON ROUTINE TO PRINT CR
                        75    SYNCHR   =    $DEC0       ;ASOFT SYNTAX CHR CHECK -
                        76    *   CHECKS TO VERIFY TXTPTR POINTS
                        77    *   TO SAME CHR AS THAT IN A-REG.
                        78    *   NORMAL EXIT IS THROUGH CHRGET
                        79    *   THEREBY INCREMENTING TXTPTR;
                        80    *   ELSE SYNTAX ERROR GENERATED;
                        81    *   Y-REG IS CLEARED EITHER WAY.
                        82    SAVE     =    $FF4A       ;MON SAVE ALL REGS.
                        83    RESTORE  =    $FF3F       ;MON RESTORE ALL REGS.
                        84
                        85    *EXTERNAL PARAMETER STORAGE
                        86
                        87    NAME     =    $06         ;PGM STORAGE OF
                        88    *   ARRAY NAME GOTTEN FROM ASOFT TEXT.
                        89    ARYTAB   =    $6B         ;ASOFT PTR TO
                        90    *   BEGINNING OF ARRAY SPACE.
                        91    STREND   =    $6D         ;ASOFT PTR TO
                        92    *   END OF NUMERIC STORAGE.
                        93    ARYPTR   =    $08         ;PGM POINTER USED
                        94    *  TO INCREMENT THROUGH MEMORY.
                        95    TEMP     =    $EB         ;TEMPORARY STORAGE.
                        96    ARYLEN   =    $EC         ;PGM VARIABLE FOR
                        97    *LENGTH OF ARRAY.
                        98    ARYDIM   =    $ED         ;PGM VARIABLE FOR
                        99    *NO. OF DIMENSIONS OF ARRAY.
                        100   REMSTK   =    $F8         ;ASOFT STACK PTR SAVED
                        101   *  BEFORE EACH STATEMENT.
                        102   ERRNUM   =    $DE         ;ASOFT ERROR CODE STORE
                        103
                        104   *PROGRAM BEGINNING
                        105
02F9: 20 4A FF          106   BLOADPGM JSR  SAVE        ;SAVE ALL REGS
02FC: A9 00             107            LDA  #0          ;INITIALIZE NAME
02FE: 85 06             108            STA  NAME
0300: 85 07             109            STA  NAME+1
0302: A9 B6             110            LDA  #182        ;CHECK FOR TOKENIZED LOAD
0304: 20 C0 DE          111            JSR  SYNCHR
0307: 20 B7 00          112            JSR  CHRGOT      ;GET THE ARRAY NAME
030A: F0 4F             113            BEQ  SYNTAX      ;AT TXTPTR
030C: 85 06             114            STA  NAME
030E: 20 B1 00          115            JSR  CHRGET
0311: F0 48             116            BEQ  SYNTAX
0313: C9 22             117            CMP  #$22        ;CMP TO QUOTE
0315: F0 0B             118            BEQ  FINDARRY    ;JMP IF ONE CHR ARRAYNAME
0317: 85 07             119            STA  NAME+1      ;ELSE STORE 2ND CHR
0319: 20 B1 00          120   FINDQUOT JSR  CHRGET      ;AND MOVE TXTPTR TO QUOTE
031C: F0 3D             121            BEQ  SYNTAX
031E: C9 22             122            CMP  #$22        ;CMP TO QUOTE
0320: D0 F7             123            BNE  FINDQUOT
0322: A0 00             124   FINDARRY LDY  #0
0324: A5 6B             125            LDA  ARYTAB      ;ARYPTR <-- ARYTAB
0326: 85 08             126            STA  ARYPTR
0328: A5 6C             127            LDA  ARYTAB+1
032A: 85 09             128            STA  ARYPTR+1
```

```
032C: B1 06     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: 85 EB     143          STA  TEMP
0341: C8        144          INY
0342: B1 08     145          LDA  (ARYPTR),Y
0344: 65 09     146          ADC  ARYPTR+1
0346: 85 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
035B: A6 F8     164 SYNTAX   LDX  REMSTK      ;RESTORE STACK FROM BEFORE THE &
035D: 9A        165          TXS
035E: A2 10     166          LDX  #16         ;SYNTAX ERROR CODE
0360: 4C 12 D4  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: E0 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  #$EA        ;MODIFY CHRGET TO
0378: 85 C0     182          STA  $C0         ;ALLOW SPACES
037A: 85 C1     183          STA  $C1
037C: 20 B1 00  184 SENDFILE JSR  CHRGET      ;GET FILENAME
037F: F0 11     185          BEQ  ENDSEND     ;AT TXTPTR
0381: C9 22     186          CMP  #$22        ;CMP TO QUOTE
0383: F0 08     187          BEQ  FINTXT
0385: 09 80     188          ORA  #$80        ;NEG ASCII
0387: 20 ED FD  189          JSR  COUT        ;AND SEND
038A: 18        190          CLC              ;TO DOS
038B: 90 EF     191          BCC  SENDFILE    ;UNCOND JMP
038D: 20 B1 00  192 FINTXT   JSR  CHRGET      ;FINISH OUT
0390: D0 FB     193          BNE  FINTXT      ;TXT TO EOL
0392: A9 F0     194 ENDSEND  LDA  #$F0        ;RESTORE CHRGET
```

```
0394: 85 C0    195           STA  $C0
0396: A9 EF    196           LDA  #$EF
0398: 85 C1    197           STA  $C1
039A: A9 AC    198 SENDADDR  LDA  #","       ;SEND COMMA
039C: 20 ED FD 199           JSR  COUT
039F: A9 C1    200           LDA  #"A"       ;SEND A
03A1: 20 ED FD 201           JSR  COUT
03A4: A5 08    202           LDA  ARYPTR     ;SEND ADDR VIA LINPRT
03A6: AA       203           TAX
03A7: A5 09    204           LDA  ARYPTR+1
03A9: 20 24 ED 205           JSR  LINPRT
03AC: 20 8E FD 206           JSR  CROUT      ;SEND CR.
03AF: A2 00    207           LDX  #0         ;CMP ARRAY LENGTH
03B1: A0 02    208           LDY  #2         ;AND DIMENSIONS TO
03B3: B1 08    209 CMPARYLP  LDA  (ARYPTR),Y ;THOSE SAVED BEFORE
03B5: D5 EC    210           CMP  ARYLEN,X   ;THE FILE LOAD.
03B7: D0 A2    211           BNE  SYNTAX
03B9: C8       212           INY
03BA: E8       213           INX
03BB: E0 03    214           CPX  #3
03BD: 90 F4    215           BLT  CMPARYLP
03BF: A0 00    216           LDY  #0         ;NOW RESTORE THE
03C1: A5 06    217           LDA  NAME       ;ARRAY NAME FROM
03C3: 91 08    218           STA  (ARYPTR),Y ;BEFORE THE
03C5: C8       219           INY             ;FILE LOAD
03C6: A5 07    220           LDA  NAME+1
03C8: 91 08    221           STA  (ARYPTR),Y
03CA: 4C 3F FF 222           JMP  RESTORE    ;RETURN VIA RESTORE
               223 *ROUTINE TO RESTORE REGS.
               224
               225 *NOTE: CODE MUST END BEFORE $3D0
               226 *(DOS VECTORS).

--END ASSEMBLY--

ERRORS: 0

232 BYTES

SYMBOL TABLE - ALPHABETICAL ORDER:

?  ARYDIM  =$ED       ARYLEN  =$EC       ARYPTR  =$08       ARYTAB  =$6B
   BLOADMSG=$02F0     BLOADPGM=$02F9     CHRGET  =$B1       CHRGOT  =$B7
   CMPARYLP=$03B3     CMPNAME =$032C     COUT    =$FDED     CROUT   =$FD8E
   ENDSEND =$0392  ?  ERRNUM  =$DE       ERROR   =$D412     FINDARRY=$0322
   FINDQUOT=$0319     FINTXT  =$038D     FOUND   =$0363     FOUNDLP =$0365
   LINPRT  =$ED24     NAME    =$06       NOMATCH =$0358     OFFSET  =$0339
   REMSTK  =$F8       RESTORE =$FF3F     SAVE    =$FF4A  ?  SENDADDR=$039A
   SENDFILE=$037C     STREND  =$6D       STROUT  =$DB3A     SYNCHR  =$DEC0
   SYNTAX  =$035B     TEMP    =$EB

SYMBOL TABLE - NUMERICAL ORDER:

   NAME    =$06       ARYPTR  =$08       ARYTAB  =$6B       STREND  =$6D
   CHRGET  =$B1       CHRGOT  =$B7    ?  ERRNUM  =$DE       TEMP    =$EB
   ARYLEN  =$EC    ?  ARYDIM  =$ED       REMSTK  =$F8       BLOADMSG=$02F0
   BLOADPGM=$02F9     FINDQUOT=$0319     FINDARRY=$0322     CMPNAME =$032C
   OFFSET  =$0339     NOMATCH =$0358     SYNTAX  =$035B     FOUND   =$0363
   FOUNDLP =$0365     SENDFILE=$037C     FINTXT  =$038D     ENDSEND =$0392
?  SENDADDR=$039A     CMPARYLP=$03B3     ERROR   =$D412     STROUT  =$DB3A
   SYNCHR  =$DEC0     LINPRT  =$ED24     CROUT   =$FD8E     COUT    =$FDED
   RESTORE =$FF3F     SAVE    =$FF4A
```

```
:ASM
                    1   ************************
                    2   *                      *
                    3   *      BSAVE ARRAY     *
                    4   *                      *
                    5   *      W. M. KOOS JR.  *
                    6   *                      *
                    7   */ LAST UPDATE: 6/16/82 *
                    8   *                      *
                    9   ************************
                   10
                   11   * THIS PROGRAM DOES A BINARY
                   12   *SAVE OF ARRAYS AND IS DESIGNED
                   13   *TO BE USED FROM WITHIN AN ASOFT
                   14   *PROGRAM VIA THE AMPERSAND CALL.
                   15
                   16   * THE PROPER SYNTAX IS
                   17   *   &SAVE ARRAYNAME "FILENAME"
                   18   *WHERE ARRAYNAME IS A VALID
                   19   *PREVIOUSLY DIMENSIONED ARRAY
                   20   *(ONLY FIRST TWO CHARS. USED AS
                   21   *IN ASOFT) AND FILENAME IS THE
                   22   *DISK FILE YOU WISH THE ARRAY TO
                   23   *BE STORED IN.
                   24   * NOTE THAT ARRAYNAME IS ONLY
                   25   *THE NAME OF THE ARRAY WITH NO
                   26   *PARENTHESIS OR DIMENSIONS.
                   27
                   28   * PREPARATIONS FOR USE:
                   29   *BRUN BSAVE ARRAY FROM THE
                   30   *CALLING PROGRAM.
                   31
                   32            ORG  $2FA
                   33
                   34   *SET UP THE & VECTOR AND RETURN.
                   35
02FA: A9 0E        36            LDA  #BSAVEPGM  ;START OF THIS PGM.
02FC: 8D F6 03     37            STA  $3F6
02FF: A9 03        38            LDA  #>BSAVEPGM
0301: 8D F7 03     39            STA  $3F7
0304: 60           40            RTS
                   41
                   42   *BEGINNING OF DOS BSAVE MSG.
                   43
0305: 8D 84        44   BSAVEMSG HEX  8D84       ;CR,CTRL-D
0307: C2 D3 C1
030A: D6 C5 A0     45            ASC  "BSAVE "
030D: 00           46            HEX  00         ;EOL
                   47
                   48   * EXTERNAL SUBROUTINES:
                   49
                   50   CHRGET   =    $B1        ;ASOFT CHRGET S/R CALL -
                   51   CHRGOT   =    $B7        ;GETS NEXT SEQUENTIAL CHR
                   52   *   OR TOKEN - LOADS A-REG FROM LOCN SPECIFIED
                   53   *   BY TXTPTR ($B8-$B9) - CARRY IS CLRD IF
                   54   *   CHR IS NUMERIC OTHERWISE SET -
                   55   *   Z-FLAG SET IF CHR IS 0 (EOL) OR : (EOS),
                   56   *   OTERWISE Z-FLAG CLRD -
                   57   *   CHRGET INCREMENTS TXTPTR BEFORE GETTING
                   58   *   CHR; CHRGOT LEAVES TXTPTR UNCHANGED.
                   59   ERROR    =    $D412      ;ASOFT ERROR PROCESSING -
                   60   *   CHECKS ERRFLG AND JMPS TO
                   61   *   HANDLERR IF ONERR ACTIVE
                   62   *   OTHERWISE PRINTS ERROR MSG
```

```
                        63   *   BASED ON CODE IN X-REG.
                        64   *   SEE ASOFT REF MANUAL FOR CODES.
                        65   LINPRT   =    $ED24       ;ASOFT PRINT 2-BYTE
                        66   *   UNSIGNED INTGER IN X-REG (LSB) & A-REG (MSB).
                        67   STROUT   =    $DB3A       ;ASOFT PRINT STRING
                        68   *   POINTED TO BY Y-REG (MSB) &
                        69   *   A-REG (LSB); STRING MUST END
                        70   *   WITH A ZERO OR A QUOTE.
                        71   COUT     =    $FDED       ;MON CHR OUTPUT ROUTINE
                        72   CROUT    =    $FD8E       ;MON ROUTINE TO PRINT CR
                        73   SYNCHR   =    $DEC0       ;ASOFT SYNTAX CHR CHECK -
                        74   *   CHECKS TO VERIFY TXTPTR POINTS
                        75   *   TO SAME CHR AS THAT IN A-REG.
                        76   *   NORMAL EXIT IS THROUGH CHRGET
                        77   *   THEREBY INCREMENTING TXTPTR;
                        78   *   ELSE SYNTAX ERROR GENERATED;
                        79   *   Y-REG IS CLEARED EITHER WAY.
                        80   SAVE     =    $FF4A       ;MON SAVE ALL REGS.
                        81   RESTORE  =    $FF3F       ;MON RESTORE ALL REGS.
                        82
                        83   *EXTERNAL PARAMETER STORAGE
                        84
                        85   NAME     =    $06         ;PGM STORAGE OF
                        86   *   ARRAY NAME GOTTEN FROM ASOFT TEXT.
                        87   ARYTAB   =    $6B         ;ASOFT PTR TO
                        88   *   BEGINNING OF ARRAY SPACE.
                        89   STREND   =    $6D         ;ASOFT PTR TO
                        90   *   END OF NUMERIC STORAGE.
                        91   ARYPTR   =    $08         ;PGM POINTER USED
                        92   *  TO INCREMENT THROUGH MEMORY.
                        93   TEMP     =    $EB         ;TEMPORARY STORAGE.
                        94   REMSTK   =    $F8         ;ASOFT STACK PTR SAVED
                        95   *  BEFORE EACH STATEMENT.
                        96   ERRNUM   =    $DE         ;ASOFT ERROR CODE STORE
                        97
                        98   *PROGRAM BEGINNING
                        99
030E: 20 4A FF 100   BSAVEPGM JSR  SAVE        ;SAVE ALL REGS
0311: A9 00    101           LDA  #0           ;INITIALIZE NAME
0313: 85 06    102           STA  NAME
0315: 85 07    103           STA  NAME+1
0317: A9 B7    104           LDA  #183         ;CHECK FOR TOKENIZED SAVE
0319: 20 C0 DE 105           JSR  SYNCHR
031C: 20 B7 00 106           JSR  CHRGOT       ;GET THE ARRAY NAME
031F: F0 4F    107           BEQ  SYNTAX       ;AT TXTPTR
0321: 85 06    108           STA  NAME
0323: 20 B1 00 109           JSR  CHRGET
0326: F0 48    110           BEQ  SYNTAX
0328: C9 22    111           CMP  #$22         ;CMP TO QUOTE
032A: F0 0B    112           BEQ  FINDARRY     ;JMP IF ONE CHR ARRAYNAME
032C: 85 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  #$22         ;CMP TO QUOTE
0335: D0 F7    117           BNE  FINDQUOT
0337: A0 00    118   FINDARRY LDY  #0
0339: A5 6B    119           LDA  ARYTAB       ;ARYPTR <-- ARYTAB
033B: 85 08    120           STA  ARYPTR
033D: A5 6C    121           LDA  ARYTAB+1
033F: 85 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
```

```
034C: F0 2A     129           BEQ  FOUND
                130
                131 *FALLS THRU TO HERE IF NAME NAME COMPARE FAILS
                132
034E: C8        133 OFFSET    INY              ;GET OFFSET TO
034F: B1 08     134           LDA  (ARYPTR),Y  ;NEXT ARRAY
0351: 18        135           CLC              ;NAME AND ADD
0352: 65 08     136           ADC  ARYPTR      ;TO ARYPTR
0354: 85 EB     137           STA  TEMP
0356: C8        138           INY
0357: B1 08     139           LDA  (ARYPTR),Y
0359: 65 09     140           ADC  ARYPTR+1
035B: 85 09     141           STA  ARYPTR+1
035D: A5 EB     142           LDA  TEMP
035F: 85 08     143           STA  ARYPTR
0361: C5 6D     144           CMP  STREND      ;CHECK FOR
0363: A5 09     145           LDA  ARYPTR+1    ;END OF
0365: E5 6E     146           SBC  STREND+1    ;ARRAY STORAGE
0367: B0 07     147           BGE  SYNTAX
0369: A0 00     148           LDY  #0
036B: F0 D4     149           BEQ  CMPNAME     ;UNCOND JMP
                150
                151 *HERE IF 1ST CHR DOESNT MATCH
                152
036D: C8        153 NOMATCH   INY              ;BUMP Y PAST 2ND CHR OF NAME
036E: D0 DE     154           BNE  OFFSET      ;UNCOND JMP
                155
                156 *HERE IF ERROR IN STATEMENT SYNTAX
                157
0370: A6 F8     158 SYNTAX    LDX  REMSTK      ;RESTORE STACK FROM BEFORE THE &
0372: 9A        159           TXS
0373: A2 10     160           LDX  #16         ;SYNTAX ERROR CODE
0375: 4C 12 D4  161           JMP  ERROR       ;REPORT THE ERROR
                162
                163 *HERE WHEN ARRAY FOUND
                164
0378: A9 05     165 FOUND     LDA  #BSAVEMSG   ;SEND BSAVE TO DOS
037A: A0 03     166           LDY  #>BSAVEMSG
037C: 20 3A DB  167           JSR  STROUT
037F: A9 EA     168           LDA  #$EA        ;MODIFY CHRGET TO
0381: 85 C0     169           STA  $C0         ;ALLOW SPACES
0383: 85 C1     170           STA  $C1
0385: 20 B1 00  171 SENDFILE  JSR  CHRGET      ;GET FILENAME
0388: F0 11     172           BEQ  ENDSEND     ;AT TXTPTR
038A: C9 22     173           CMP  #$22        ;CMP TO QUOTE
038C: F0 08     174           BEQ  FINTXT
038E: 09 80     175           ORA  #$80        ;NEG ASCII
0390: 20 ED FD  176           JSR  COUT        ;AND SEND
0393: 18        177           CLC              ;TO DOS
0394: 90 EF     178           BCC  SENDFILE    ;UNCOND JMP
0396: 20 B1 00  179 FINTXT    JSR  CHRGET      ;FINISH OUT
0399: D0 FB     180           BNE  FINTXT      ;TXT TO EOL
039B: A9 F0     181 ENDSEND   LDA  #$F0        ;RESTORE CHRGET
039D: 85 C0     182           STA  $C0
039F: A9 EF     183           LDA  #$EF
03A1: 85 C1     184           STA  $C1
03A3: A9 AC     185 SENDADDR  LDA  #","        ;SEND COMMA
03A5: 20 ED FD  186           JSR  COUT
03A8: A9 C1     187           LDA  #"A"        ;SEND A
03AA: 20 ED FD  188           JSR  COUT
03AD: A5 08     189           LDA  ARYPTR      ;SEND ADDR VIA LINPRT
03AF: AA        190           TAX
03B0: A5 09     191           LDA  ARYPTR+1
03B2: 20 24 ED  192           JSR  LINPRT
03B5: A9 AC     193           LDA  #","        ;SEND ANOTHER COMMA
03B7: 20 ED FD  194           JSR  COUT
```

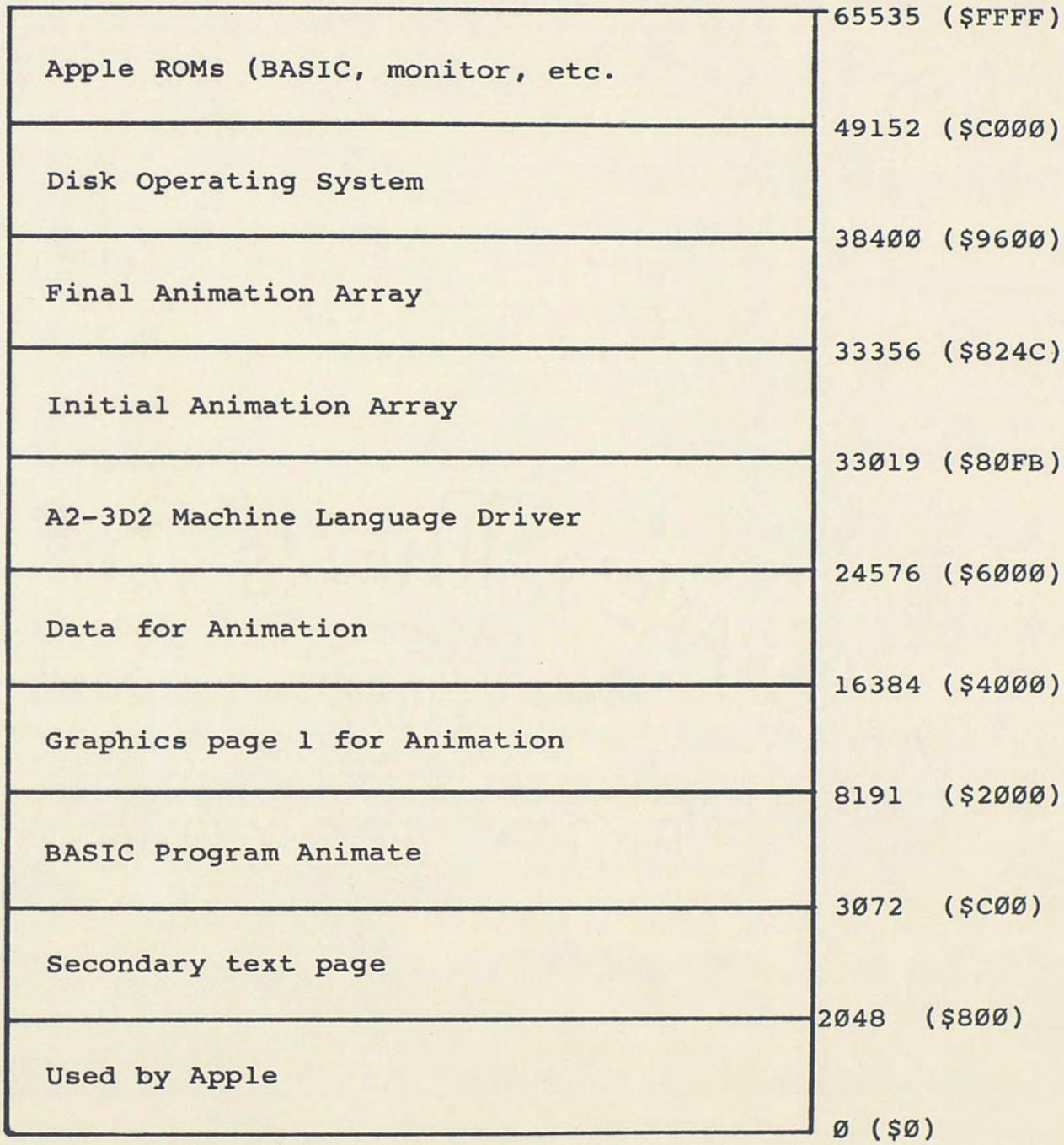
```
03BA: A9 CC    195          LDA  #$CC
03BC: 20 ED FD 196          JSR  COUT
03BF: A0 02    197          LDY  #2          ;SEND LENGTH VIA LINPRT
03C1: B1 08    198          LDA  (ARYPTR),Y
03C3: AA       199          TAX
03C4: C8       200          INY
03C5: B1 08    201          LDA  (ARYPTR),Y
03C7: 20 24 ED 202          JSR  LINPRT
03CA: 20 8E FD 203          JSR  CROUT       ;SEND CR
03CD: 4C 3F FF 204          JMP  RESTORE     ;RETURN VIA RESTORE
               205  *   ROUTINE TO RESTORE REGS.
               206
               207  *NOTE: CODE MUST END BEFORE $3D0
               208  *(DOS VECTORS).

--END ASSEMBLY--

ERRORS: 0

214 BYTES

SYMBOL TABLE - ALPHABETICAL ORDER:

   ARYPTR  =$08        ARYTAB  =$6B        BSAVEMSG=$0305      BSAVEPGM=$030E
   CHRGET  =$B1        CHRGOT  =$B7        CMPNAME =$0341      COUT    =$FDED
   CROUT   =$FD8E      ENDSEND =$039B   ?  ERRNUM  =$DE        ERROR   =$D412
   FINDARRY=$0337      FINDQUOT=$032E      FINTXT  =$0396      FOUND   =$0378
   LINPRT  =$ED24      NAME    =$06        NOMATCH =$036D      OFFSET  =$034E
   REMSTK  =$F8        RESTORE =$FF3F      SAVE    =$FF4A   ?  SENDADDR=$03A3
   SENDFILE=$0385      STREND  =$6D        STROUT  =$DB3A      SYNCHR  =$DEC0
   SYNTAX  =$0370      TEMP    =$EB

SYMBOL TABLE - NUMERICAL ORDER:

   NAME    =$06        ARYPTR  =$08        ARYTAB  =$6B        STREND  =$6D
   CHRGET  =$B1        CHRGOT  =$B7     ?  ERRNUM  =$DE        TEMP    =$EB
   REMSTK  =$F8        BSAVEMSG=$0305      BSAVEPGM=$030E      FINDQUOT=$032E
   FINDARRY=$0337      CMPNAME =$0341      OFFSET  =$034E      NOMATCH =$036D
   SYNTAX  =$0370      FOUND   =$0378      SENDFILE=$0385      FINTXT  =$0396
   ENDSEND =$039B   ?  SENDADDR=$03A3      ERROR   =$D412      STROUT  =$DB3A
   SYNCHR  =$DEC0      LINPRT  =$ED24      CROUT   =$FD8E      COUT    =$FDED
   RESTORE =$FF3F      SAVE    =$FF4A
```

APPENDIX E

MEMORY MAP AND ARRAY FOR SUBPROGRAM ANIMATE

SUBLOGIC DATA SHEET
(Courtesy Sublogic Corp., Champaign, IL)

MEMORY MAP OF SUBPROGRAM ANIMATE



INITIAL ANIMATION ARRAY

ARRAY LOC		COMMAND	DATA	COMMENTS
33019	$80FB	$07	83 ($53)	Mixed Graphics
33021	80FD	07	80 ($50)	Color Graphics
33023	80FF	07	87 ($57)	Hires Mode
33025	8101	07	84 ($54)	Disp Page 1
33027	8103	11	11	No Op
33029	8105	09	01 ($01)	Write P1
33031	8107	05	x,y,z,P,B,H	Viewpoint
33041	8111	OF(F.O.V.)	FF,5F,FF,7F,45,25	Field of View
33048	8118	14 (SRES)	01	(Set Hi Res 280X192)
33050	811A		RESERVED FOR	Output Array Control
33053	811D			Line DWG Control
33055	811F			Interpretive Jump
33058	8122	11		No Ops
33062	8126	SP 01	xyz A0	
33069	812D	CP 02	xyz AJ	
33076	8134	RP 03	xyz P	MECHANISM
33083	813B	CP 02	xyz BJ	SKELETON
33090	8142	RP 03	xyz P	
33097	8149	CP 02	xyz B0	
33104	8150	11		No Ops - RESERVED
33111	8157	11		
33118	815E	SP 01	0,0,0	
33125	8165	CP 02	x,y,z ƵM	
33132	816C	RP 03	xyz Ƶ4	Ƶ = Axis
33139	8173	RP 03	xyz Ƶ5	
33146	817A	SP 01	xyz Ƶ6	
33153	8181	CP 02	xyz Ƶ7	
33160	8188	CP 02	xyz Ƶ9	"Ƶ"
33167	818F	CP 02	xyz Ƶ8	
33174	8196	SP 01	xyz TA	
33181	819D	CP 02	xyz AN	UNIT VECTOR A
33188	81A4	RP 03	xyz AL	
33195	81AB	RP 03	xyz AR	

INITIAL ANIMATION ARRAY, CONTINUED

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

FINAL ANIMATION ARRAY (OUTPUT ARRAY)

ARRAY LOCATION	COMMAND	DATA	COMMENTS
33356 $824C (MEM)	06	2-D Lines	2D Lines For To Create Mechanism Skeleton For jth Position
MEM+26 - MEM+33	27 (81b)		No Ops
MEM+34			Pause Command
MEM+35	06		2D Lines To Erase jth Position Of Mechanism Skeleton
MEM+70	06	2-D Lines	Begin 2D Lines For $(J + 1)^{th}$ Position

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

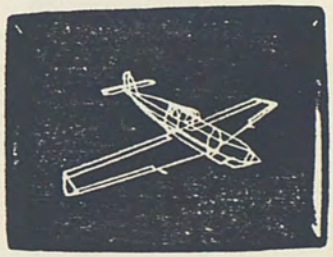
Hardware requirements
Apple II microcomputer and video monitor.

Memory Requirements
48K.

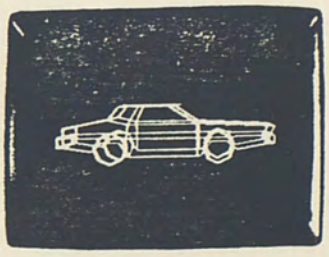
Software Requirements
A2-3D1.

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.

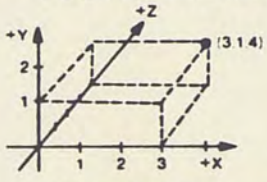


"Apple" is the registered trademark of Apple Computer Inc.

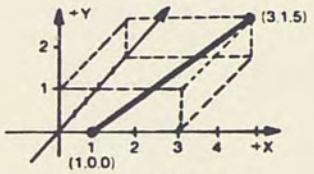
Description

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 . . .

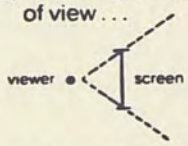


a) A point in 3D space



b) A line in 3D space

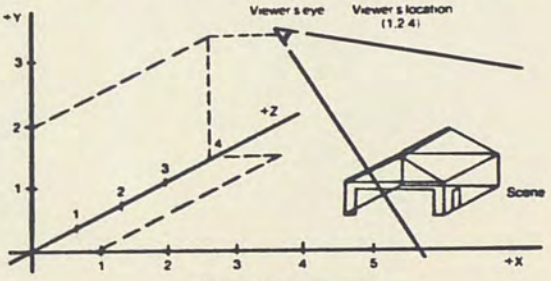
b) "Zoom" between wide-angle and telephoto fields of view . . .



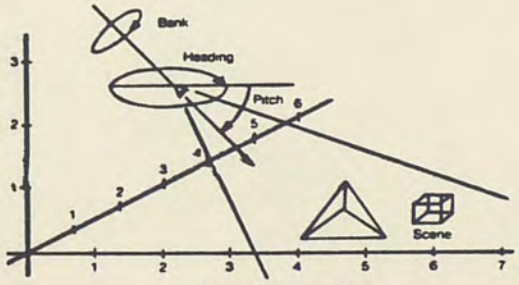
a) wide view AZR=32767

b) narrow view AZR=8192

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



Viewer's location in space



Viewer's 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 a ±32767 range, and you have full-circle viewing freedom of pitch, bank, and heading.

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

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

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

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

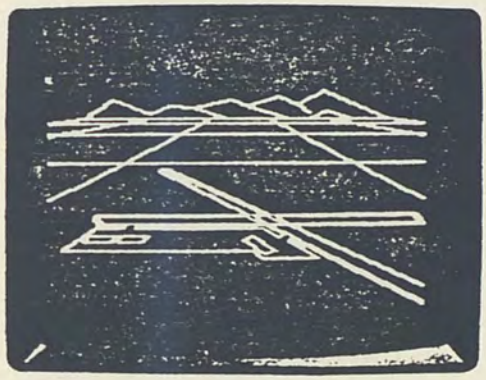
The **Load and Go Manual** guides you through an orientation session with the A2-3D1 program. Load the 3D-2D transformer, load the DEVELOP program, and view the scenes waiting for you. The manual will show you how to change location and direction of view one step at a time.

The load and go manual will also guide you through data base development by discussing how to create and enter your own data bases.

The **Technical Manual**, for advanced applications, describes the 3D-2D transformer algorithm in depth. Patchpoints and methods for hardware multiply, data input 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	A2-3D1	A2-3D2
Interpretative Functions		
Pure Point (140 x 192)	0,x,y,z	0,x,y,z
Start Point (140 x 192)	1,x,y,z	1,x,y,z
Continue Point (140 x 192)	2,x,y,z	2,x,y,z
Ray Point (140 x 192)	3,x,y,z	3,x,y,z
Clipper Control	4,on/off	4,on/off
Viewer Position D.P., Pseudodegrees	5,x,y,z,p,b,h	5,x,y,z,p,b,h
Draw 2D Line on Screen	6,x,y,x',y'	6,x,y,x',y'
Display Screen Select	7,code	7,code
Erase Screen / Fill Screen	8,code	8,code
Write Screen Select	9,code	9,code
Plot a 2D White Point	10,x,y	10,x,y
Interpretive Jump	11,adrlsb,adrmsb	11,adrlsb,adrmsb
Set Line Drawing Mode	12,mode	12,mode
Turn on Output Array	13,adrlsb,adrmsb	13,adrlsb,adrmsb
Screen Size Select	14,w,h,cx,cy	14,w,h,cx,cy
Field of View Select	15,axr,ayr,azr	15,axr,ayr,azr
Easy Initialize	16	16
No Operation	17	17
Set Color Mode	—	18,col
Independent Object Call	—	19,stat,loc,addr
Set Resolution	—	20,res
Hi-Res (280 x 192) Line 2D	—	21,xl,xh,y,xl,xh,y
Set Hi-Res Bias	—	22,xl,xh,y
Hi-Res (x=256 Limited) Line 2D	—	23,x,y,x',y'
Hi-Res (280 x 192) Point Plot 2D	—	24,xl,xh,y
Hi-Res (x=256 Limited) Point Plot 2D	—	25,x,y
Skip Segment	—	26,size,status
Pause for n/5ths of a Second	—	27,time
Set 3D to 3D Array Gen Address	—	28,adrlsb,adrmsb
Set 3D to 3D Array Gen. Status	—	29,status
End of File	undefined	121 (79 hex)
Callable Functions		
Sine/Cosine Calls	yes	yes
Multiply (SP and DP)	—	yes
Divide (DP)	—	yes
Erase (hi/low page)	—	yes
Hi-Res Point Plot	—	yes
Color Point Plot	—	yes
Hi-Res Line Draw	—	yes
Color Line Draw	—	yes
Set Display Resolution	—	yes
General Features		
Initialize Input Buffer Ptr.	—	yes
Line Length Limit	32767 max.	113508 max.*
World Movement	limited by overflow	unlimited**
Program Location(s)	2048+24576	24576
Program Length	4864 bytes	8443 bytes

*Distance from -32767, -32767, -32767 to 32767, 32767, 32767.
**As long as the value of -32768 is avoided in data bases and eye position.

Ordering Information

See your dealer or order directly from SubLOGIC. The A2-3D1 with A2-3D2 Enhancement is $84.90 on disk. You may update to A2-3D2 at any time. Contact SubLOGIC for details.

Shipping weight of the packages is approximately five pounds.

Shipping charges: US and Canada add $6.50 for first class mail, $3.00 for UPS, $4.50 for COD (UPS) orders. Foreign add $20.00 ($27.00 Australia) for airmail. Illinois residents add 5% sales tax.

MasterCard and Visa accepted.

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

1. Krouse, J. K. "Designing Mechanisms on a Personal Computer," Machine Design, No. 6 (March, 1983), pp. 94-99.

2. Users Guide to ADAMS. 5th ed. Ann Arbor, Mich.: Mechanical Dynamics Inc., 1981.

3. Users Guide to DRAM. 5th ed. Ann Arbor, Mich.: Mechanical Dynamics Inc., 1978.

4. Paul, B. "Dynamic Analysis of Machinery Via Program DYMAC." International Automotive Engineering Congress and Exposition, Detroit, Michigan, 1977, Society of Automotive Engineers Paper No. 770049.

5. Sheth, P. N. and Uicker, J. J. "IMP (Integrated Mechanism Program), A Computer-Aided Design System for Mechanisms and Linkages." American Society of Mechanical Engineers, Journal of Engineering for Industry, 1972, pp. 454-464.

6. Knight, F. L. "Computer-Aided Design and Analysis of Mechanisms." NASA, 16th Aerospace Mechanisms Symposium, Kennedy Space Center, Florida, 1982, NASA Conference Publication 2221, Available from the National Technical Information Service, Springfield, Va, pp. 175-187.

7. Ardayfio, D. D. "Synthesis Techniques for Spatial Four-Bar Mechanisms - A Status Report." ASME Paper No. 82-DET-20, 1982.

8. Suh, C. H., and Radcliffe, C. W. Kinematics and Mechanisms Design. New York: John Wiley and Sons, 1978.

9. Einstein, S. A., and Goodrow, D. S. Expediter User Manual. Coarsegold, Calif.: On-Line Systems, 1981.

10. Artwick, B. Sublogic A2-3D2 Extended Graphics Animation Package Technical Manual. Champaign, Ill.: Sublogic Company, 1981.